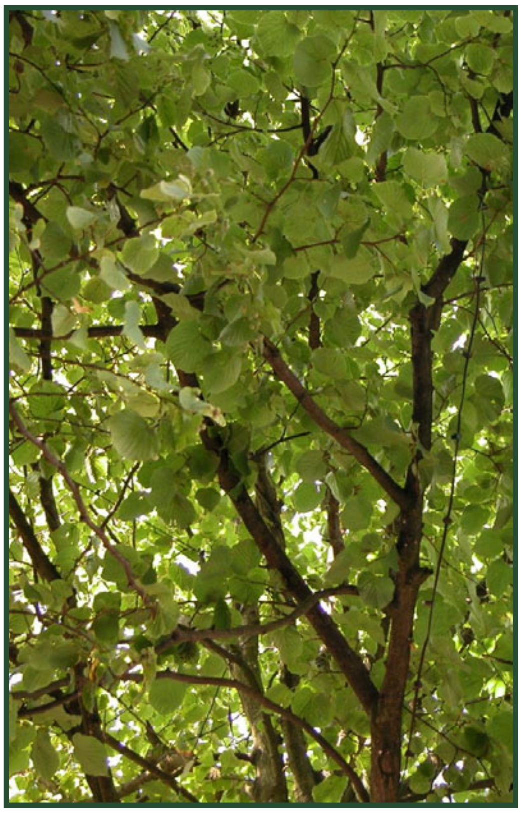
IntelliForest
Poznan University of Technology, Poland

Final Report



Team Members

Piotr Holubowicz
piotrekhol@wp.pl

Lukasz Langa
lukasz.langa@gmail.com

Pawel Lichocki
pawel.lichocki@gmail.com

Szymon Wasik
szymon.wasik@cs.put.poznan.pl

Mentor

Mikolaj Sobczak, Ph. D.
mikolaj.sobczak@cs.put.poznan.pl

CSIDC 2006

1 Abstract

Forests are the most complex ecosystems in the world. They cover 3.9 billion hectares, which is almost a third of the earth's land surface excluding Antarctica and Greenland, and are home to more than 50% of the world's plant and animal species [1]. In the times of massively growing human impact on the environment, more and more people understand the need to protect them. Their devotion cannot be left without support.

Considering this matter, we developed an idea of IntelliForest. It is a network-centric system, which core is a complex network of sensors and routers called **IntelliNet**. IntelliNet works as a platform for gathering and sharing data about the state of the forest. The system may then be extended with the use of any of the wide range of **endpoints**. The most important endpoint is the Management System, which is capable of storing and analyzing data as well as processing user requests received over the Internet. A mobile endpoint called the LEAF enables users to control the system on the spot, whereas endpoints like Mobile Motes, Unmanned Aerial Vehicles (UAV) or Information Points serve as sources or consumers for additional information.

IntelliForest is designed to be highly customizable so as to be able to help various types of users. A forester, warned by the system that illegal logging might be happening, will send an UAV to monitor the situation. An ecologist investigating the influence of air pollution on growth of plants will place appropriate sensors according to his needs and will have the measurements done automatically. A tourist visiting a wild forest will be able to recognize local animals by looks or voice after getting appropriate hints at the Information Point. A fire brigade will receive a fire alarm after smoke is detected in a deep part of the forest.

After consulting with specialists from Poznan Agriculture Academy and Poznan Greenhouse, we formulated the following objectives for the system:

- The system must be **modular** so that it is easy for individuals to adjust it to their needs.
- The system must be **extensible** so that the users may integrate their applications or endpoints with it.
- The system must be **scalable** so that it can be deployed in small parks as well as in large forest areas.
- The system must support **sharing the information**, which is particularly important in the knowledge-based societies.
- The system must support **accounting**, which is necessary for making the services available commercially.

In this report, we present the ways in which IntelliForest addresses these demands. We concentrate on the core network, IntelliNet, and discuss minutely the software and hardware solutions that give it the full functionality. In the next chapters we describe in detail the most powerful and representative of endpoints, namely the Management System and the LEAF, equipped with IntelliMap. Finally, we analyze the implementation issues of endpoints such as UAV, Information Points and Mobile Motes.

2 System overview

2.1 System architecture

The centre of IntelliForest is IntelliNet, a wireless network that collects and transmits the data. There is a variety of endpoints that may be connected to IntelliNet. The Management System, capable of storing and analyzing the data, uses the Internet and a dedicated Gateway as a proxy. Most endpoints, however, are used locally and for this reason are able to communicate directly with IntelliNet over radio. The main goal of some endpoints, like LEAF and Information Point, is to retrieve data from the system and make use of them. Others, like Mobile Motes, UAV or additional sensors, serve as sources of supplementary data.

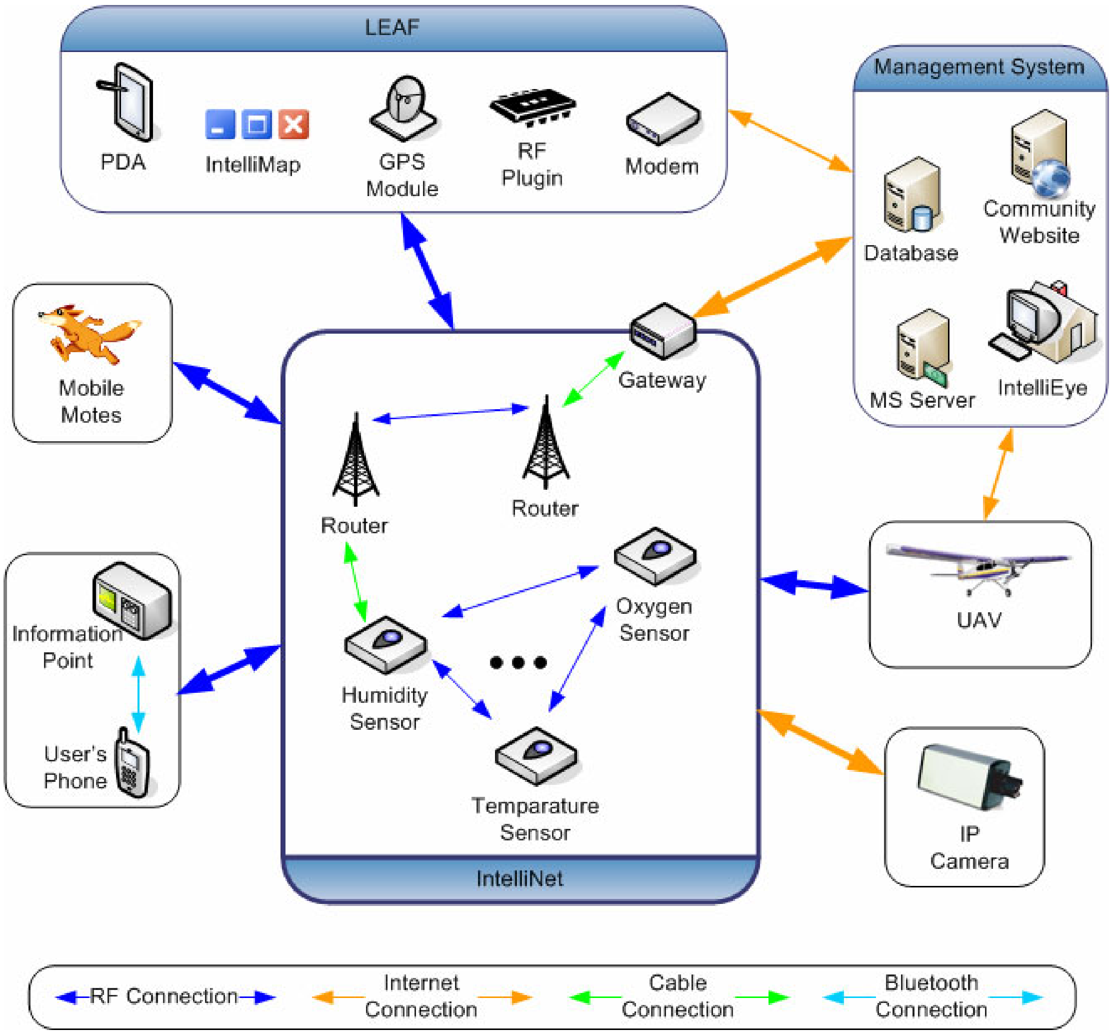


Fig. 2.1 IntelliForest architecture

2.2 Performance requirements

IntelliForest is meant to be useful for people whose work concerns forest in order to make this work easier. This means that the system must be **reliable**, so that it does not require unnecessary attention from its administrators and no information is lost in case of any failure. It also requires the system to be **secure**, immune to possible abuses.

A characteristic of IntelliForest is its variety of applications. It may be used for different purposes and may operate in diverse environments. Therefore, the system needs to be **flexible**. Covering larger areas should not influence its performance unreasonably. Its architecture should be modular, so that adjusting it to individual needs is easy and not costly.

2.3 Design methodology

We decided to use the **Feature-Driven Development** (FDD) methodology [2,3] to create IntelliForest. Taking into consideration the complexity of our system and the limited amount of time designated for its development, we particularly valued the idea of short, feature-driven iterations that compose the overall implementation.

The first two processes of FDD resulted in defining the concept of IntelliForest and its basic architecture. In this stage we consulted specialists from the Cathedral of Dendrometry at Poznan Agriculture Academy and we had numerous brainstorming sessions. In the 'Plan By Feature' stage we distributed different feature sets among ourselves and prepared a development plan, which resulted in a precise Gantt chart. **Lukasz** was responsible for the Management System part, **Szymon** worked on LEAF while **Pawel** and **Piotr** created the IntelliNet team. Finally, in the 'Design By Feature' and 'Build By Feature' phases we implemented the relevant features step by step. The first stage guaranteed that, unlike in traditional methodologies, only features actually implemented were designed in detail.

Testing was performed simultaneously with implementation; additional integration tests were executed after the completion of every major feature set.

2.4 Innovation

The innovative concept of IntelliForest is to use a specifically designed network of sensors as a universal platform for a variety of applications. The network – IntelliNet – is heterogeneous and comprises two layers and three types of devices. This distinguishes it from typical sensor nets, making it **highly scalable**, and yet preserving the flexibility and accuracy of information typical of sensor nets. IntelliNet, due to its specific structure and dedicated communication algorithms, is characterized by an **uncommonly long lifetime** of its nodes.

However, the true novelty of IntelliForest lies not only in IntelliNet, but rather in using IntelliNet as a source of information, concerning the environment, but serving different people and different purposes. We designed a set of endpoints that **enrich** IntelliFor est. The Management System along with IntelliEye allows processing the data, their visualization and analysis, up to decision supporting. The LEAF with IntelliMap gives users mobility while Information Points function as interfaces for the public. In fact, any endpoint able to communicate with IntelliNet may gather data or make use of them, enhancing the IntelliForest and becoming part of it.

Overall, the innovative possibilities offered by our system include:

- Automatic large-scale data collecting. Before IntelliForest, many types of measurement were made manually, which strongly limited their range and increased the costs.
- Data storage with support for their analysis. IntelliForest is able to detect critical situations and react appropriately, whereas with the use of extensible tool like IntelliEye, the data can be just the starting point for advanced analysis.
- Flexibility and extendibility. There is hardly a limit for sensors that collect data, nor is there for the applications that use them.
- Both local and global field of activity. Gathered information, apart from its collective significance, may be easily used locally. An example of that may be a LEAF equipped with IntelliMap, giving information about local air pollution or warning about low forest bed humidity.

3 Implementation and engineering considerations

3.1 The IntelliNet

The IntelliNet comprises two layers. **The lower layer** consists of nets of sensors which are organized in a topology of a tree. The root of the tree is usually connected to a router. Routers, equipped with long range RF transmitters, create **the upper layer** of IntelliNet. Routers also implement the tree topology, with the root connected to the Management System's Gateway.

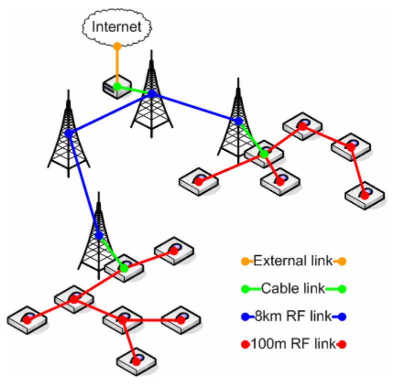


Fig. 3.1 IntelliNet architecture

Data are generated by sensors and sent to appropriate destination, most typically the Management System. Feedback and data coming from other endpoints may need to be transmitted between any two nodes of IntelliNet. Packets are transmitted by sensors and the router network.

The Gateway's concept of work is related strictly to the Management System it cooperates with, and thus it is described in the Management System section (see 3.2).

3.1.1 Hardware

Assumptions

Every device must be characterized by low power consumption and RF communication ability. Sensors should be equipped with a short range radio transceiver, and routers with a long range one. Additionally, all nodes must provide at least 6 kB of persistent memory in order to implement data buffering. Sensors must also be able to perform the measurements.

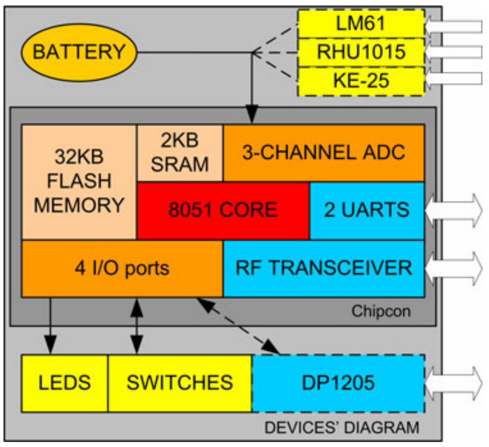


Fig. 3.2 Node general model

IntelliForest sensors' and routers' design

We found that **Chipcon's single-chip UHF transceiver** with integrated 8051 microcontroller CC1010 [4] suits our demands perfectly. Its short-range radio (up to 100 meters) ideally fits the lower layer, but is insufficient to work well in the upper layer. For this reason, routers will be extended with Semtech DP1205 transceiver modules [5], which operate on the same frequencies as Chipcon. Semtech has an advantage of a range of up to 8 km which enables IntelliNet to cover a much larger area.

Furthermore, Chipcon has 32 kB of integrated Flash memory and hardware support for DES encryption. It is also equipped with a 3-channel analog-digital converter and 4 digital I/O ports, all of which allow connecting various analog or digital **peripherals**. We use this possibility to extend the Chipcon with different sensing options, which is the crucial ability of IntelliNet nodes. The list of exemplary sensing elements is presented in the table below. It must be emphasized that modularity of the IntelliNet makes the list of potential sensors endless and simplifies adding new ones in the future.

	LM61 is a low-power **temperature** sensor available in a 3-pin SOT-23 package.		**RHU1015** is a precise low-cost **humidity** sensor.		**KE-25** is an **oxygen** galvanic cell type sensor. It requires no power supply and has a life expectancy of approx. 5 years.

In addition to that, Chipcon is able to measure the power of received radio signal. The accurate battery state check may be performed with a use of a Dallas Semiconductor DS2438 Smart Battery Monitor. All peripherals must be turned off in sleep mode. This happens automatically in case of low-power modules powered directly from Chipcon's outputs. If this is not the case, the use of a simple transistor key is required.

Power supply

When deciding on the **power supply** we considered: nominal voltage, maximum current draw, nominal capacity, size, operating temperature range. Finally we decided to use lithium-thionyl chloride cells manufactured by Saft: LS33600 (size D) and LS14500 (size AA). They both suit our requirements best and differ mainly in capacity and size, so a tradeoff between these factors may be made easily.

Prototype development

In order to optimize the effort put into building the prototype devices, we decided not to construct ready-to-sell version of nodes. Instead, we use Chipcon Evaluation Modules along with specially designed boards (see figure 3.3), which connected to the modules enhance their functionalities significantly. Each of the boards has four LEDs, two switches of general

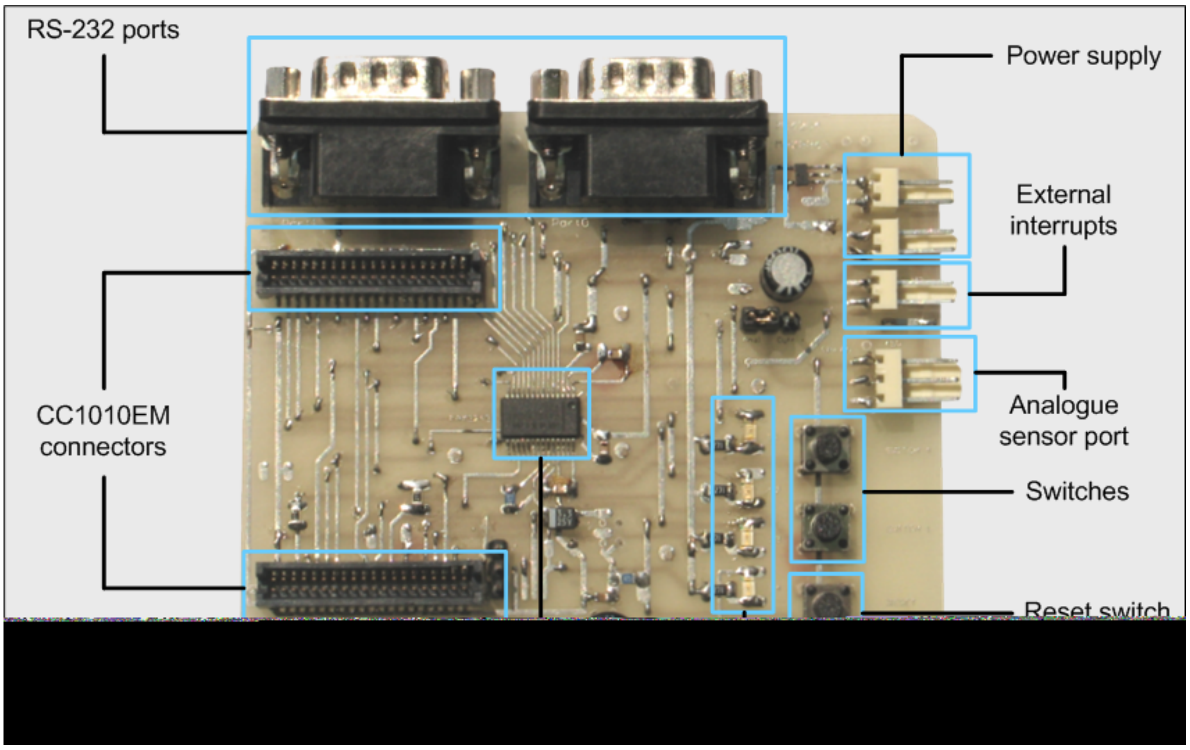


Fig. 3.3 The node

use, one reset switch, two RS-232 ports, one analogue port, and two inputs for power supply (one for 3V and one for 3V-10V). This way, we have obtained a general platform for testing, debugging and developing purposes.

Next step in creating the prototype was enhancing the devices with sensing options. The temperature meter LM61 is already placed on Chipcon Evaluation Module. Other sensors may be connected via analog or digital port to the board. Figure 3.4 presents the connection of all peripherals (RHU1015 humidity sensor, batteries) and Chipcon module to the board.

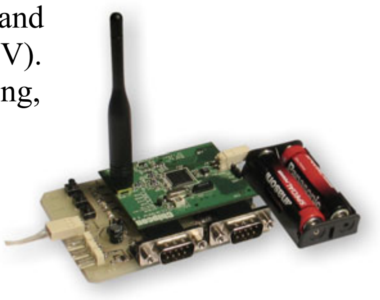


Fig. 3.4 The node with peripherals

Finally, the batteries and packaging issues had to be solved. Although we decided to use the SAFT batteries, because of their very high quality and good properties, the prototype nodes – for ease of the development process – are powered with typical alkaline batteries. As for the package, we use waterproof plastic boxes, which are commonly available.

3.1.2 Firmware assumptions and objectives

When designing the IntelliNet communication protocols we considered the following facts:

- All devices in IntelliNet (except for the Gateway) are battery-operated.
- Although most of the data gathered are not time crucial (long-term measurements), certain data types require fast reaction of the system.
- Very often environment conditions do not change rapidly (inertia of several minutes and more), however detection of certain events must be performed with maximal temporal accuracy.

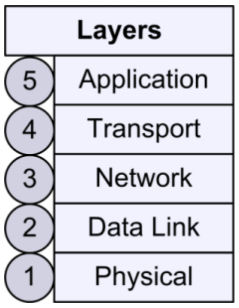


Fig. 3.5 Tannenbaum network model

This led us to formulating the following objectives for the communication protocols, supported by the concept of the S-MAC protocol [6]:

- Every node must be in **sleep mode** for vast majority of time.
- Latency can be traded off for power saving.
- The amount of traffic can be traded off for response time, depending on data type. Whenever it is possible, the traffic should be limited (data buffered and aggregated) to save on unnecessary radio activity.

As a guideline for the communication protocols design, we chose the reference model suggested by A. S. Tannenbaum [7] (see figure 3.5 and sections 3.1.3 through 3.1.7).

3.1.3 Physical layer

Physical layer is hardware-dependent. IntelliNet devices use CC1010 for radio communication. This involves the use of Frequency Shift Keying and Non-Return to Zero or Manchester encoding [7]. The upper and lower layers IntelliNet operate on different frequencies within the unlicensed ISM band (868/869 MHz in Europe, 915/916 MHz in the USA [8]).

3.1.4 Data link layer

In the link layer, we use the Simple Packet Protocol (SPP) [9] implemented by Chipcon. It offers unicast and broadcast, as well as Automatic Repeat Request (ARQ), all of which reduce energy waste. Additionally, it handles error detection (CRC). On top of SPP, we designed our own communication protocols including time synchronization, periodic listen and sleep, message passing and automatic reconfiguration.

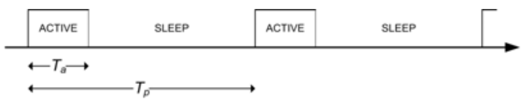


Fig. 3.6 Basic active-sleep timeline

The basic scheme – synchronized master-slave communication

Every node wakes up at fixed intervals T_P. It remains active for T_A time period and then goes back into the sleep mode. Since the sleep time is much longer than the active time, every pair of potential sender and receiver must be synchronized. To solve this problem, the S-MAC algorithm [6] introduces the idea of virtual clusters, which are groups of nodes maintaining their own synchronization schedule. This concept, dedicated for flat topology networks, ensures flexibility for the price of greater power consumption on the edges of clusters. IntelliNet, however, is designed to fulfill different requirements, and these are accomplished better by using the S-MAC algorithm as a base for the synchronized master-slave communication.

In order to maintain synchronization with its neighbors, every node must either accept a synchronization schedule of one of its nodes, or keep one independent from others. Once forcing a schedule, a node cannot later adopt it indirectly, as this leads to instability. As a result, nodes of one subnet are organized into a topology of a tree, with the root determining the synchronization schedule. For the upper layer, the root is connected directly to the Gateway. For the lower layer, in each subnet the root sensor is connected directly to the accompanying router. Every node is serves role of a master and a slave, alternately.

Figure 3.7 presents the basic communication scheme. Every T_{SYN} time, the master broadcasts a synchronization message. Next, the communication phase begins. The `RTS`/`CTS` protocol solves the **hidden terminal problem** [6]. Then, the packet is transmitted and acknowledged. Both nodes suspend entering the sleep mode until transmission is finished.

Every node listens for random time before starting transmission, or listens for maximum time if it does not intend to transmit anything. The node that stops listening first, wins the medium (Slave B). It issues its `RTS` message and receiver responds with `CTS` (Master A), canceling its transmission, if it was been planned. A message sent by the master (either `RTS` or `CTS`) is received by all slaves competing for the medium and all but the one participating in transmission terminate their radio activity and advance to the work phase (Slave C).

To eliminate collision between the lower and upper IntelliNet layers, respective devices operate on different frequencies (see 3.1.3). To minimize the risk of collision within one layer, every master forces its own synchronization schedule, independent of all its neighbors. If a node fails to send its packet, it waits random time before resending it to prevent deadlock.

Every node has a 16-bit MAC address assigned, ensuring correct identification of nodes.

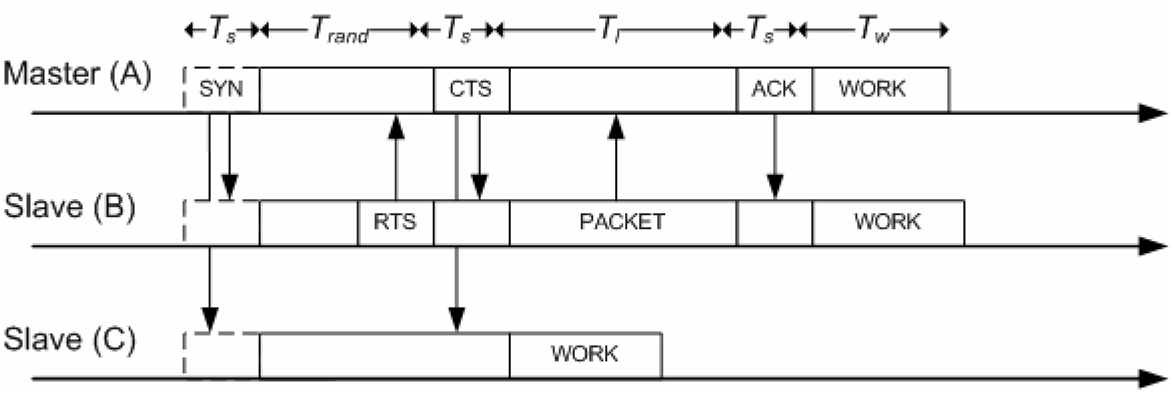


Fig. 3.7 T_S = 15-20 *ms*, T_L = 15-242 *ms*, T_M = 100 *ms*, T_W = 5-30 *ms*

Adding new nodes to the network

Specific procedures apply when connecting new nodes to the network. When a router or sensor is first turned on, it enters **LFM** (Look For Master) phase, during which it listens T_{SYN} time for `SYN` messages of the devices in proximity. Once it gets some messages, it processes the nodes' characteristics and synchronizes with the node that fits it best. This decision is based on three factors: the master's distance to the root of the subnet, its current number of slaves and the strength of its signal. Within $2 \cdot T_{SYN}$ time the device is connected to the network. Random time after a successful LFM phase, nodes perform **LFS** (Look For Slaves) phase: they broadcasts `NEW` messages for 1 period, making all so far standalone nodes listen for the next `SYN`. In this way all so far standalone nodes become connected.

Initially, we considered LFM to be an active phase, like LFS. However, after calculating the optimal synchronization period T_{SYN} we found that listening T_{SYN} time costs an amount of energy comparable to the energy used for broadcasting packets for 1 period and listening during the next one. Finally, we decided on the passive version because it is more flexible – there is no limitation of neighboring nodes entering LFM phase simultaneously.

Sleep time determination

In the first design of IntelliNet the sleep time was assumed to be 10 seconds for the lower layer and 7.5 seconds for the upper layer. Later however, the lower layer evolved and in the end was implemented similarly to the upper one. Furthermore, after consultation and meetings with potential end-users (scientist from Agricultural University, Poznan Greenhouse and foresters) it turned out that their needs vary highly and a tradeoff between communication latency and nodes' batteries lifetime should be tuned to each client respectively. In long term scientific research of natural habitat environment conditions, for instance, nodes' lifetime is much more important then the latency of data flow. In other cases, when batteries changes are easy to be made, it is sufficient that nodes last for a year, which allows shortening the latency. For these reasons it was desired to provide a mechanism of flexible sleep time determination. The value of the sleep time influences T_{SYN} (synchronization period), and both of them combined together along with the net load have major impact on the estimation of nodes' lifetime. For these reasons it was desired to check the influence of sleep time on the T_{SYN} time first.

Maintaining the synchronization

In order to prevent long-term clock drifting nodes have to periodically synchronize their clocks. To determine the synchronization period's length T_{SYN}, we considered quartz frequency tolerance of $\theta = 30\,ppm$ [10,11]. For every minute, the clock may be T_E late or early:

$$T_E = \theta \cdot l = 3 \cdot 10^{-5} \cdot 60\,s = 1.8 \cdot 10^{-3}\,s = 1.8\,ms$$

This results in $2 \cdot T_E$ of relative clock difference. Therefore, each device must increase its communication period by $4 \cdot T_E = 7.2\,ms$ ($2 \cdot T_E$ of preamble and $2 \cdot T_E$ of postamble time).

We checked how the changes of sleep time influence the T_{SYN} time in the lower layer. Graphs

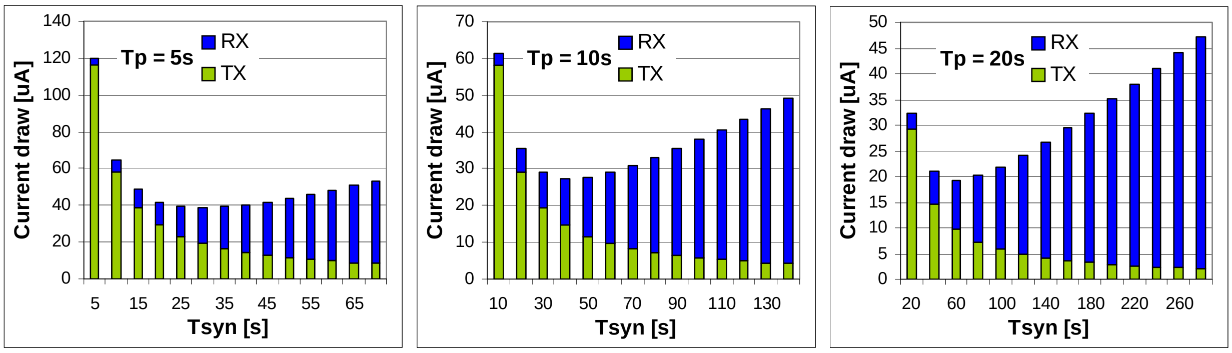


Fig. 3.8 T_{SYN} power waste for different T_P (lower layer)

in figure 3.8 present the calculation of the additional power consumption caused by adopting different synchronization periods T_{SYN}. TX stands for the energy used for sending `SYN`, while RX is the energy wasted on listening for additional time, increasing by $4 \cdot T_E$ every minute. The results obtained show $\boldsymbol{T_{SYN} = 30\,s}$ for $T_P = 5$ s, $\boldsymbol{T_{SYN} = 40\,s}$ for $T_P = 10$ s and $\boldsymbol{T_{SYN} = 60\,s}$ for $T_P = 20$ s to be the best values. Overall, it turns out that an additional current draw used for maintaining the synchronization is equivalent to a constant current draw of no more than 40 uA. This result was taken into account in power consumption calculations.

Power consumption calculations

We assumed the active time T_A to be **250 ms** on average (as the resultant of the data link layer communication protocols and packet transfer time). The current draw during active phase is approx. **25 mA** for sensors and **35 mA** for routers (as the resultant of different current draws while listening, transmitting and having radio off). In the idle phase the current draw is approx. **50 uA** (including the energy used for maintaining the synchronization).

	Sensors	Routers
T_P	5s, 10s, 30s, 45s	5s, 10s, 30s, 45s
T_A	Min: $T_A = 120\,ms$ Max: $T_A = 442\,ms$ Average: $T_A = 250\,ms$	

As a result, for exemplary period length $T_P = 10$ s the expected lifetime of a sensor is **2.56 years,** and **1.87 years**[1] for a router. The latency on a single transmission between neighboring nodes is equal to T_P, in this case 10 s. The total delay (between the sensor at the bottom of IntelliNet and Gateway) will be up to several minutes depending of the IntelliNet topology (depth of the tree). Analyzing the predicted lifetimes, we reached the conclusion that T_P should be between 5 and 20 seconds, ensuring lifetime of one up to six years, still preserving acceptable latencies of data transfer.

Figure 3.9 presents expected lifetime of a sensor and router depending on length of T_P.

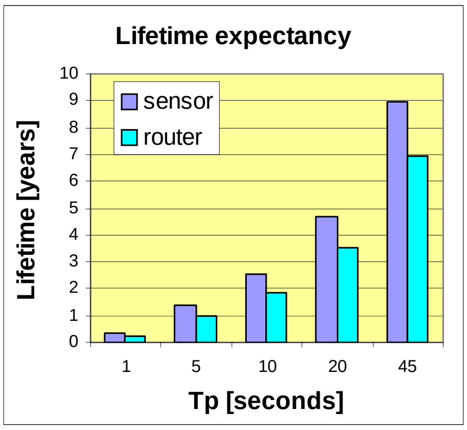


Fig. 3.9 Lifetime expectancy

Security:

To ensure security, packets transmitted between every pair of nodes are encrypted. For the encryption process, 2 algorithms are used:

- **Diffie-Hellman** algorithm [30] is used to establish common key. Every node generates its private and public keys at the beginning and exchanges the public keys when starting communication with another node. On the base of the partner's public key, common key is calculated and stored.
- **3DES** algorithm [7] is used to encrypt the packets. Its key is obtained by the Diffie-Hellman algorithm described above. 3DES is supported by the Chipcon hardware.

3.1.5 Network layer

LFM and LFS procedures applied to Routers provide them with automatic reconfiguration possibility (see example below).

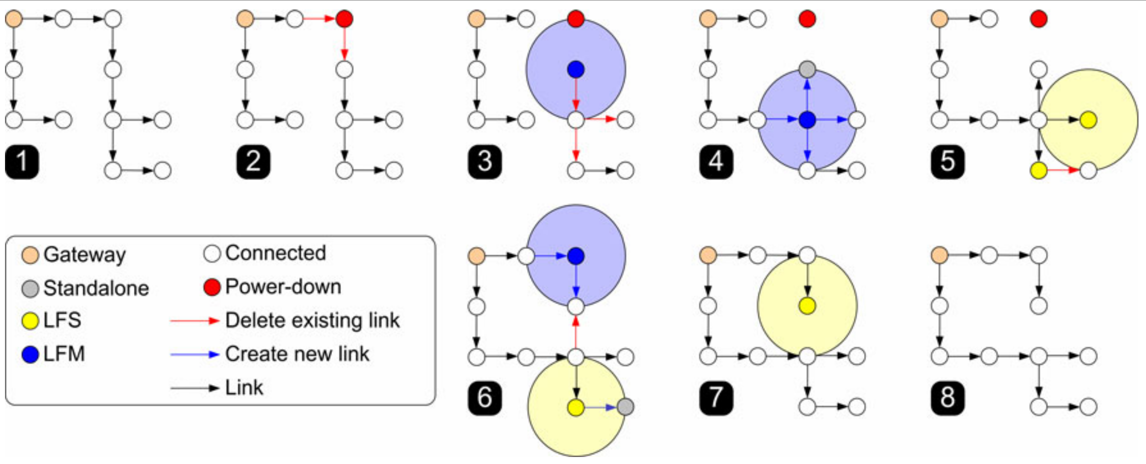
When designing the routing algorithm for IntelliNet, we found that most of the ad-hoc algorithms, like AODV or DSR [12], do not assume any specific network topology. In IntelliNet, however, the synchronization requirement forces the tree topology with a master-slave scheme. Moreover, IntelliNet is expected to change topology relatively rarely, so the speed of establishing new routes is not crucial. Taking that into consideration we decided to implement a table driven link-state routing algorithm – **a modified version of RIP algorithm**. RIP is characterized by limited information about the network topology stored in each node and relatively long convergence time, both of which fit to the IntelliNet concept. Additionally, we accepted updating the routing tables regularly and after topology changes, but we adapted this feature to our needs. We resigned from connectionless routing table broadcasts (UDP in RIP) in favor of automatic acknowledgement provided by the data link layer of IntelliNet. This way, only modifications of routing tables need to be sent, which allows for saving on the amount of communication.

Every node knows addresses of nodes in the sub-trees of each of its slaves. This information is sufficient for the node to be able to route any message down or up the tree. This means, in particular, that nodes keep information only about a subset of nodes from their own layer. In

[1] We assumed the use of LS33600 battery with 80% efficiency [17,18]

order to expand the routing algorithm onto many sensor nets, and even many IntelliNets, we designed a special **IntelliNet Node Address** – INA. It consists of 3 parts – address of IntelliNet, address of router (and its sub-network) and the node's MAC address. First 2 parts are determined dynamically when the node connects to the network, and may be changed later in case of reconnection. The latter part is unique for the whole system and guarantees that no two INA addresses appear twice. Using the INA addresses, gateways and routers can route the packets without detailed knowledge of the IntelliNet structure.

Finally, regular routing table updates and synchronization messages create a cohesive mechanism of detecting the link state. Repetitive failure to receive a synchronization message causes the node to disconnect from its master, and failure to receive certain number of consecutive updates makes the node delete the slave from its routing table.



In a working network (1), suddenly one router turns off (2). Soon, its slave notices that and attempts to find another master (3). It fails, and being standalone it stops synchronizing its slaves. Soon its slave starts its own LFM attempt (4). Having connected to a new master, it adopts its synchronization schedule and reconnects its slaves (5) using LFS. The reconnection phase advances (5, 6), while at the same time the powered-off router turns on again. It reconnects and starts LFS phase. It offers a shorter path to an already connected router and, as a result, intercepts it (7). In the end, the network is in an optimal configuration (8).

3.1.6 Transport layer

The nature of application developed for the nodes assumes that every message that is transmitted fits exactly into one data packet of the data link layer. Therefore, there is no need to deal with additional encapsulation, which is normally done in the transport layer. Furthermore, considering the reliable links provided by the data layer, and aiming at reducing the amount of traffic in IntelliNet, we found that implementing the connectionless protocol of transmission is sufficient to comply with the objectives of the system.

3.1.7 Application layer

Three basic groups of application data can be distinguished: measurement results (produced by sensors), scheduled times of measurements (delivered to sensors) and configuration data that carry requests, settings modifications, etc. Detailed specification of these formats can be found in Protocol Specification chapter (see 3.1.8).

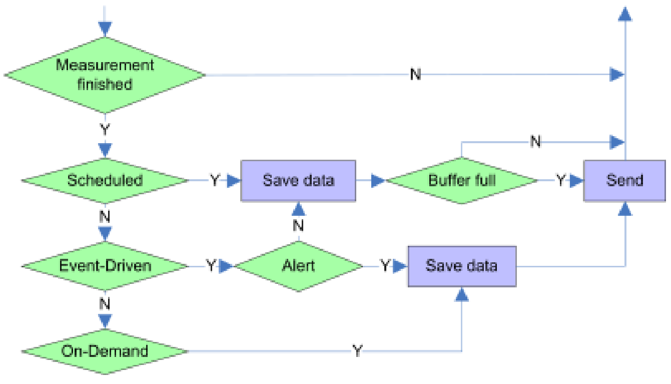


Fig. 3.10 Sensor measurement management

There is a concept of **virtual time** within IntelliNet. It is set by the Gateway and forwarded using the SYN messages. It is counted in seconds and is used to determine the times of measurements, in particular the scheduled and on-demand ones.

The measurement scheme

Three different types of measurements are performed. **On-demand** is an on-line way of requesting and receiving results immediately. This scheme is enriched by **scheduled measurements**, which are made at explicit moments in time, defined by the user. Finally, there are sensors that detect and identify **events**. If required, each single measurement is checked against **alert conditions**, for example, if max or min thresholds were exceeded. In case of an on-demand request or an alert data are sent immediately. Otherwise results are aggregated and buffered in order to maximally utilize the 232 bytes of the message packet.

3.1.8 Protocol specification

```
type FRAME is record of
      tag: byte
      sId: long
      rId: long
end of record

//data link data
type SYN extends FRAME is record of
      time: long
      numberOfSlaves: byte
      hopsToGw: byte
end of record

type RTS extends FRAME is record of
      rtsId: long
end of record

type PACKET extends FRAME is record of
      data: MESSAGE
end of record

type MOBILE extends FRAME is record of
      name: char[4]
end of record

type CTS extends FRAME is record of
end of record

type ACK extends FRAME is record of
end of record

type NEW extends FRAME is record of
end of record

//network data
type UPDATE extends FRAME is record of
      numberOfElements: byte
      changes: array[1..n] of UPD_EL
end of record

type UPD_EL is record of
      ID: long
      action: byte
end of record
```

```
//application data
type MESSAGE extends FRAME is record of
end of record

type TIMES extends MESSAGE is record of
      delay: long
      period: long
      numberOfElements: byte
      times: array[1..n] of TIME
end of record

type CONFIGS extends MESSAGE is record
of
      numberOfElements: byte
      elements: array[1..n] of
CONFIG
end of record

type RESULTS extends MESSAGE is record
of
      numberOfElements: byte
      results: array[1..n] of RESULT
end of record

type TIME is record of
      value: long
end of record

type RESULT is record of
      type: byte
      time: long
      value: long / float
end of record

type CONFIG is record of
      code: byte
      value: long / float
end of record
```

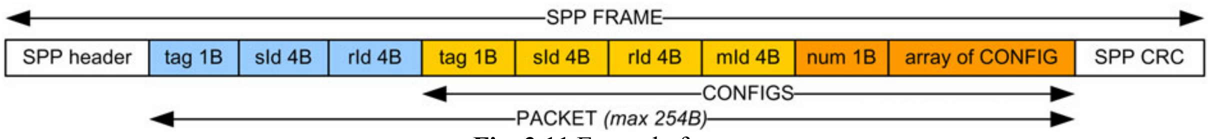


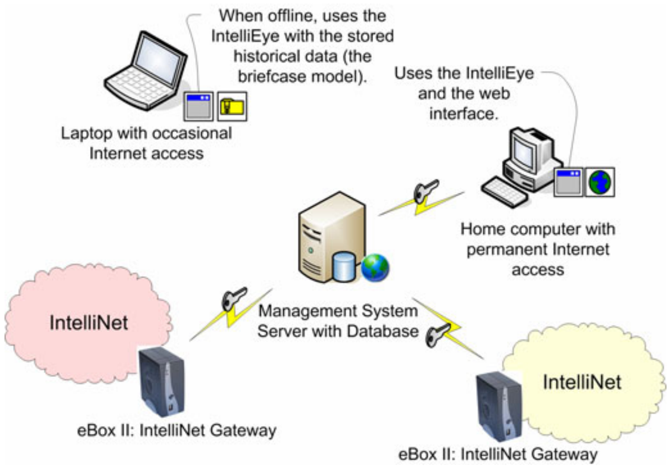
Fig. 3.11 Example frame

3.1.9 Tools used

When developing the IntelliNet, we used the Keil compiler with libraries provided by Chipcon. For the purposes of project management and versioning, TortoiseSVN was used.

3.2 Management System

The Management System (MS) is the most important endpoint of IntelliForest. It enables remote users to view the current state of the forest and analyze historical data. The Management System also allows user interaction with IntelliForest such as sending requests for on-demand measurements, setting thresholds for alerts, inputting user feedback on various forest places and events.



When offline, uses the IntelliEye with the stored historical data (the briefcase model).
Uses the IntelliEye and the web interface.
Laptop with occasional Internet access
Home computer with permanent Internet access
IntelliNet
Management System Server with Database
IntelliNet
eBox II: IntelliNet Gateway
eBox II: IntelliNet Gateway

Fig. 3.12 Management System architecture

We decided to build the MS using a managed framework. We did research on various frameworks and architectures, such as J2EE, PHP, Zope or CherryPy, but they were rejected and .NET has been chosen instead. The main advantages of .NET are:

- **seamless integration** of various parts of the system: large parts of source code can be used in multiple parts of MS. For instance, MS is using its own protocol with encryption, authentication and connection states. The source code for the network part is common for all parts of Management System, even embedded devices using .NET Compact Framework like Gateways. There are also parts of the source code shared between Management System and LEAF (IntelliNet protocol classes, the map component, etc.).
- **highest performance** of all major managed application frameworks: there is a significant amount of processor time needed for managing encrypted communication (see 3.2.5), data manipulation and storage, etc. .NET is performing well enough to handle encryption also within the least powerful parts of the MS (the Gateways).
- powerful and easy client-side software **plugin implementation** with IronPython. IntelliForest provides a lot of ways to measure every aspect of the forest. Thanks to IntelliEye plugins, it is not limited for application domains originally covered by the system developers.

The Management System consists of three parts: Management System Server software with a backend database server, a set of IntelliNet Gateways and client-side software – IntelliEye.

Microsoft .net 2.0
SQLServer 2005
ASP.net 2.0
Visual C# 2005

3.2.1 Management System Server and Database

The main role of the **Management System Server** (MSS) is to provide safe access to and from the database to other parts of the MS. This includes: receiving data from IntelliNet Gateways and storing them in the database, sending requests from users' software to IntelliNet through the Gateways, exchanging data with IntelliEye. The methods guaranteeing safety of all the communication are data encryption and user authentication (see 3.2.4 and 3.2.5).

The database stores all historical data. This enables users to make sophisticated data analysis and provides a complete view of the forest status. The key features which were pivotal in the choice of the database were: seamless integration with other parts of the MS, sophisticated data mining services, high performance and reliability.

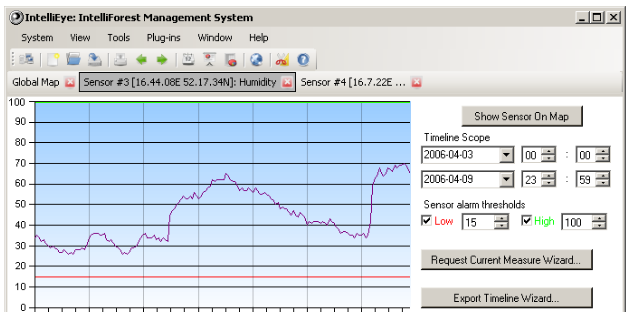
The database chosen for implementation is SQL Server 2005, which meets all the requirements. The MS uses **stored procedures** and **triggers** to provide information about the status of the forest in an event-driven way. This solves the problem with polling and delays typical of services based on pure SQL. They are implemented in C#, which ensures there are no compatibility issues between IntelliEye, MSS, Gateways. SQL Server 2005 Analysis Services provide **On-Line Analytical Processing** algorithms used for data mining.

The database used to store measurements has evolved from a sensor-centric scheme to a **place-centric scheme**. The latter scheme provides additional features, such as storing manual measurements (not done using sensors, for instance measuring tree perimeters or making site-specific notes) and keeping historical measurements available even after reconfiguration of sensor placement within the forest.

During the research at early development phases, we discovered that typical Web service architectures like XML-RPC or SOAP are not flexible enough for the MS, the main limitation being backend HTTP, which does not keep connections. **Keeping connections** is essential in corporate environments, where firewall settings and intranets prevent incoming connections, as well as in situations requiring sending notifications instantly. Aware of these bounds, we decided to develop MSS as a full-fledged functional Server instead of using stateless Web Services.

Hardware used: every x86 computer which can handle up to 100 simultaneous MSS and SQL Server connections.

3.2.2 User's Home Computer



IntelliEye is the user interface of IntelliForest. It enables users to view the current state of the forest on a map, get current and historical data on any collected property, make data analysis (from simple graphs of any collected attribute's change within selected time to advanced data analysis and decision support), export the data to external formats (see chart).

The software uses **plugins written in Python** to extend its functionality. IntelliEye provides an easy API to write plugins for custom data analysis, data presentation and data export. Pluggable architecture ensures that additional functionality can be added to the software at a later stage as needed by specific users.

Plugin type	Description	Examples
Data presentation	Provides a way to display the data collected by IntelliForest. If user has enough rights, feedback is possible.	The map, user feedback form, parameter timeline.
Data analysis	Provides a method to summarize raw data, which leads to expanding knowledge about the forest.	Static parameter analysis (average values, std. deviation, mode, max, min, etc.), fire threat analyzer, animal movement analyzer.
Data export	Provides a method to export presentation or analysis data to a well-known format.	YAML, XHTML, XML, PDF, BMP, Microsoft Office Word, Microsoft Office Excel.

The software uses the **briefcase model** (term introduced by Borland [13,14] also called disconnected computing or simply mobile computing). It is an example of a multi-tier software architecture. It enables users to benefit from collected historical data even when they are not connected to the Internet at the moment (not connected to the database). This is especially important for ecologists on the move and for foresters, who can use laptops to help them with their job. The briefcase is implemented using Sqlite, an **embedded database**. While being **online**, the user gets information directly from the MSS (which enables reacting on realtime events like alarms or animal spotting) and sends requests directly to the MSS. Meanwhile, all data from the MSS is archived on the locally available embedded database. If the user is **offline**, locally stored information are used. The files containg embedded database storage are encrypted using user credentials which protects confidential data from exposure.

Hardware used: any computer with Windows and .NET Framework 2.0. Local storage of measurements depends on free disk space.

3.2.3 IntelliNet Gateway

Microsoft .net 2.0 Compact Framework

Windows CE 5.0

The main function of **IntelliNet Gateway** is to safely forward data between IntelliNet and the MS's database. The MS is able to handle many Gateways, representing different IntelliNets. This ability can be used to protect IntelliNet against potential Gateway failure. For any IntelliNet, in particular a large one, two or more Gateways can be located and working, thus dividing the big network into smaller IntelliNets. Should any of the Gateways fail, all of its routers will automatically join other IntelliNets with working Gateways, preserving the network consistency.

This concept evolved from the idea of having an inactive backup Gateway, activated as needed. Dividing one IntelliNet into a set of smaller ones has however some advantages like better load balancing, shorter paths from sensors to the MS and thus shorter latency. Considering the applications and most typical use cases, there is no contraindication against letting all Gateways work, leaving it to the network self-reconfiguration mechanisms to maintain consistency.

The Gateways use their embedded disks to buffer data that cannot be sent to the MSS immediately (for instance because of Internet connection failure).

Hardware used: ICOPTech eBox II embedded computer with Vortex86 200 MHz CPU, 128 MB RAM, 64 MB Flash EmbeDisk HDD, etc.

3.2.4 AAA: Authentication, Authorization and Accounting

IntelliForest is by design dedicated to users such as foresters, scientists, ecologists and tourists. Since the needs and expectations of each of these groups differ, the Management System, as an IntelliForest main interface, should be appropriately adjustable. For this reason, we decided to implement the MS basing it on the concept of AAA [15].

Authentication ensures that actions in the system are not anonymous. This is done by the requirement to log into the system using a username/password pair. See 3.2.5.

Authorization makes sure that restricted actions are only accessible to users with appropriate privileges. Authorization is done by dividing users into several groups. See table 3.1.

Accounting provides a way of tracking users' activity within IntelliForest. Two scenarios are implemented: **subscription** and **pay per use**. Both are very flexible to ensure that also less typical requirements of the user can be handled.

The AAA protocols may also be present in other endpoints, i.e. the LEAF.

Rights	Public	Ecologist	Forester	Admin
Group description	Fewest rights, used primarily	Can view all stats* and optionally	Full access to the stats and	Full rights, including user

Rights	Public	Ecologist	Forester	Admin
	by tourists.	request on-demand measurements**.	IntelliNet management.	management.
View forest stats available to the public	Yes	Yes	Yes	Yes
Provide feedback by commenting on places on the map or special events	Moderated	Yes	Yes	Yes
View non-public stats	No	Yes*	Yes	Yes
Edit IntelliNet configuration (sensor placement, alarm thresholds, etc.)	No	No	Yes	Yes
Make on-demand requests	No	No**	Yes	Yes
Moderate user feedback	No	No	Yes	Yes
Manage user accounts	No	No	No	Yes

Table 3.1 User groups

* can be restricted to a limited set of stats on a per user basis
** can be changed on a per user basis

3.2.5 Data flow

The Management System is by itself a complex distributed system. Data flows between the Management System Server (MSS) and the Gateways, between MSS and IntelliEye instances. Most of the data sent between its parts are not supposed to be public. That's the reason behind strong encryption and authentication in all network communication. The encryption is based on two main components:

- **the Secure Remote Password Protocol (SRP)** [32] which is a password-authenticated key agreement protocol which allows users to authenticate themselves to a server, which is resistant to dictionary attacks mounted by an eavesdropper, and does not require a trusted third party. It uses the authentication credentials of the user.
- **the Advanced Encryption Standard (AES)** [33], also known as Rijndael, which is a block cipher adopted as an encryption standard by the US government

The data sent between MS parts are mostly data structures, such as measurements from IntelliNets and IntelliNet configuration changes. These data structures, while well defined, are complex. This implies that the communication protocol used must handle this kind of data well. At first, we considered XML. However, it turned out that its SGML roots are a major drawback: it is not easily machine parseable, in case of more complex structures it is not easily human readable, it introduces tag overhead increasing the size of data to be transferred. That is why we decided to use **YAML**, a format more appropriate for data serialization, solving all these issues. It is also easily convertible to XML, if necessary.

3.3 The LEAF

LEAF is a mobile endpoint of IntelliForest that supports users being on the move. It consists of a PDA that is equipped with an RF plugin, GPS device and a user specific application – IntelliMap. This allows collecting data directly from IntelliNet nodes and changing their settings right on the spot. The LEAF cooperates with the Management System by offering its data to the user and gathering feedback.

3.3.1 Hardware

IntelliMap requires PDA with Windows Mobile 5.0 operating system. Every device conforming to this requirement will work. User needs only to install free .NET Compact Framework 2.0 which is not preinstalled on Windows Mobile 5.0 devices.

In order to use location functionality users need to have a working **GPS device** compatible with Windows Mobile 5.0. IntelliMap uses Windows Mobile Native GPS API.

RF plugin is designed to enable user to interchange information between IntelliNet and LEAF. It is attached to PDA through a Compact Flash extension port (for a developing purposes we used Billionton CF-232 card which converts Compact Flash port to RS-232 serial port). RF plugin contains a CC1010 microcontroller which can connect to any Base Station or Sensor in proximity. Because it is a non-standard device, it requires dedicated driver developed in Microsoft Platform Builder for Windows CE 5.0.

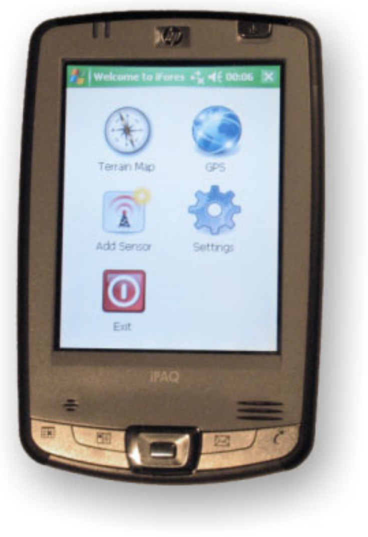


Fig. 3.13 IntelliMap menu

3.3.2 IntelliMap

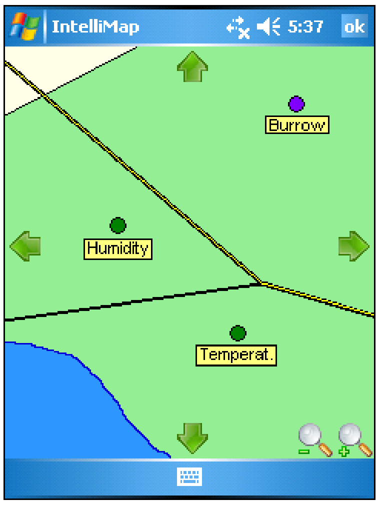


Fig. 3.14 IntelliMap digital map

IntelliMap is developed using Microsoft Visual Studio 2005 with Window Mobile 5.0 for Pocket PC SDK. It is based on .NET Compact Framework 2.0 and is mainly written in C#. However, modules crucial for efficiency, such as processing and displaying the map, are written in native code in C++, which makes the application run quickly and smoothly. The C++ code is called from managed code using COM Interop technology.

Digital map

IntelliMap implements highly interactive digital map with rich and intuitive GUI. It displays three basic, abstract sorts of objects:

- **spots** – non-dimensional objects, i.e. ant-hills, burrows, animal feeders, IntelliNet nodes,
- **lines** – one dimensional objects such as roads, trails, power lines, railroads, rivers,
- **polygons** – complex objects such as forest sectors, lakes, fields, buildings and many more.

Every object has a list of default properties, which may be browsed and modified with a single click. Moreover, the user is granted a possibility to define new categories of objects and associate them with non-default properties by editing a simple XML file. All the features stated above combined with the availability of adding and deleting objects provide users with a very flexible tool to create their own maps or edit existing ones.

IntelliMap takes an advantage of the MapInfo Interchange Format (MIF) – developed by MapInfo Company, widely used in GIS and other positioning software. Its popularity allows displaying or modifying a multitude of maps which

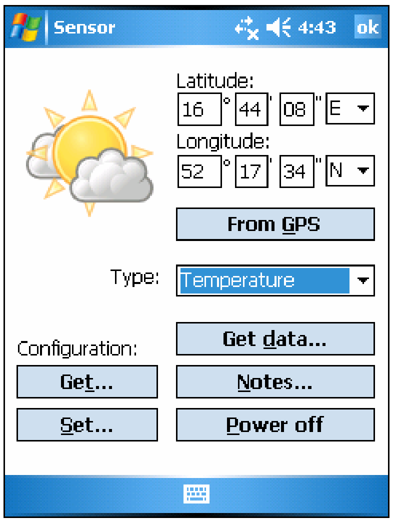


Fig. 3.15 IntelliMap sensor management

have already been created. Moreover, MIF is a very complex, yet flexible and easily extendable format [18]. IntelliMap implements displaying the basic subset of its objects, offers basic map edition functionality and extends it with forest specific objects. To get less usual options a user may purchase a third party software – for example MapInfo MapX Mobile.

User profiles

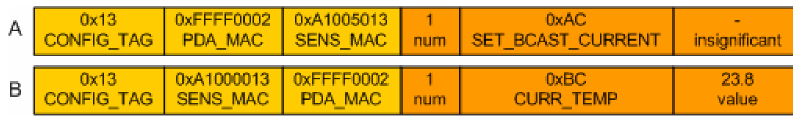
Objects and properties are user-profile specific (see table 3.1). For instance, foresters will make use of managing sensors from PDA while access to that feature is restricted for the tourists. The application was developed using certain preprocessor's defines, which allow building strictly for a dedicated user. This method strongly enhances security, as our solution simply does not distribute the code for users who should not execute it.

Integration with IntelliNet

The RF plugin gives the users possibility to communicate with sensors: collect the most up-to-date data or manage locally their configuration. On one hand, LEAF is able to observe the radio communication between nodes, displaying any packet that is in the air. On the other hand, it can construct any packet itself and unicast or broadcast it to the nodes in proximity. These two abilities, added to the normal functionality of IntelliNet, allow for:

- sniffing, controlling the status of any part of IntelliNet after a proper authentication
- updating settings of any node, which may be particularly useful at startup of a node
- making requests addressed to any node of IntelliNet and collecting responses, which includes asking for data that is being collected and stored.

In order to correctly interact with a complex network, the LEAF should have certain knowledge of it, such as exact INA addresses of distant nodes. It is automatically updated every time the LEAF synchronizes with the Management System. For simpler use cases, however, no such knowledge is necessary, which makes the LEAF operate independently from other endpoints.



A	0x13 CONFIG_TAG	0xFFFF0002 PDA_MAC	0xA1005013 SENS_MAC	1 num	0xAC SET_BCAST_CURRENT	- insignificant
B	0x13 CONFIG_TAG	0xA1000013 SENS_MAC	0xFFFF0002 PDA_MAC	1 num	0xBC CURR_TEMP	23.8 value

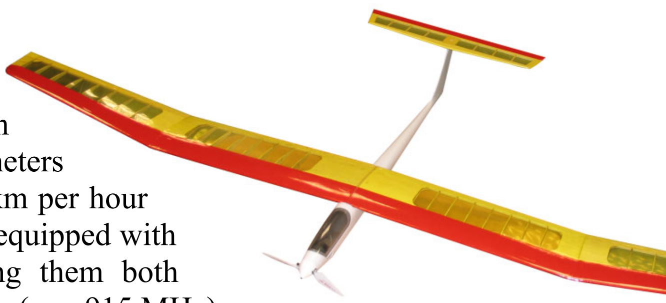
Fig. 3.16 example MESSAGE sent (A) and received (B) by the PDA

Integration with the Management System

In some use cases, in order to correctly cooperate with IntelliNet, the LEAF must periodically synchronize with the Management System, i.e. when entering or exiting the IntelliNet area. This can be done using Internet or LAN connection and may be carried out on the spot, with the use of dedicated WiFi or Bluetooth synchronization points (for example), or at home, using the IntelliEye application. In this way, with the use of the GPS module, any information added by the LEAF to the MS database is automatically enriched with the geographical location. This is useful when changing the structure of IntelliNet, for instance.

3.4 Other endpoints

3.4.1 IP camera and UAV



Unmanned Aerial Vehicle is a motorglider with an autonomous and automatic navigation system. Two meters wing-span and brushless electric engine let him fly 40 km per hour (25 mph) at 300 m (985 ft) above the ground level. It is equipped with Chipcon CC1010 and DP1205 transceiver. Combining them both provides long-range communication on 868 MHz (or 915 MHz) bandwidth for remote control. The route is set with IntelliEye by listing consecutive geographical points to fly over. Although configuration takes place before take-off, it may be changed during the flight in case of unexpected events or danger (for example fire threat).

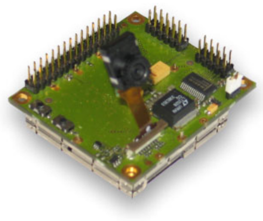


Fig. 3.17 IP camera

UAV is used for automatic monitoring of selected areas and collecting data from faraway sensors. Therefore, it is equipped with wireless camera Telit GM862. Its embedded GPRS modem allows sending video image at 900 MHz. There are no restrictions on used technology, in the future GPRS may be successfully replaced by WiMax and more advanced models of cameras may be used.

In case of a need for static video-monitoring system (used for example in the campsites), user may place additional, separate camera modules where desired. They may be often conjugated with other sensors, such as movement sensor, and be activated and deactivated by them.

UAV is an irreplaceable source of information. In case of natural disasters, such as fires or floods it helps rescue and fire brigades. If desired, the UAV will gather measurements from sensors, especially those unattached to IntelliNet.

3.4.2 Information Point

Information Point is a small static endpoint that is **addressed to the public**. It is located on the trails, at campsites or other tourist places. Similarly to LEAF equipped with IntelliMap, Information Point is capable of retrieving data from IntelliNet, either by getting them from local nodes or by sending queries to the Management System. Unlike the LEAF, however, Information Point presents only a limited set of public-oriented information and does not allow interfering with IntelliForest configuration. On the other hand, Information Points may offer additional multimedia content, such as sounds, videos, photos or pictures. In this way a user will not only know how many tagged deer were observed within 100 m the previous day (see 3.4.3), but will be also shown their photograph and detailed description.

Information Points are dedicated to public users (see table 3.1) and do not require the use of a dedicated RF plugin. They implement the Bluetooth LAN access profile which allows users with any Bluetooth-enabled device, such as mobile phone, handheld or even laptop, to connect to it using their web-browser. More advanced scenario assumes equipping an Information Point unit with a touch screen, providing even more interactivity to users.

The upload of data may be done via IntelliNet (in case of configuration or text data). Multimedia content, such as videos, must be fetched via Bluetooth or Internet (if connected) only by an authorized user.

Information Point uses ICOPTech eBox II embedded computer with Windows Mobile CE 5.0 and USB Bluetooth dongle – BlueTake BT007X.

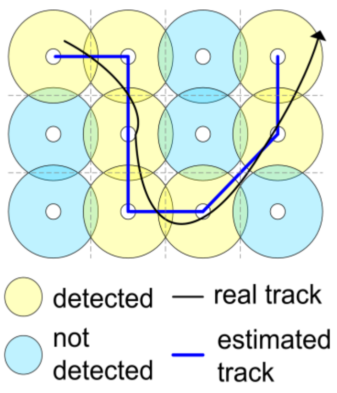


Fig. 3.18 Track estimation

3.4.3 Mobile motes

Mobile motes are small wireless transceivers detectable by dedicated sensors of IntelliNet. They are carried by animals so they must be small, lightweight, and have long lifetime. Mobile motes broadcast short identity messages, while it is up to the dedicated sensors to receive them and make them available for other endpoints. For instance, such information may be used online by the LEAF for the purpose of animal tracking, while the Management System may offer more advanced analysis.

Each animal detection may trigger an alert. For this purpose sensors implement so called black lists. When an identity message is received, sensor searches the black list for it, and if it finds a match it instantly sends its results to the Management System (if there is one). The identity code of each mobile mote consists of four characters, which allows for assigning precise codes to the animals depending on their species, sex or age. Users may define a complex system of identity codes, thanks to which they may easily add to black lists queries for certain species, all males or females, etc. The mobile mote identity message is programmed with a use of LEAF, which enables *in-situ* configuration.

Mobile motes and corresponding sensors may in the future be implemented with the use of RFID technology, as for example the TI-RFid long-range readers and Tag-it transponders from Texas Instruments. Our prototype operates CC1010 transceivers. For example, when T_P = 10s the broadcast times of motes and listen periods of sensors comply with a timing schedule that ensures detection within **25 seconds** while being in *sleep mode* for at least 90% of time. This combines relatively fast detection with a long lifetime of **9 months**[1]. In case of different schedules of nodes' activity, the lifetimes or detection delays change respectively.

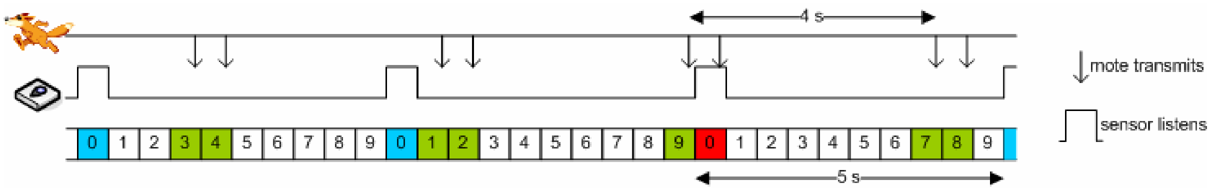


Fig. 3.19 Mote detection schedule

3.5 Costs

IntelliForest is a highly extensible and modular system. Its overall cost depends strictly on the types and amounts of elements it consists of. In general, the cost of sensor is around 15$ with the cheapest meters (i.e. thermometer); for more expensive meters (i.e. oxygen meters) the price rises respectively. The cost of other components is presented in the table below. PDA and web servers are not listed since they can be any devices commonly used by the users.

The overall cost of the system has been calculated for each case study (see 3.7) but even knowing the exact application the overall cost often differs depending on the scale of research or other factors. For instance, the medium-scale IntelliForest used for microclimate research costs app. 12760$, while small-scale system used for controlling the animal traps would cost app. 850$. The detailed prices and calculations are given in separate documents.

3.6 Verification and testing

System Component	Cost
Sensor – depending on type	> $15
Router	$25
RF Plugin	$30
Gateway (eBox II)	$230
Information Point	$40
Active RFID tag	$10
Active RFID Reader	$200
GPRS Video Camera	$400
UAV – depending on model	> $3000

The Feature-Driven Development (FDD) methodology introduces the idea of **features**, which are basic increments creating the whole implementation. The testing process was done simultaneously with development – every new feature was verified, very often using spike solutions (e.g. simple packet spammer to test packet handling). Moreover, we often used unit tests to ensure the correctness of changes. This quickened the error detection and allowed to correct them at an early stage. Therefore, the overall cost of the software validation was relatively low.

With successive completion of the so-called **sets of features**, more complex scenarios were also tested, involving a few implemented features. An example of this is prepar-

[1] Mobile motes powered with one AA-sized LS14500, dedicated sensors powered with one D-sized LS33600

ing the routing updates and delivering them correctly to appropriate node.

It was essential for the development of IntelliNet to determine the quality of RF communication in real-life conditions. To determine the limits of proposed solutions, we performed signal power tests in the Poznan Greenhouse, with its humidity over 85%, temperature of about 31 degrees Celsius (88°F) and dense flora. In such conditions, comparable only to those encountered in the rainforests, we estimated the range of the low-power RF signal to be approximately of 50 meters (165 ft). For the purposes of this test we developed an RF signal power meter on Chipcon.

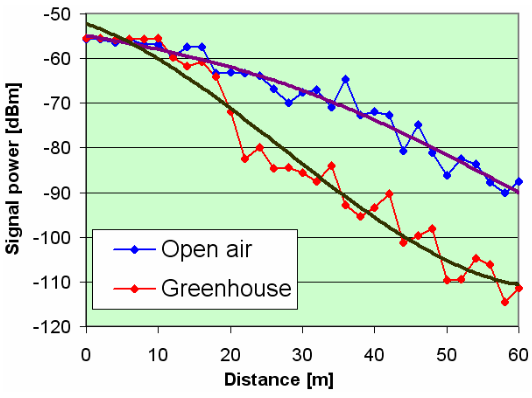
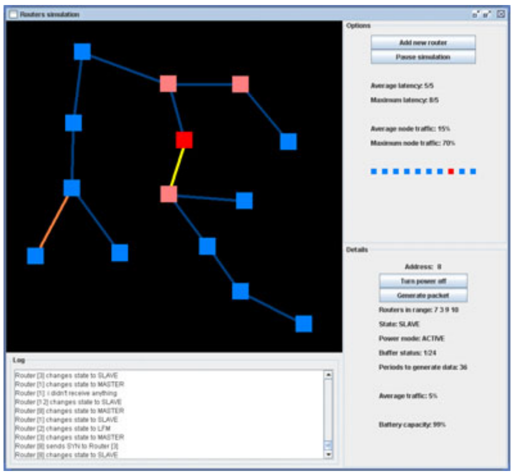


Fig. 3.20 Signal absorption [868 MHz, +4 dBm]

The verification of the communication algorithms was done using the CC1010 Evaluation Boards and later, with the devices developed by us. We connected the nodes to PC via RS-232 port and with the use of HyperTerminal verified that the all paths are followed properly. We left a small network for 24 hours and checked that measurements are generated and all packets delivered to the Management System.



Average latency: 5/5
Maximum latency: 8/5
Average node traffic: 15%
Maximum node traffic: 70%
Address: 8
Routers in range: 7 3 9 10
State: SLAVE
Power mode: ACTIVE
Buffer status: 1/24
Periods to generate data: 36
Average traffic: 5%
Battery capacity: 99%

```
Router [3] changes state to SLAVE
Router [1] changes state to MASTER
Router [1]: i didn't receive anything
Router [1 2] changes state to SLAVE
Router [8] changes state to MASTER
Router [1] changes state to SLAVE
Router [2] changes state to LFM
Router [3] changes state to MASTER
Router [8] sends SYN to Router [3]
Router [8] changes state to SLAVE
```

Fig. 3.21 Routers' simulation

The overall performance of IntelliNet is determined by system's parameters: T_A, T_P and T_{SYN}. T_A time is a constant factor, as it depends on nodes' work characteristics. T_P was chosen as a tradeoff between the latency and lifetime. Finally, having T_A and T_P determined, T_{SYN} was calculated and proven to be optimal (see 3.1.4).

Having six CC1010 Evaluation Modules together with dedicated boards, we conducted tests of small two-layer nets. This confirmed that the layers cooperate correctly. Additionally, in order to build a large scale model for testing purposes, we developed a simple simulation of the upper layer of IntelliNet using Java and Eclipse. This allowed us to observe the transmission of data in the network and to successfully verify the correctness of general routines, like changing network structure, and of characteristics, such as maximum latency depending on the distance of source and traffic congestion in network's bottlenecks. The tests were performed for networks differing in structure (broad or deep tree), longest path length and overall number of nodes.

The Management System was heavily tested against any abuse or possible failure. The relational logic tests (conformance with the Extended Entity-Relationship model) and random data population (performance and stability) were performed to validate the SQL database. The Management System was heavily tested against any abuse or possible failure. The relational logic tests (conformance with the Extended Entity-Relationship model) and random data population (performance and stability) were performed to validate the SQL database. Moreover, Management System Server security was tested against SQL injection [19], XSS, buffer overflows, cookie spoofing [20], Denial of Service and broken requests [21]. Finally, the IntelliEye's plugin mechanism was verified against malicious code and privilege escalation [20].

The heart of the Management System is the Management System Server (MSS) + SQL Server duo, the intermediate interface for stored data access and data analysis. Thus, the hardware and software used to host these applications must be fast enough and reliable even in a multithread, highly concurrent environment. We tested MSS and SQL Server on a P4 2.4 GHz 1 GB RAM system with MS Windows Server 2003 Std, MS SQL Server 2005 Std and IIS 6.0 with .NET 2.0 installed.

IntelliMap and the GPS module was tested using Poznan city map in MIF format on HP iPAQ hx2490 handheld. Having IntelliNet finished and tested we conducted integration tests. The attempts to gather data from two neighboring temperature sensors and to change their settings were fully successful.

3.7 Case studies

Concurrently with the development process we conducted detailed research on market opportunities for IntelliForest. As a result, we obtained a list of ten case studies. Each of them has been suggested by a specialist and minutely consulted with him. A few of these consultations have already ended with an invitation for further cooperation, as a part of a research project for example.

- Dr Paweł Rutkowski from Poznan Agricultural University is using data-logging sensors to examine the **microclimate** changes. This helps him to investigate the influence of water flow in environment on plants compounds and – indirectly – on forest types. Mr. Rutkowski reckons IntelliForest, automatically collecting and sharing the data, a much bigger support to his research.
- The region of Poznań is a place of natural existence border for the **beech trees** (*Fagus Sylvatica*). Beech forests renew themselves without human attention, but on different areas with different intensity. This phenomenon has been observed by foresters from the Forest Inspectorate Lopuchowko and remains unexplained. Thus, it is desired to conduct long-term (several years) environment monitoring in order to find the correlations and dependencies. IntelliForest's sensor infrastructure and Managament System's abilities of data analysis perfectly fit those demands.
- **Hermit beetle** (*Osmoderma Eremite*) is a threatened species strongly bounded with its habitat. It lives in a radius of approx. 200-400 meters (650-1300 ft) around the rotten old trees and logs, where it nests and in case of some threat is unable to find new habitat and is doomed to die. As a part of a European Union project "Nature 2000" the insect will be bred and later dispersed over the forest. It is crucial to constantly monitor the colonies to discover dependencies between environment conditions and hermit beetle life cycle. The breeding is scheduled to take place in a village Bald Mill. The entire process will take probably 4 years to complete.
- Biedrusko is one of ten Polish **military training fields**. Every morning, before starting the shooting, officers need to know precisely if the forest's bed humidity is high enough. A net of dispersed humidity sensors that may be contacted from the office or on the spot is a solution that perfectly addresses this issue.
- Every year thousands of tourists visit the Snowy Raven looking for the **Spring Snowflakes** (*Leucojum vernum*) plants. This is a threatened species which seeds are distributed by ants. It is a matter of great importance to provide a proper ecological education for single tourists, families, children, even entire classes. The chief forester, Zbigniew Szeląg, reckons IntelliForest's Information Points as a suitable solution.
- Przemysław Rumianowski is a person responsible for compost production in Forest Inspectorate Lopuchowko. To improve the production's efficiency he observes and analyzes the process of **mineralization of organic substances**. For his research he needs a set of wireless sensors, which will supersede the currently used manual meters.

- Workers of the **National Park of Karkonosze** perform biological monitoring connected with wildlife protection. The purpose of it is to guarantee that the environment conditions meet the requirements for certain newly (re)introduced species, such as butterflies. Most of the measurements are done by standalone sensors which store the data in their flash memory. The maintenance of such sensors is very time consuming. It is desired to introduce more automation.
- Piotr Śniady from the **Poznań Greenhouse** is interested in using the IntelliForest for monitoring the conditions inside and outside the glasshouse. Apart from constant control of features such as temperature, humidity and insolation, IntelliNet would be connected to already existing devices, such as mist makers.
- Dr Grzegorz Górecki from Poznan Agricultural University conducts **biometric research of game**. As part of his research, with the cooperation of the Game Investigatory Centre of Zielonka, he uses dedicated animal traps set up in natural habitats. Using IntelliForest as a means of remotely controlling the status of the traps is highly desired, since it would let the scientists avoid making unnecessary trips and increase the efficiency of capturing the animals.
- Fire threat forecast and **fire prevention** is one of the responsibilities of Forest Inspectorates. From the beginning of April to the end of September forest bed's humidity is measured and monitoring from air is often performed. According to dr Andrzej Łabędzki from Poznan Agricultural University the implementation of IntelliForest along with UAVs would highly improve the fire detection processes.

3.8 Teamwork

Feature-Driven Development, which we adopted as the programming methodology, had some impact on the way we organized and distributed work in our team. We assigned every component of IntelliForest to one of the team members, so as to ensure strong class ownership. We performed code reviews after every major change in the code, particularly in IntelliNet, in which Pawel and Piotr were working on different modules simultaneously. In any component, once a feature was implemented, it was presented to other team members during a meeting in the office. This approach allowed each of us to organize work flexibly, yet ensured that **progress was visible and under control** – which is one of the objectives of FDD.

4 Summary

It took us two months to complete the first three steps of the Feature-Driven Development schedule. As a result, by numerous consultations and discussions, in January 2006 we achieved a well formed model of IntelliForest. The following months were devoted to detailed design and implementation. At the current stage, we can state that IntelliForest, in its fundamental functionality, has been developed and tested.

- The core of the system, IntelliNet, is functioning according to our specification. The communication algorithms have been implemented and proven to work correctly.
- The Management System has its database and Web services implemented and integrated with the IntelliEye application.
- The LEAF is fully operational, including using the GPS with digital maps and cooperating with IntelliNet.
- The Gateway has been implemented on eBox II and is fully integrated with IntelliNet and Management System.
- Information Point prototypes are working according to the Bluetooth-usage scenario.
- Mobile motes are implemented and cooperate correctly with IntelliNet.
- We started working on the UAV.

At the same time, we designed and built prototype devices for IntelliNet nodes. They allowed us to set up a complete, small-scale IntelliForest, and to implement and test various usage scenarios.

Simultaneously to developing the hardware and software of IntelliForest, we performed consultations with potential end users. We presented the design and prototype to scientists of such branches of science as game investigation, silviculture, dendrometry, entomology, to workers from Poznan Greenhouse and to foresters from the Zielonka Forest and Karkonosze National Park. Deep interest and support from them helped us in developing the project but also showed us promising and challenging perspectives for the future. We have already received three invitations to participate in **projects financed by the European Union**, and two offers of **implementing the system commercially**. At the time of creating this document, these talks are mostly in the preliminary phase, yet the objectives and possibilities of all sides make us believe that IntelliForest will soon be set up and working.

And it will be in this very moment that IntelliForest will truly support people in **preserving, protecting and enhancing the environment**, for the benefit of us all.

5 References

[1] http://www.fao.org, Food and Agriculture Organization of The United Nations

[2] Scott W. Ambler, "Feature Driven Development (FDD) and Agile Modeling", Essay, http://www.agilemodeling.com/essays/fdd.htm

[3] http://www.featuredrivendevelopment.com/, Feature Driven Development Portal

[4] "Single Chip Very Low Power RF Transceiver with 8051-Compatible microcontroller", Chipcon, December 2004

[5] "DP1205 – C868/C915 MHz RF modules", Semtech, August 2005

[6] Wei Ye, John Heidemann, Deborah Estrin, "An Energy-Efficient MAC protocol for Wireless Sensor Networks", In Proceedings of the IEEE Infocom, New York, June 2002

[7] A. S. Tannenbaum, „Computer Networks", Pearson Education Inc., 2003

[8] P. M. Evjen, "AN001 SRD regulations for license free transceiver operation", Chipcon, 2003

[9] "SmartRF CC1010 IDE User Manual", Chipcon, November 2002

[10] A. El-Hoiydi, J.-D. Decotignie, J. Hernandez, "Low Power MAC Protocols for Infrastructure Wireless Sensor Networks", CSEM Technical Report, Swiss Center for Electronics and Microtechnology, November 2003

[11] S. Vetti, "AN019 Crystal oscillator issues for CC1000 and CC1010", Chipcon

[12] Elizabeth M. Royer, Chai-Keong Toh, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks", 1999

[13] Eric Harmon "Delphi/Kylix Database Development", SAMS, 2001

[14] http://www.crystal-soft.com/globalapps/Briefcase/briefcase.htm, „Briefcase Model", Article by Crystal Soft, 2002

[15] RFC 2975, "Introduction to Accounting Management", 2000

[16] Allen Jones, "C# Programmers' Cookbook", Microsoft Press, 2003

[17] "LS33600 3.6 V Primary lithium – thionyl chloride High Drain D-size cell", Saft

[18] http://resource.mapinfo.com/static/files/document/1074660800077/interchange_file.pdf, MapInfo Data Interchange Format

[19] Chris Anley, „Advanced SQL Injection in SQL Server Applications", NISR, 2002

[20] Michael Howard, „19 Deadly Sins of Software Security", McGraw-Hill Osborne Media, 2005

[21] Michal Zalewski, „Silence on the Wire", No Starch Press, 2005

[22] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, David Culler, "An analysis of a large scale habitat monitoring application", Proceedings of the 2nd international conference on Embedded networked sensor systems, Baltimore, 2004

[23] Jason Lester Hill, Ph.D. Thesis "System Architecture for Wireless Sensor Networks", University of California, Berkeley, Spring 2003

[24] Vibhore Goyal, Dual Degree Dissertation on "Design of Development Kit for Wireless Sensor Module" under the guidance of Prof. U. B. Desai, June 2005

[25] "SmartRF® CC1010DK DK User Manual", Revision 2.01, Chipcon, February 2003

[26] K. H. Torvmark, "AN017 Low Power Systems Using the CC1010" with attachments, Chipcon

[27] S. Hellan, "AN018 CC1010 Debugging Hints and Troubleshooting", Chipcon, 2003

[28] http://msdn1.microsoft.com/en-us/default.aspx, MSDN Library

[29] Edited by Theodore A. Bookhout, "Research and management techniques for wildlife and habitats", The Wildlife Society, Bethesda, Maryland 1994

[30] Simon Singh, "The code book. The science of secrecy from ancient Egypt to quantum cryptography", Świat Książki, Warszawa 2003

[31] Jadwiga Małachowska, Jerzy Wawrzoniak "Ocena uszkodzenia lasu na Stałych Powierzchniach Obserwacyjnych Monitoringu Biologicznego w 1993 r" (Forest damage estimation on fixed areas of biological monitoring in 1993), Warszawa 1994

[32] http://srp.stanford.edu/

[33] http://csrc.nist.gov/CryptoToolkit/aes/rijndael/