Eco Net

Technical Note

Version 0.1
December 2007

TOC

Introduction

This technical note introduces Olsonet's *'Eco Net'* blueprint, shown from several different angles, with varying detail level.

The *Vocabulary* section outlines the terminology used.

The *Demonstration* section studies a functional demonstration. As a side effect, we present *Virtual Underlay Emulation Engine (VUEE)* in action, touching on its tremendous potential as a development and maintenance tool.

The *Deeper Look* section discusses main differences between the demo version and any industrial *Eco Net* incarnation. In this section we also point at a hidden and deceiving gap between the lightness and clarity of the visible functionality, and challenges intrinsic to any robust and scalable deployment of ad-hoc (sensor) networking. This gap is not advertised by vendors offering functional fragments, e.g. 'standard' protocol stacks and, in our view, is one of the key reasons for scarce industrial acceptance of ad-hoc networking. Hardware variants are discussed as well, hinting on the ways praxes based on the *Eco Net* blueprint morph into practical application / solution.

The *Eco Networks* section is a high level view at the blueprint itself.

Vocabulary

Eco Net blueprint welds together two other Olsonet's blueprints, *Tags and Pegs* and *Routing Tags*. We named the blueprint *Eco Net*, because it was conceived and designed when working with our illustrious friends from http://eosl.eas.ualberta.ca/, and its first practical deployment helps with their ecological research. In this document, the nomenclature is somewhat inconsistent, as we use synonymous terms, picking the one that better fits any given functional piece.

Peg: An intentionally stationary or low mobility node, forwarding packets from other *Pegs*, and gathering information about *Tags* in the vicinity.

If position tracking is part of the functionality, *pegs* form a grid of reference points with known locations.

If data aggregation takes place, *Pegs* become *aggregators*. Aggregators have local static storage or pass data through OSSI, or both.

Aggregator: a *Peg* that aggregates payloads from messages received from *collectors* (*Tags*).

Tag: A mobile or stationary node periodically transmitting information about its physical environment. It is capable of receiving packets, but in sporadic and restricted ways. Usually, *tags* don't route packets (although they may).

Collector: A *Tag* with attached instruments (sensors, actuators). It collects data from its environment, and passes them to aggregators. Optionally, collectors route traffic from their peers to aggregators.

Collectors may be equipped with static memory, to store collected data at the source (e.g., for reliability, auditing).

Master: a *Peg* node identified as the *sink, tributary, gateway*, etc., where most of the packets originate or terminate. Used in routing optimization and network synchronization.

Network Id: A logical network identifier. Several (logically separate) networks with different network identifiers may operate in the same physical neighborhood.

Local Host: A (settable) logical node identifier, unique within a given network id.

Host Id: *Hardware Id, Equipment Serial Number*, etc. A constant and unique identifier flashed into the node or obtained from a dedicated hardware resource.

Master Id: The logical node identifier of a *master*.

OSS(I): *Operations Support System (Interface).*

RSS(I): *Received Signal Strength (Indicator)*

Demonstration

Instead of a flashy façade, we focus on the essential elements of functionality. The screen snapshots have been taken from the demo praxis running under VUEE, but the functionality is similar and the code identical on the network of real nodes built around MSP430 and CC1100. For brevity, we highlight only main functions. Complete demonstration should include this presentation within VUEE, followed by a demonstration of physical nodes with attached sensors, running the very same software.

As in many other practical applications, in many praxes derivable from *Eco Net*, networking plays an important but subdued role. Quality and calibration of sensors, and their PicOS drivers are pivotal in *Eco Net*. Storage design is important, too. We do expect networking to play increasingly significant role in larger deployments or follow up installations (see the *Eco Networks* section), but for now we focus on data collection and presentation. Thus, the main goal of the discussed experiment is to verify the sensors' parameters, their connectivity, and their device drivers within PicOS. Networking, which has been already verified in other applications, plays a secondary role here, hinting at possibilities of functional add-ons and illustrating some inevitable trade-offs, as the one between the power budget and the flexibility and intensity of deployment.

Equipment

Olsonet's general experimental board EMSPCC11 has been adopted for the collector (*Tag*) node by incorporating the necessary extra hardware, i.e., sensors. A similar board (devoid of the sensing components) has been used as the aggregator (*Peg*). Two collector nodes have been built, each looking like this:

Sensors:
PAR: <http://www.apogeeinstruments.com/pdf_files/QS.pdf> (the QSO-S variant)
Temperature and relative humidity:
<http://www.sensirion.com/en/pdf/product_information/Data_Sheet_humidity_sensor_SHT1x_SHT7x_E.pdf> (SHT75)

The essential components of Olsonet's EMSPCC11:
MSP430F1611 microcontroller: <http://focus.ti.com/docs/prod/folders/print/msp430f1611.html>
CC1100 RF module: <http://focus.ti.com/docs/prod/folders/print/cc1100.html>
On board data storage: <http://www.atmel.com/dyn/resources/prod_documents/doc3443.pdf>

Sensor connection, verification, calibration, and reconciliation with already used and trusted devices takes a lot of effort, and sometimes personal commitment and bravery, as recorded on the roof of the Tory Building on the University of Alberta's campus:

Networking

The general setup may involve multiple aggregators. An aggregator listens to messages periodically broadcast by the collectors placed in the neighborhood. It maintains an inventory of all collectors in the vicinity and reports local events (sensor readings, collector malfunction or absence) to the master. It also stores all sensor readings in its static memory.

The master is typically one of the aggregators that happens to be connected to an OSS; an entity controlling the network. As an aggregator, the master also handles collectors in its vicinity; however, it does not aggregate information from other aggregators, as this role is reserved for the OSS.

A collector periodically broadcasts information acquired via attached sensors. The demonstration focuses on the collection and communication of sensed data and ignores some other aspects of functionality present in the blueprint. In particular, the praxis is equipped with a location discovery mechanism based on correlation of multiple RSSI readings (by multiple *Pegs*) corresponding to *Tag* transmissions at different power levels. Although this functionality is retained in the demo praxis, it is not used by the OSSI. (See the "*TNP demo*" technical note for *Tags & Pegs* functionality.)

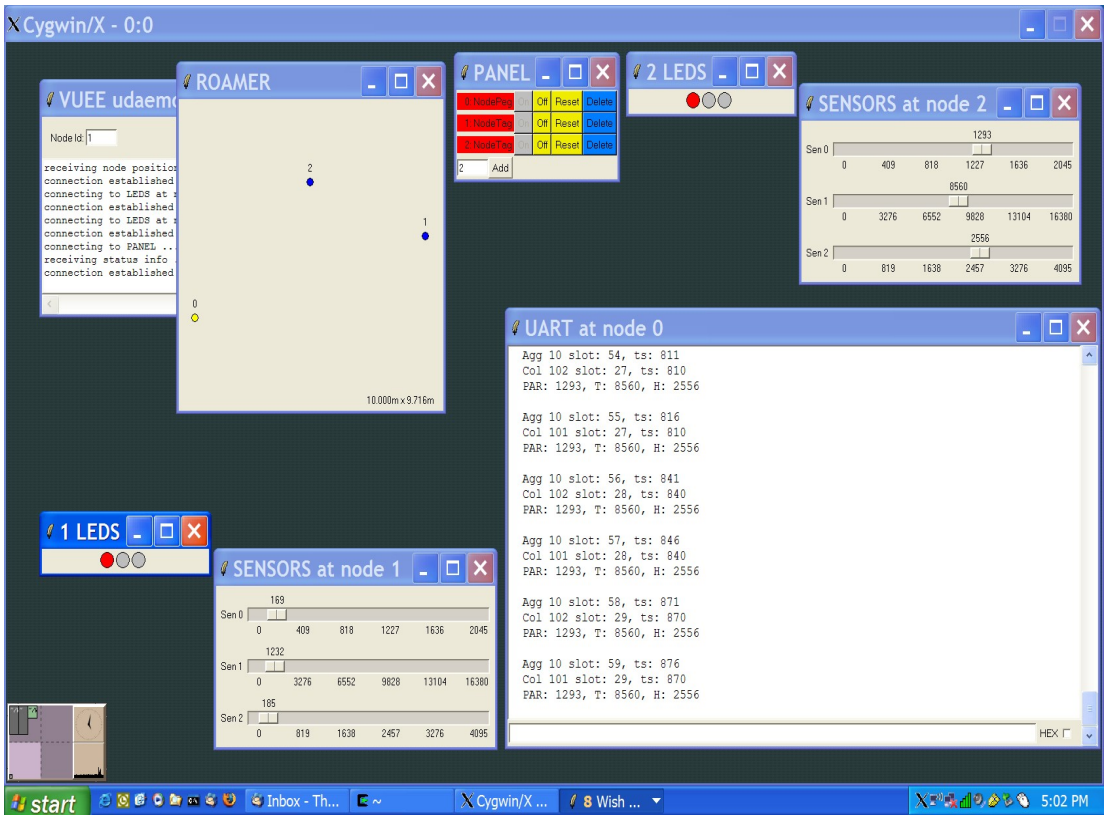
A collector reads its sensors at a (dynamically) configurable rate with the default of one reading per 60 seconds. This rate also determines the basic *duty cycle* of the collector implementing power-efficient operation. Having broadcast a message with sensor readings, the collector briefly opens its receiver in expectation of an acknowledgment or other (control) messages addressed to it. The message is retransmitted up to 4 times, or until the node receives an acknowledgment from any of the aggregators in its range. The sensor reading is stored in local static memory and marked as '*confirmed'* or '*collected'*, depending on whether it has been acknowledged by an aggregator.

An aggregator receives messages from collectors, acknowledges them, writes the sensor readings to its static memory, and either passes them on via OSSI (if it happens to be the master), or forwards them to the master.

The master interfaces to the OSS (via RS232), and communicates relevant events from all aggregators. Also, it accepts network configuration and maintenance commands from the OSS, as well as queries about aggregators, collectors, and sensors.

A simple VUEE model

In the following snapshot of VUEE emulation of the network, the OSSI is represented as an ASCII terminal. The interface has two modes: the ASCII mode is used for direct communication with the master (useful for development, demonstration, and experiments), while the binary mode implements a reliable and more efficient protocol suitable for communication with a program.



The network configuration for VUEE is described as an XML file (*eco_demo.xml*). It consists of one aggregator (id 10, node 0 in the Roamer window), and two collectors (id 101, 102, nodes 1 and 2 in the window). The *Panel* shows that all nodes are ON (i.e., powered up). The UART window (attached to node 0) is showing the near real time output from the aggregator that combines sensor readings from collectors 101, 102. This setup emulates the Tory Building experiment at UofA.

A few comments

As in most data logging applications, sensor readings are stored at the source. In this case, they are also stored at the aggregator, for easy (single-point) retrievals. The postulated functionality of the demo network assumes a single aggregator per a population of collectors (50-200) operating as an autonomous setup. Normally, such an aggregator is interfaced to the OSS only intermittently, e.g., when a human acquisitor decides to pull the aggregate data corresponding to a certain reasonably large period (say 1 month). Data storage at collectors was not required, but turned out to be natural owing to the availability of storage on EMSPCC11. Note that the existing blueprint provides considerably more functionality than that requested for the demo. In particular, it can deal with a dynamic population of multiple aggregators and masters possibly interfacing multiple setups to higher-level stations. A sophisticated OSS system associated with a master can easily implement high-bandwidth connectivity to the outside world, e.g., via satellite links.

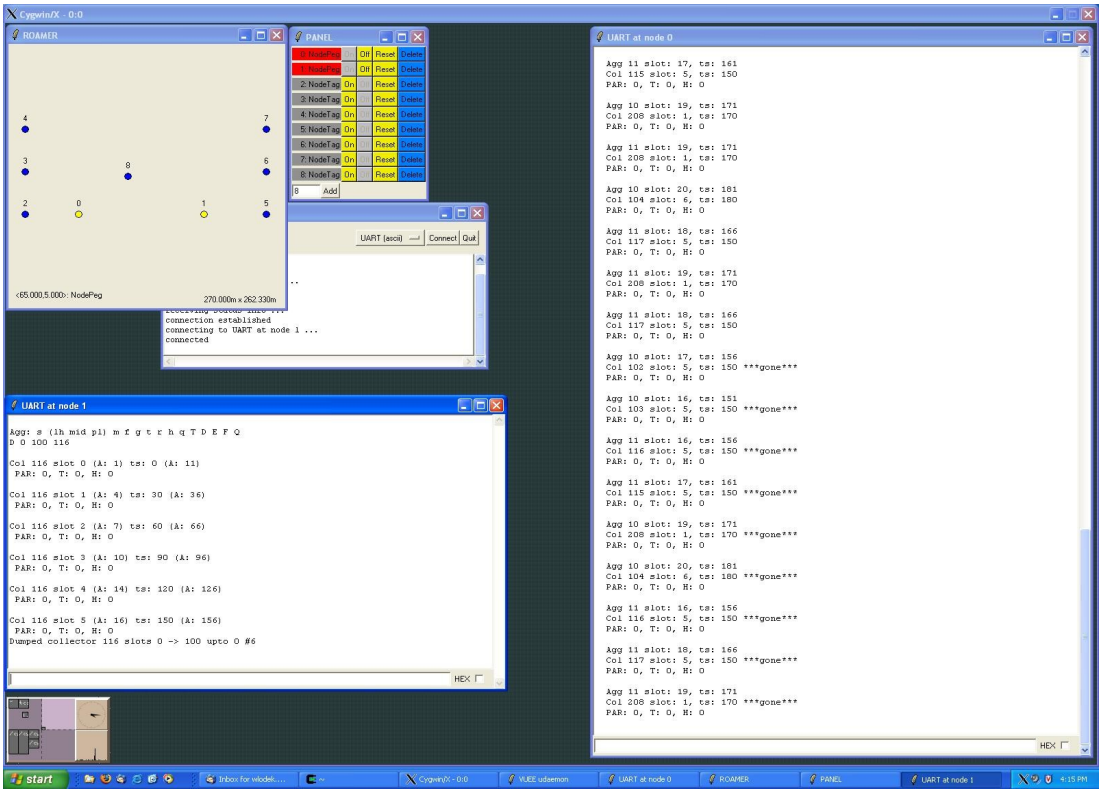
Collected data sample are time-stamped. The time stamps are relative to the master's time, which can be set from the OSS, to provide a reasonably reliable real time clock for data correlation. The “slot numbers” associated with the data samples (appearing in the master's output) directly refer to EEPROM locations. They provide a data index of sorts, which may be useful for tests and debugging. This index is generally useless at the OSS level, where the time stamps should play the role of data tags.

In this demonstration, the LEDs present on the collector nodes show the collection status, communication, and confirmation. This is an option that may be desirable in some circumstances and unwanted in true deployments (especially when efficient power utilization is of concern). It is estimated that a collector will be able to operate continually for 2-3 years on a single battery (around 1Ah capacity).

Sensor emulation in VUEE has nothing to do with real instruments; however, the VUEE implementation of the praxis is critical not only for development, but also for later deployments, OSS integration, communication links verification, etc. In general, VUEE should be as useful for the users, as it is for developers. Note that the the model is driven by exactly the same source code as the real-life network.

A more complex model

Here is a snapshot from a more involved model (file *large_eco.xml*):



The network consists of two plots:

- Agg 10 with collectors 102, 103, 104, 208 (nodes 0, 2, 3, 4, 8 in the Roamer window).
- Agg 11 with collectors 115, 116, 117, 208 (nodes 1, 5, 6, 7, 8).

The aggregators are 140 m apart, within the RF range, and aggregator 10 is designated as the master (with OSSI, presumably with large storage, satellite links, etc.). The snapshot is taken after all collectors have been stopped and reported '*gone*' on the master. Also, a query about collector 116 has been issued to aggregator 11 (see 'UART at Node 1').

Main points:

- A single OSSI setup can naturally serve more than one plot.
- Data retrieval options cover: immediate (near real time) transfer to OSS, on-demand retrieval from aggregators, direct (manual/emergency) retrieval from collectors. Note that the latter are capable of collecting data independently of the network. In particular, a stand-alone collector is a functional device.
- Additional redundancy is present when a collector is visible from multiple aggregators. Note that this may be a pro (intentional backup of critical data) or a con (wasted resources on overlapping aggregators).

Note that the generic functionality of the Tags and Pegs blueprint brings in many options that have not been requisitioned for the demo, but, depending on the circumstances, may prove useful. The inherent flexibility of forwarding, available in all Olsonet solutions, makes it possible to retrieve data from collectors located out of range of their aggregators. Logically separate sets of collectors and aggregators can coexist in the same area and even assist themselves in the task of data forwarding. The networks can be reconfigured dynamically with different aggregators assuming the role of masters and collectors being reassigned to different aggregators, including ones not directly reachable from them. On top of that, the network can take advantage of specialized equipment available at some nodes (e.g., GPS receivers) to paint its topology and convey it to the OSS.

A Deeper Look

We hint at several details that should be considered during the design and implementation of a real application (within our framework, *application* is synonymous with *solution*). Although, with our holistic methodology, we do not formally recognize layers, it is convenient to classify matters as pertinent to applications, praxes, routing, OSSI, etc.

Application

The collector's hardware will likely be different from the *Peg's*. Depending on the interfaces to the physical world, the praxis functionality may be quite different as well. The number of LEDs, the presence of an LCD, the number of sensors, and the presence of sensors directly supporting the requisite functionality (e.g. accelerometers for location engines), have profound impact on the way the praxis is derived from the blueprint. Static memory should be engineered to store needed information,

but also to stay within the power budget. Enclosures and sensor mounting depend heavily on intended deployments.

It may be that the hardware of the *Pegs* designated to act as masters (have OSSI capabilities) should be different from other *Pegs* (aggregators). If the OSSI is simple and the needed hardware ubiquitous, as in the RS232 case, this distinction is not worth too much attention. However, if the OSSI consists of sophisticated and expensive equipment like satellite modems, Bluetooth, or even Ethernet, it makes sense to divide *Tags* into subsets with different functionality. If the nodes with OSSI capabilities are not expected to act as aggregators, it may be beneficial to borrow from other blueprints, and introduce a third application (say the *Custodian*). The logistics for maintenance, distribution, customer support, etc. must be considered before making this decision, as it would be difficult and expensive to change it after the solution has been designed.

In short, physical deployment constraints, data collection intensity, and access to the nodes will define main differences between the demonstration and the application.

Praxis

The power budget considerations for *Tag* nodes have been carefully designed in. Those of *Pegs* are believed to be more relaxed. If network-wide duty cycles are needed, they can be incorporated easily. However, they may preclude useful functionality, e.g. instant alarm delivery. Since it doesn't make sense to pollute the blueprint with a functionality that is irrelevant in many applications, *Pegs* don't have this capability by default.

Similar reasoning is applied to message encryption and user authentication. Note that encryption / authentication is trivial to add, while network-wide duty cycles are not.

The demonstration praxis lacks several important diagnostic and troubleshooting functions, e.g. route trace, and remote command execution. They're always needed in real deployments.

OSSI

Text based interface is impractical, for several reasons. Compact commands in binary formats are more efficient. For demonstrations without a GUI, text commands are maintained, but the *Eco Net* blueprint assumes binary OSSI, with a clearly defined command set.

Data integrity should be verified on the OSS side, since it has more resources to use for this non-core but expensive and important functionality. Embedded software should shield itself only from disastrous input.

If the OSSI can be reliable (i.e. checksums, flow control, etc.) it should be made so. Usually this is easy for new systems, and close to impossible if the praxis interfaces to an established infrastructure.

Routing

For typical deployments, routing is insignificant. *TARP* parameters (route recovery, slack, cache sizes) should be carefully adjusted only for large deployments. VUEE should play a vital role in the studies to verifying resources used for caching, as well as the system stress testing, vicious scenarios, etc.

Some novelties (postulated for TARP, but so far deemed unnecessary), like fuzzy acknowledgments, parallel route elimination, or RSSI thresholds for cache updates may be considered, but because of the assumed liberty in engineering the network of *Pegs* and their immobility, the routing should be robust with default *TARP* functionality.

Network engineering

For typical deployments, there is not much to worry about; only density, distances, and plot size should be specified. Site surveys and RF channel modeling is crucial for a successful deployment only if extensive routing is involved. For complex setups, deployment rules should set desirable distance ranges and power levels. Note that this is not about creating a rigid infrastructure, but rather about rules that make the network flexible and expandable without unwanted side effects. This may be a complex task, usually beginning with an RF survey of the deployment site, followed by the RF channel model, followed by VUEE runs.

Eco Networks: the blueprint

Introduction

Eco Net is a highly customizable ad-hoc sensor networking solution that answers to these main requirements:

R1. Sensor operations are integral part of the solution. The following is included:

- ✓ verified connectivity and operations of all involved sensors
- ✓ hardware integration
- ✓ data retrieval
- ✓ OSS interface (data formatting)

R2. Reliable data delivery, optimized data retention and redundancy.

R3. Power management optimized for intended deployments.

R4. No in-field configuration for intended deployments, minimal configuration for exceptional setups.

R5. Easy routine operations and maintenance (OAM).

Main functionality

Eco Net collects environmental data through installed sensors, aggregates the collected data in dedicated nodes, and offers the aggregated data through an OSSI.

Supported sensors

Any reasonable equipment can be accommodated. See the *Demonstration* section for already implemented drivers. The addition of a new sensor usually takes hours; however, calibration may be required in some cases.

Node types

Note that we outline functional types. The actual nodes can have vastly different resources, or they can be identical, depending on the application.

Collector (Tag, Sensor node)

Reads attached sensors in configured time intervals, formats the data, and reliably forwards them to aggregators. Optionally, collectors retain formatted data for redundancy and disaster recovery scenarios.

Routing Collector (Routing Tag, routing sensor node)

As the collector, with routing capability. Hardware and software are identical as on the collector, but the routing duties implicate bigger power requirements. Also, the network configuration and deployment are different if routing collectors are present.

Aggregator (Peg)

Maintains data received from the collectors, and offers them through OSSI, or through RF links to the gateway. Maintains a plot of collectors: acknowledges data reception, reconfigures collectors on the operator's demand, responds to queries.

Gateway (Master, sink node)

An optional entity that maintains a network of aggregators and provides connectivity to a remote OSS. Usually, the gateway is a larger hardware platform with relaxed power budget. It is useful when a number of sensor plots (aggregators) are deployed in a given area, or if the deployment is secluded or difficult to access. OSS connectivity may be extended via satellite links, cellular networks, Ethernet, WiFi, Bluetooth, etc., with obvious consequences on the power budget. Large storage capacity may be required, too. Gateways may provide required functionality with vastly different resources; it may be an aggregator with an integrated or attached Bluetooth module and a bigger battery, or it may be a fully functional server connected to traditional networks. Based on intended deployments, an optimal gateway will be proposed as part of the solution.

Router

In a cluster with multiple aggregators, it may happen that the distance between aggregators exceeds the RF range. In that case, routers provide missing connectivity between all aggregators and the gateway. The router is a simple node with all resources devoted to routing packets to and from the gateway.

Redundancy

Eco Net facilitates unattended distributed data collection and aggregation. Depending on the data importance and uniqueness, several redundancy schema may be deployed:

- Redundant sensors can be attached to the collector, and configured such that the *'protecting sensor'* becomes operational only if the *'working sensor'* fails. The collector decides which sensor is *active* or *standby*, thus offering 1:1 protection based only on the initial configuration in the labs. 1:N redundancy is trivial to achieve in highly customized setups.
- Redundant collectors don't seem to add practical value in most deployment and their presence impacts power budget, so this functionality is treated as custom development.
- Redundant aggregators offer 1:1 redundancy for data aggregation. They protect from *'power down'* events or hardware failures. Data storage redundancy or 1:N protection are offered as custom development, as they inevitably have a big impact on power budget, and seem to offer little improvement in most practical scenarios.

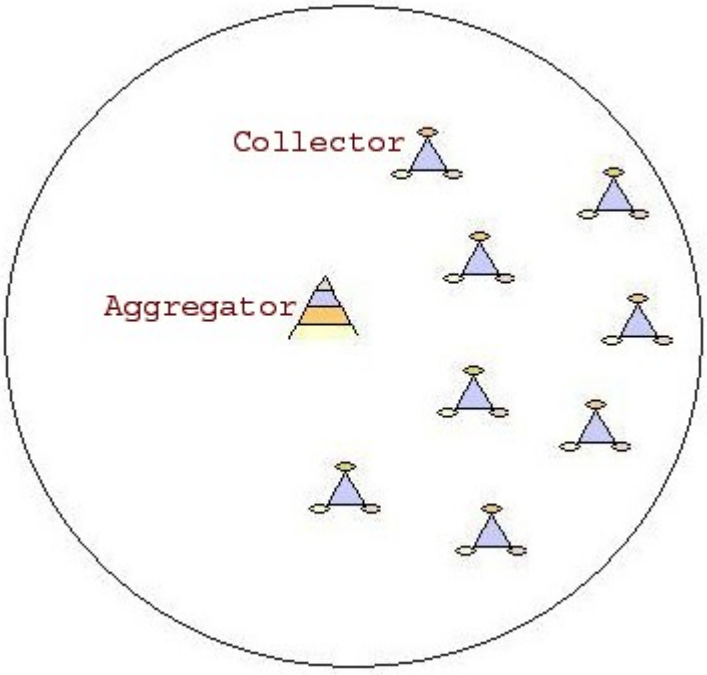
Security

If the collected data are sensitive, or a deployment may be subject to a remote malicious attack, optional authentication and encryption is offered. However, a secure setup requires more elaborate configuration, and takes out flexibility in a deployed network scalability, as most extensions must be assisted by a trusted operator.

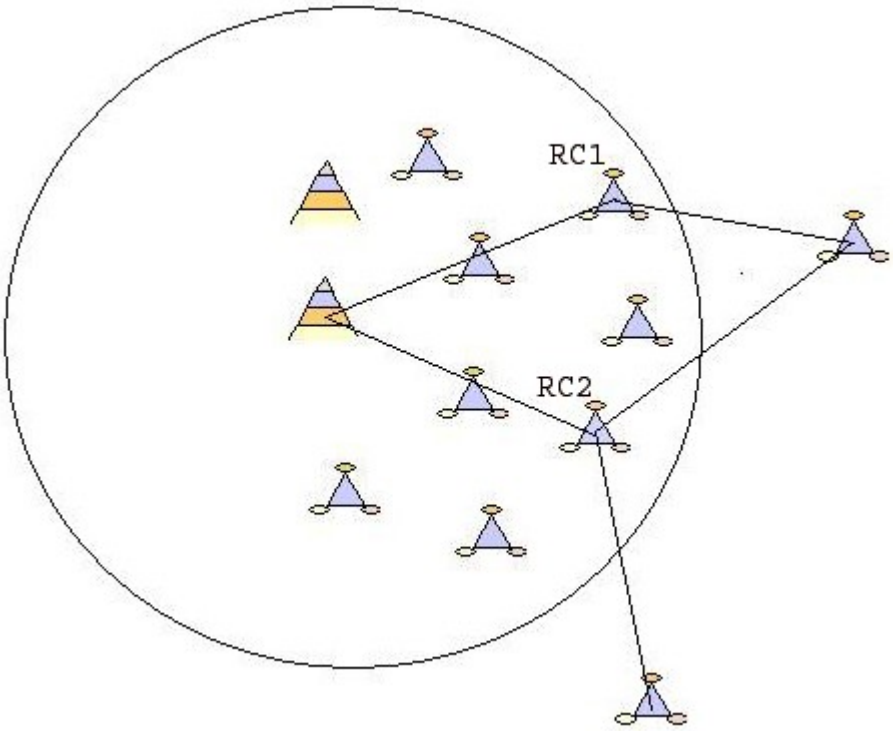
Eco Networks: deployments

The drawings and short descriptions help understand the commonalities and distinctive features of typical deployments.

Independent plots



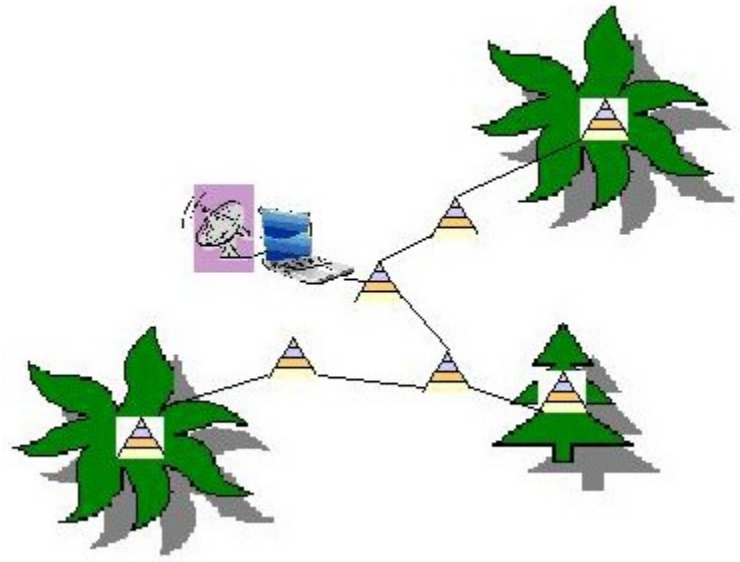
Simple plot, likely closest to today's practices.



Extended plot with redundant aggregators. RC1 and RC2 are designated as routing collectors, active and standby aggregators ensure redundancy on this level.

Clusters

Remote simple plot with gateway. Seclusion or difficult access may justify a communication gateway to traditional networks.



Cluster of plots, with routers and a gateway.

Eco Net Headquarters

Gateways in Brasil, Costa Rica, and North Canada connected via satellite and Internet to an operations room at UofA. It should be noted that from the technical viewpoint this setup is as challenging as the simplest plot. It is more expensive to build and operate, but there are no additional technical challenges within the network. OSS, database, GUI, data processing, etc. are in different, but already explored domains.