Chapter 10
Volcano Monitoring: A Case Study in Pervasive Computing

Nina Peterson, Lohith Anusuya-Rangappa, Behrooz A. Shirazi, WenZhan Song, Renjie Huang, Daniel Tran, Steve Chien, and Rick LaHusen

Abstract Recent advances in wireless sensor network technology have provided robust and reliable solutions for sophisticated pervasive computing applications such as inhospitable terrain environmental monitoring. We present a case study for developing a real-time pervasive computing system, called OASIS for optimized autonomous space in situ sensor-web, which combines ground assets (a sensor network) and space assets (NASA's earth observing (EO-1) satellite) to monitor volcanic activities at Mount St. Helens. OASIS's primary goals are: to integrate complementary space and in situ ground sensors into an interactive and autonomous sensorweb, to optimize power and communication resource management of the sensorweb and to provide mechanisms for seamless and scalable fusion of future space and in situ components. The OASIS in situ ground sensor network development addresses issues related to power management, bandwidth management, quality of service management, topology and routing management, and test-bed design. The space segment development consists of EO-1 architectural enhancements, feedback of EO-1 data into the in situ component, command and control integration, data ingestion and dissemination and field demonstrations.

Keywords Wireless sensor networks · Pervasive computing · Environment monitoring · Quality of service · Situation awareness

10.1 Introduction

New emerging technological advances have opened the door for more sophisticated pervasive computing applications to integrate more robust and reliable techniques. These advancements have led to the development of more powerful physical hardware available at a cheaper price, allowing for greater expansion of the applications.

N. Peterson (✉)
School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99163, USA
e-mail: npicone@eecs.wsu.edu

A.-E. Hassanien et al. (eds.), *Pervasive Computing: Innovations in Intelligent Multimedia and Applications*, Computer Communications and Networks,
DOI 10.1007/978-1-84882-599-4_10,

Our optimized autonomous space – in situ sensor-web (OASIS) application is designed to monitor Mount St. Helens, an active volcano in the state of Washington and provide feedback to Earth scientists, so they have the required information to make crucial decisions (for example, ordering evacuations and air traffic routing). Through the development of our earth-hazard-monitoring sensor-web, we will be able to demonstrate the ability to mitigate volcano hazards through the first "smart" in situ network. In order to accomplish this, we are integrating a continuous feedback loop between two primary components: a ground in situ component and a space component. The in situ ground sensor network is composed of tiny Imote2 sensor motes that collect data from five different sensors: GPS, seismic, infrasonic, RSAM and lightning. We dealt with systems issues, such as power management, bandwidth management, QoS management, topology and routing management and test-bed design issues. The space component currently consists of NASA's earth observing (EO)-1, though we hope to expand to other spacecrafts as well. EO-1 is a satellite that will monitor the volcano from space, providing additional science data in conjunction with the data gathered from the ground component. It is currently operational, and to support the OASIS requirements, several architectural enhancements were made. These include a continuous feedback loop between EO-1 and the in situ component allowing for command and control integration and data ingestion and dissemination. The combined OASIS will have two-way communication capability between the ground and the space assets, using both space and ground data for optimal allocation of limited power and bandwidth resources on the ground, and use smart management of competing demands for the limited space assets.

This case study is a 3 year project with a full scale implementation to be complete in 2009. By August 2008, we expect the first field deployment to occur. This deployment will consist of a self-configuring, self-healing wireless sensor network that will be deployed on Mount St. Helens and linked to the command and control of the EO-1 satellite. The ground sensor-web element will use in situ observations (seismic, gas and ground deformation) to trigger high-resolution data taken by EO-1, which will be down-linked back to the ground sensor-web control centre. These data will be automatically processed and analyzed where the results are ingested into a dynamic and scalable communication bandwidth allocation scheme to optimize communication and power usage.

An active volcano provides a challenging environment to examine and advance in situ sensor-web technology. The crater at Mount St. Helens is a dynamic three-dimensional communication environment, with batteries as the only reliable energy source. Various geophysical and geochemical sensors generate continuous high-fidelity data, whose priority depends on volcanic status. There is a compelling need for real-time data with sensors occasionally destroyed by falling rocks. Hence, an in situ network must be self-configuring and self-healing, with a smart power and bandwidth management scheme and autonomous in-network processing. In order to accomplish this, we have assembled a multidisciplinary team involving

sensor-network experts, space scientists and Earth scientists, who are developing this prototype dynamic and scaleable hazard monitoring sensor-web for applying it to volcanic monitoring.

10.2 Literature Review

Wireless sensor network research has been growing rapidly in recent years [1–3]. Existing applications include habitat monitoring, infrastructure surveillance and environmental monitoring [4–9]. To meet the unique challenges of the volcanic environment, OASIS will significantly advance topology management [10], power and bandwidth management, real-time data gathering and autonomous in-network processing technologies. At present, topology management typically makes use of hierarchical architecture, which enables scalable management [11–14], and flat architecture, which enables better fault-tolerance and more concurrent communications [15,16]. OASIS will introduce a new topology management scheme combining hierarchical control structure with flat routing topology to enable a fully self-configuring and self-healing network. To prolong network lifetime, a smart power adjustment and role rotation algorithm will also be developed. At present, bandwidth [17] and power management schemes [18] typically use contention-based medium access control (MAC) protocols [19,20] and non-priority-differentiated normal routing protocols [19, 21, 22]. OASIS will develop a time-optimal and energy-efficient packet scheduling algorithm to coordinate the traffic, in which sensor nodes autonomously determine communication packet priorities based on mission needs and local bandwidth information. Bayesian network techniques will be applied here. In addition, a data aggregation algorithm will be developed to reduce bandwidth demand. OASIS will also, for first time, take advantage of in situ situation awareness strategies to capture subtle environmental changes in real-time, with the aid of smart bandwidth management. In addition, OASIS will apply a new interactive protocol at the sensor nodes and control centre, as part of integration of both space and in situ sassets.

Automated scheduling and planning environment (ASPEN's) extension to request assets and re-plan in cases where requests are not granted will leverage the negotiation methods used in distributed coordination using the SHAC (shared activity coordination) protocol [23]. This is a generalization of existing re-planning capabilities in ASPEN used to automatically negotiate ground contacts for the EO-1 mission. However, this negotiation approach will utilize a general sensor capability tasking structure and use SensorML (sensor markup language) descriptions of potentially relevant sensors to drive the request generation and backtracking when observation requests are not granted. The existing EO-1 sensor-web demonstrated crude aspects of sensor-web automated response [24]. All of the existing triggers use static combinations of sensors thresholds mapped onto response sensors that are manually defined by scientists. This will significantly extend the capability to distributed control centres and request oriented protocols and capability-based matching of sensors to requests based on SensorML.

10.3 Ground Component

Our ground component is composed of a suite of wireless sensor nodes, each connected to an array of deformation monitoring sensors. These deformation monitoring sensor arrays consist of GPS (global positioning system), seismic, infrasonic, RSAM (real-time seismic-amplitude measurement) and lightning sensors. In order to protect the sensors from the harsh elements, each suite and its sensor array are housed in a box. Each box, shown below in Fig. 10.1, will be dropped by helicopter on Mount St. Helens. In order to allow for proper placement of the boxes on the crater, each is equipped with a metal tripod-like structure, called a spider designed and created by earth scientists at cascades volcanic observatory.

Although the wireless sensor nodes are tiny compact devices, each is equipped with a complex yet lightweight software architecture. This architecture is composed of a link layer, network layer, transport layer and application layer. The lowest level is the link layer that is responsible for maintaining proper connectivity and link quality between nodes. Next, the network layer ensures that the routing of packets, including sensor to sink and sink to sensor, is executed according to the appropriate algorithms. The network layer also implements the smart-broadcasting algorithm, discussed in detail in Sect. 10.3.4.2. On top of the network layer lies the transport layer, which is responsible for the transportation of all packets including both reliable transport as well as best-effort transport. The type of transportation depends on the situation and the goals of the transport. Lastly, the application layer is the top-most layer which is composed of the sub-components: the data sensing module, the network management module and the situation awareness module. The sensing module, discussed in more detail in Sect. 10.3.6, is vital to the overall reliability

Fig. 10.1 Spider being deployed by helicopter

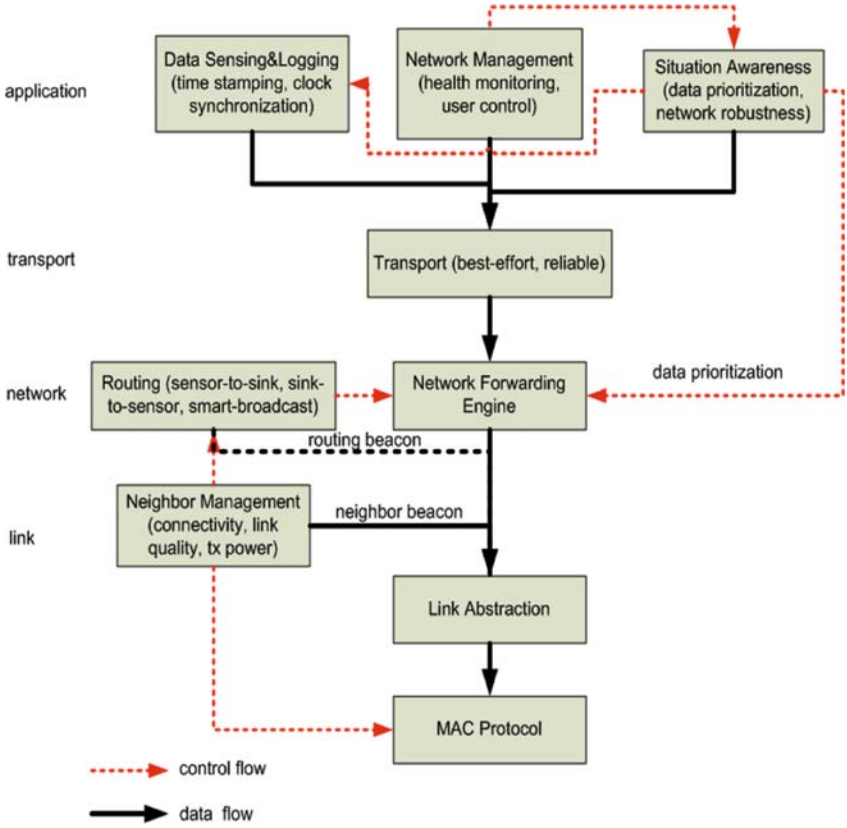

Fig. 10.2 In situ node software architecture graph

of the network as it controls the clock synchronization and timestamping of packets, making sure that the network is in sync. It is necessary that the Earth scientists be able to both monitor and control the network. In order to accomplish this, the scientists interact with the network management module that enables them to monitor the current status of the network as well as make any adjustments to network parameters such as the priority of a particular type of data, as necessary. Fig. 10.2 shows the architectural arrangement of the sensor node's software. Different components of this software architecture will be discussed in detail.

10.3.1 Power Management

The main challenge facing the power management of almost all wireless networks and in particular a volcano sensor-web is striking a balance between the need for high-rate data and the desire for network longevity. The OASIS smart sensor nodes have the built-in ability to adjust transmission power according to local density.

Time-division multiple-access (TDMA) schemes inherently conserve more energy compared to contention-based schemes like carrier sense multiple access (CSMA), because the duty cycle of the radio is reduced and no contention-introduced overhead and collisions are present. We assume that sensors always transmit data to their parent/sink node during their allocated TDMA slot. To save energy, nodes may only need to transmit data after they detect an interesting event. In this case, we optimize the node-to-sink communication scheme to make sure that every node efficiently utilizes bandwidth (and saves power at the same time) when not communicating to the sink nodes. We also developed a dynamic transmission range setting algorithm to maximize energy and communication efficiency in the network.

Considering the space-reuse and power efficiency, we needed to optimize the distance (in hops) between each node and the sink. For instance, if the transmission power is low, the acquired data have to travel through more hops; in such a case, if we use single channel, the end-to-end throughput will be significantly reduced. To handle this issue, we increased the transmission power under the permission of the power budget. If the distance to a sink node is small, we can decrease their transmission power, which is also necessary in terms of relay traffic – the nodes close to the sink usually work more heavily for traffic relaying and can run out of power earlier. To solve the problem, which occurs when nodes closer to the gateway consume more power and therefore, may die earlier than nodes further from the basestation, we have built on ideas from the cluster-based LEACH protocol [21] and developed a cluster rotation protocol to maximize network lifetime. This role rotation algorithm aims at letting every node to have equal duty to deliver data to the gateway, in order to maximize network lifetime. In addition, we are investigating the possibility of deploying some "actor" nodes to relay messages for the sensors, so that their lifetime can be prolonged.

Moreover, the OASIS bandwidth management scheme has the added benefit that it establishes a time-optimum communication schedule, as well as a cooperative power schedule of sensors. Since each node alternates between reception/transmission and sleep states, the wasted energy associated with idle listening and contention is eliminated during normal operation.

10.3.2 Bandwidth Management

Due to the inherent limitations of the wireless capacities of the sensors used in environmental monitoring, bandwidth management is crucial to the overall successful transmission of data from the acquisition point to the basestation. The limited bandwidth between the gateway and the sensor network constrains the high-fidelity operations and the real-time acquisition requirements.

We made use of the "many-to-one" network scenario by applying a time-optimal scheduling algorithm [25]. In this algorithm, each node locally calculates its duty cycle after the initial network deployment, and continuously adjust its schedule if the network topology changes. Every node either sends/receives messages or goes

to sleep, which eliminates the energy waste of idle listening. Another advantage of this scheduling algorithm is the mitigation of interference between concurrent communication pairs. Consequently, both energy and communication efficiency are maximized. A GPS receiver at every node accommodates both synchronization and deformation measurements.

During active periods when bandwidth demands are highest, the network will prioritize the information flow in the network and reserve bandwidth for high-priority data, based on mission-needs. Every node has a buffer management mechanism to differentiate the priorities of different kinds of traffic, and schedule the time slots allocated for each type of traffic. For instance, if during volcanic activity, gas measurements are deemed the highest priority, other data may be buffered to make more bandwidth available for gas data. Cluster coordinators are able to automatically identify and select the minimum set of sensors that will provide mission critical data. Bayesian network techniques were applied to address the sensor selection problem [26].

To further optimize bandwidth utilization, in situ data reduction, compression and aggregation are driven by science requirements. For example, when necessary, seismic data, typically recorded at 100 Hz, will be reduced by two orders of magnitude at the node level by reporting an average real-time seismic amplitude monitor (RSAM) parameter, which is an established measurement of both earthquakes and volcanic tremors [27]. In addition, continuous seismic data will be streamed into a buffer at each node, and when seismic events are triggered, the buffered waveform with precise time markers will be compressed and delivered to the control centre for higher level processing.

10.3.3 QoS Management

The middleware layer provides communication, quality of service (QoS) and bandwidth management for the wireless sensor network. This is accomplished through situation awareness, prioritization and robustness.

10.3.3.1 Situation Awareness

The functionality that the situation awareness component within the middleware layer provides is to ensure that the network is conscious of its environment and responds dynamically to changes. If a physical phenomenon such as an eruption occurs within a particular geographic region of the network, that region is categorized as an area of interest. The categorization of physical phenomenon will take place both automatically as well as manually. Automatic categorization is done through a set of predefined situations described by the Earth scientists. However, in addition to this automated detection, it is also possible for the Earth scientists to manually determine that a physical phenomenon is occurring by examining the data

and categorizing it as a region of interest. Therefore, the sensor nodes within the region of interest are given a higher priority, guaranteeing that data and communication from those sensors are processed promptly and without loss. For example, if lava is emitted from a portion of the volcano, that area of the network is classified as high priority. Since the network is aware that this is a particular area of interest, it makes certain that all data collected from that region are communicated in a timely manner and without loss or delay, even in the presence of network saturation and/or congestion. It is important to note that an area can be categorized as an area of interest due to physical changes detected either by the ground sensors or another external source, such as EO-1. Additionally, the combination of both the ground and space component provides the most sophisticated and accurate representation of the physical phenomena being detected.

10.3.3.2 Prioritization

Prioritization of data and communication within the middleware layer is based on both the context and the situation of the wireless sensor network. The context of the network pertains to the congestion within the network, as well as management of the assignment of priorities to data and communication from each of the individual nodes. In addition, the prioritization component is also conscious of the situation in which the network lies as described in the previous section. Thus, prioritization within the middleware component is twofold, incorporating both the context of the network and the current situation of the network. In order to accomplish this prioritization scheme, the middleware layer implements Tiny-DWFQ.

Tiny-DWFQ is our newly designed tiny-dynamic weighted fair queuing scheduling algorithm, which responds to an automated feedback in order to continually adjust to the current environment and dynamically reconfigure its algorithm in order to reflect the current state of the network. Dynamic adjustments allow Tiny-DWFQ to accomplish its ultimate goal of ensuring high level QoS for the most significant data, while still maintaining QoS requirements for less significant data.

One of the major obstacles in fulfilling this requirement is designing and implementing a lightweight algorithm, specifically for resource constrained wireless sensors, which is robust enough to handle continuous real-time data flows. Due to the limited computational capacity of wireless sensor networks (including memory, processor speed and communication bandwidth), our algorithm is designed specifically to meet these demanding requirements. Bandwidth limitations [28] in any wireless network provide a restricted amount of resources to be shared amongst competing data flows. Thus, in order to maximize the available resources, we designed a congestion reactive scheduling algorithm, Tiny-DWFQ, which addresses the extremely limited memory and computational requirements in its design of the scheduling of packets.

In order to meet the overall needs of many different types of networks, it is necessary for data to be categorized based on its importance or priority level within the network. For example, in our network scenario (OASIS), we are deploying wireless

sensors on the crater of Mount St. Helens for volcanic monitoring and predication. The scientists who monitor the data do not view each data type as equally important, rather specific data (e.g., Lightning) are significantly more important than other data (e.g., GPS). Ideally we are able to transmit all of the data all of the time; however, when network congestion occurs due to bandwidth constraints, a decision must be made as to which data should be sent and in which order. If, due to the large number of packets, it is not possible to send or buffer all of the data, we must also determine which packets should be dropped. These decisions are the responsibility of our Tiny-DWFQ scheduling algorithm.

Within the design of Tiny-DWFQ, there are several critical items which factor into the algorithms execution: data priority, node priority, current congestion and available resources. In our model, we take into consideration both the importance of the type of data and the importance of the node where the data originated. This allows for a fine-grained granularity in our prioritization scheme, which gives the user a more powerful tool to utilize. For example, if activity is detected on the volcano, the data from the nodes within that area are considered more important and the data from those nodes are given a higher priority, than the data from nodes outside the region. However, all of the data from the nodes within the region of interest are not equally important (the lightning data are still more important than the GPS data). Thus, in order to capture this, we need to distinguish between data priority and node priority. In order to accomplish this, we assign a dynamic weighted priority (DWP), which incorporates both data and node priorities.

10.3.3.3 Link Layer Prioritization

Besides implementing prioritization of data and communication within the middleware layer based on network traffic, we also enhance prioritization in the link layer to guarantee QoS. We want to ensure that high priority packets can correspondingly have a higher packet delivery rate towards the sink. The hidden terminal problem in wireless networks can force concurrent transmissions, which cause received packets to become corrupt. We refer to this as the channel failure so that we can distinguish it from buffer failures, which cause packet loss due to buffer overflow. Without any optimizations, the probability of transmission failure with one-hop communication can be up to 50%. Additionally, if the communication is multi-hop, the situation can be much worse. For example, if the one-hop packet loss is p and a mote is of depth k within the communication tree, then its packet loss rate is $1-(1-p)^k$. If $p = 50\%$, $k = 5$, packet loss rate can be as high as 97%, which is unacceptable and results in a waste of both network and power resources. Consider communication that must traverse four-hops, if the transmission of the first three hops succeeds but last hop fails, then all of the energy that was consumed in the transmission of the first three hops is wasted. Thus, in order to provide a remedy, it is necessary for hop-by-hop retransmissions to be carried out in order to reduce the packet loss ratio and conserve the limited energy resources. Moreover, since the priority of each individual packet is different, we have a higher obligation to ensure the successful delivery of

higher priority packets. Thus, in order to accomplish this, we correlate the number of retries for each packet with its priority. For some ordinary packets, one retry is enough; but for those that report certain important events, more retries are necessary. In our algorithm design, we have implemented a mapping that contains a static table for the mapping from priority to the number of retransmissions. The number of retransmissions is predefined based on empirical results obtained through simulations. When transmission failure occurs, the look up table is used to determine whether or not the current retry count has exceeded the maximum retry count for that packet's priority, if it has the transmissions stop; otherwise another transmission occurs.

10.3.3.4 Robustness

The QoS of the system includes, but is not limited to, ensuring fault tolerance within the network. Thus, failure of sensor nodes should and does not hinder other nodes' functionality. A minimal set of sensor nodes is determined to ensure that the network is capable of maintaining functional communication within the wireless sensor network. In addition, the system is able to dynamically recover from node failures exhibiting self-healing mechanisms. Polling and hinting relationships within the routing protocols ensure the desired robustness as well as the service provisions provided.

There are two types of node behaviours which must be addressed in order to ensure QoS: misbehaving nodes and failed (dead) nodes. A misbehaving node is any node that is communicating invalid or erroneous data. These data may be for one particular data type or for all data types. If erroneous data are detected at a node, two things must take place. First, we do not want the limited bandwidth to be consumed by the erroneous data. In order to accomplish this, we must adjust the priority of the bad data flow(s) so that it(they) will be given the lowest priority and be transmitted only if there is no congestion. Second, if a node dies and is, thus, no longer collecting and transmitting data, we still want to be able to predict the data values, which would have been collected by the node if it were functioning. Additionally, if a node is misbehaving, we want to be able to artificially generate accurate data as if the node were functioning correctly.

Currently, complex artificial intelligence algorithms are capable of implementing machine learning techniques, which can predict future behaviour based on past observations. However, it is not feasible to implement these algorithms on each individual sensor due to their limited resources. Instead, we are designing and implementing an algorithm that will be primarily housed at the control centre (a more powerful laptop computer) and will only need to send update messages to individual nodes. Our algorithm is based on the physical characteristics of the environment and the physical properties of the data, the relationship between the nodes and the past behaviours of the nodes. Due to the innate physical characteristics of the data being collected, such as seismic data waves, we can utilize these physical properties and the determined distance between the failed or misbehaving node and its neighbours to estimate the missing or erroneous data. However, some physical

phenomena such as lightning cannot be estimated based on physical relationships between node. Thus, historical data and the relationship between the historical data of neighbouring nodes must be used to approximate the missing data.

10.3.4 Topology and Routing Management

The in situ sensor-web is a *self-organizing* and *self-healing* network capable of responding to environmental changes (such as eruption progression) and dynamically restructuring the communication topology, so that all nodes are able to find alternative energy-efficient paths of reaching the gateway in the presence of node failures. To maximize both the flexibility and efficiency of the network, we combine hierarchical logical control with flat routing topology. The hierarchical control architecture allows us to enable scalable management, while routing can take advantage of all of the physical network connections in order to maximize efficiency.

The routing module provides the logical backbone upon which sensor samples are disseminated to the U.S. geological survey (USGS) data management systems, also known as sensor to sink routing, while command and control elements address and communicate with individual sensors, referred to as sink to sensor routing.

A coordinator in each logical control cluster is responsible for network management and situation awareness. Unlike other hierarchical control architectures, the coordinator does not control routing in clusters. The coordinator role rotates among all of the nodes in the cluster, so that the energy consumption of each node is balanced.

The routing algorithm was developed based on all network connections, without being restricted by a logical control cluster, making concurrent communication within a cluster possible. Under normal circumstances, the data flow follows a many-to-one routing paradigm and forms a data diffusion tree [4], with the gateway as the sink. In a data diffusion tree, each node periodically monitors its parent's status. If its parent dies, it broadcasts a *join request* and waits for other upper level node to accept it as a child. This idea also applies to integration of newly joining sensors, eliminating the need to reconfigure the network as it grows. The routing protocol of command and control flows (e.g., network management and situation awareness) was developed based on geocasting protocols [29, 30], which enable interaction according to geographical positions and boundaries.

One key aspect of the routing module is its ability to adapt autonomously to both changing conditions within the sensor network, such as topology updates and link-quality changes, and application concerns, for example, increased network activity due to event detection. Another key component of the routing module is its ability to balance application network demands with overall network service level. Additionally, the routing module also provides a consistent interface to the OASIS application modules in order to minimize the need for cross-layer network programming.

The routing module is responsible for autonomous network formation and network topology maintenance over the life of the sensor network, referred to as self-organizing. Upon deployment, a sensor node detects that its routing table is empty and begins interaction with the MAC module to determine its neighbours and potential paths towards the sinks. Probes are then sent towards the sinks in order to verify that such a path is available. The sinks acknowledge the probes, thereby completing the network initialization for that sensor node and populating the sink's routing table with the information required for sink to sensor routing. Network formation is complete once all of the sensor nodes within the network have determined at least one path to a sink, which indicates that all sensor nodes can be addressed by the sinks.

The routing module autonomously constructs a stable routing topology, both from any initial state and when additional nodes are deployed. Nodes may join the network at any time, and upon joining they are incorporated into the routing topology. Node removal is also handled autonomously. The only a priori logical or geographical information required during network formation is the identification of sink nodes. In order to avoid contention and make efficient use of the available bandwidth, the routing topology makes use of multiple paths whenever available.

Once the routing topology has been established, the routing module provides three different types of routing to the application layer: sink to sensor, sensor to sink and reliable broadcast. The type of routing to be used is requested by the application layer when a packet is sent. All routing types employ multi-hop routing techniques as needed to achieve successful packet delivery. Nodes between the source and destination, referred to as intermediate nodes, are responsible for the appropriate prioritization of traffic based on both the traffic type and the bandwidth considerations.

Sink to sensor routing is an explicit one-to-one routing, using the MAC address of the sensor as the destination address. Sensor to sink routing is a logical routing algorithm wherein the sending application only designates the destination as its sink. The determination of which of the potentially multiple sinks will be used is done by the routing module. Sensor to sink paths are weighted initially by the neighbours' signal strength and number of hops to the sink. These weights are continually adjusted based on interactions with the MAC module in order to ensure that new and missing neighbours are detected and an optimized route is available. When the first hop of the primary sink-path is determined to have insufficient bandwidth, the routing module employs multiple paths to deliver sensor traffic to the sinks. Because sink routing is logical, this mechanism is transparent to the application layer.

Reliable broadcast requires guaranteed delivery by the routing module; whether guaranteed delivery is supported for the other routings is to be determined based on design considerations within the MAC module and the application layer. Reliable broadcast provides a multicast mechanism that addresses all sensor nodes. Packet delivery is guaranteed to ensure coherent command and control.

In addition to being self-organizing, the routing module is also responsible for assuring that the network is self-healing. Thus, the network must continue to function

for as long as a network path is available. The routing module must appropriately select new routing neighbours to overcome temporary environmental conditions, which affect link quality. Additionally, the routing module must react according to the permanent loss of a network node.

The routing module plays an active role in the environmentally aware nature of the OASIS project's in situ network by cooperating with the sensing and data-delivery applications to deliver the highest service level possible given the current network conditions. Application data prioritization rules are encapsulated in order to make autonomous decisions about discarding data to preserve the overall health of the network and provide fair queuing to all sensor nodes. The routing module exposes an interface to the command and control elements to allow for explicit prioritization rules to be established, for example, providing preferential treatment for a specific node or subset of nodes. The routing module's policy interface enables command and control overrides of autonomous bandwidth management decisions. Bandwidth management requires that the routing module be cognizant of the nodes states, the network state and of the nature of the traffic being routed. Intermediate nodes have to make traffic prioritization decisions based on local congestion and memory considerations. The semantics of application interaction, for example, whether to queue locally or throttle the sender, are determined. In order to provide the desired QoS, the routing module can request reduction in transmission power from the MAC module as appropriate when full radio power is unnecessary. The routing module facilitates interactions between the MAC module and the application layer in order to conserve the limited power resources whenever possible. One mechanism for accomplishing this is through packet queuing, which allows for a reduced radio duty cycle. This also facilitates bandwidth management and congestion management provided by the MAC module. Packet queueing is employed both at the source and at intermediate nodes.

10.3.4.1 Sensor to Sink Routing

In order to enhance our system's performance, we have optimized the routing protocol MultihopLQI, which is employed for sensor to sink routing in our system. MultihopLQI is distance vector routing protocol. It is contained in the TinyOS-1.x distribution using a link quality indicator (LQI) as its routing metric. The LQI measurement is a characterization of the strength and quality of a received packet. The LQI value is generated using a combination of the received signal strength indicator (RSSI) and the correlation values for the first eight symbols of each frame used with the CC2420 radio. Once the LQI value has been generated, an additional formula is used to correlate this into the one-hop cost for each link. The total path cost for each node is the sum of the path cost for its parent and the one-hop cost to its parent. MultihopLQI uses Bellman Ford's algorithm to discover the ideal path to the sink, which is the one with the lowest total cost.

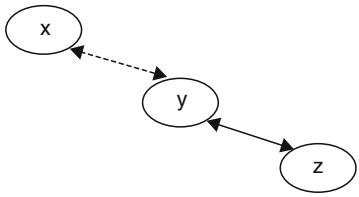

Fig. 10.3 Count-to-infinity problem

Count-to-Infinity Problem

MultihopLQI as implemented in TinyOS1.x has the inherent count-to-infinity problem typical of distance vector routing protocols. The count-to-infinity problem occurs when a link is broken between two nodes, say x and y, shown below in Fig. 10.3.

If node y, which is linked to node z, looks in its routing table, it sees that node z has a path to node x which is two hops. However, node y has no way to know that this path is through itself, and thus, when its link to node x was broken, so was node z's link to node x. Not knowing this, node y updates its routing table to reflect its new hopcount of three (two hops to from node z to node x and one hop from node y to node z). When node z sees this update, it will update (increase) its hop count to node x, since its path is through node y. This will continue resulting in the count-to-infinity problem.

Destination-sequenced distance-vector (DSDV) routing is a protocol that can be used to avoid this path looping problem. However, DSDV has the problem of slower convergence because a delay of advertising routes is introduced to dampen topology fluctuation. In order to address this, we have developed a method that uses time to live (TTL) in conjunction with both the sequence number and source address to detect loops which are occurring. When a packet is initially sent out from the first hop, the TTL field (5 bits) in the routing layer header is initialized to 31. The TTL value is decreased by one each time a packet traverses an intermediate node. When the TTL value becomes zero, the packet is discarded. Each sensor node maintains a history queue, which records the source address, the sequence number and the TTL of all forwarded packets in its history. Before forwarding a packet, the node refers to its history queue. If there already exists an entry that has the same source address and sequence number, but a different TTL, then it is flagged as a loop. The TTL portion is necessary in order to distinguish between an actual loop and the retransmission of a packet. For example, if for some reason a sender fails to receive an acknowledgement (ACK) for a packet, the sender will retransmit the packet. If TTL was not employed, there would be no way to distinguish this scenario (which is not a loop) from an actual loop (such as count-to-infinity). However, with the implementation of TTL, this misjudgment of loops can be avoided. If a loop is detected, MultihopLQI breaks it by invalidating the path to the sink and proactively discovers a new path.

Link Metric Estimation Enhancement

MultihopLQI uses LQI as its link metric. Through our own experimentation, we confirmed the widely reported belief that LQI has high variation in its time scale. More specifically, our experiments showed that instead of using a simple LQI, if the average LQI is taken, it is able to more accurately reflect the packet delivery rate (PDR). Thus, we accomplished this by applying the exponentially weighted moving-average (EWMA) to the neighbour management module in order to smooth out the values of the LQI within each time frame (window). We choose 0.25 for α. Hence, when a beacon arrives, instead of using the LQI value of that packet, the routing layer requests and receives the average LQI cost for the entire path from the neighbour management module. This calculation is shown below in (10.1)

Average LQI path cost

$$\text{LinkEst}(t) = (1 - \alpha) \times \text{LinkEst}(t - 1) + \alpha \times \text{LinkEst}(t) \tag{10.1}$$

Alternative Path Backup

Sometimes particular sensor nodes within the network become overloaded or congested. Additionally, environmental conditions may cause interference, for example, ash cover for particular nodes. In these instances, not only will the node experience temporary loss, but their children will also be confronted with successive transmission failure because their buffer queues will quickly become full and forwarding packets will not be able to be inserted. This situation is extremely serious because the node may still be able to receive beacons from its parent in spite of transmission failure, and assume that its current link quality is good (when in fact it is not). We have addressed this problem by using the cooperation of the neighbour management module in order to back up all of the available paths. But we acknowledge that many sensor nodes are RAM-strict. Thus, in a network with high density, a node could potentially have more neighbouring nodes than the neighbour table can accommodate. If this is the case, it is necessary to apply a replacement strategy. In our replacement strategy if the table is full, then the table's entry, which contains the lowest link quality value, is replaced with entry containing the higher link quality.

Another important situation in which we must determine a backup path is if link failure occurs due to the parent node being "dead." In our design, if ten successive packets fail to be sent out successfully, the link is judged as dead. The backing up of this path is accomplished by the neighbour management module. When a beacon message is received, the path cost and source address of that message are passed into and stored in a table in the neighbour management module. Thus, when the network layer needs an alternate path, it makes an inquiry to the neighbour management module for a new parent. If there is a new parent available, it is returned.

Fast Network Self-Organizing and Updating

When a node detects that a loop has occurred, it will invalidate its current parent. Next, the node must wait until the new beacons from other nodes arrive in order to discover a new path. The length of time that the node must wait for the discovery of a new path is directly proportional to the length of the beacon period, in most cases, beacon packets are issued periodically. A comparatively longer beacon period can reduce extra communication traffic, but it also makes nodes in the network insensitive to links, which have become broken, and results in latency in the route discovery of new paths. A long beacon period will cause the node to remain in a waiting state in which it will experience high packet loss. However, a short beacon period also has the disadvantage of introducing additional overhead through increased traffic. Thus, a balance must be achieved in choosing the proper beacon period that is neither too fast nor too slow in order to speed up the network's self-organizing process.

In order to accomplish this, we disseminate good route information. This means that if a sensor node changes parents (from TOS_BCAST_ADDRESS to a valid node address), it will send out a beacon immediately to notify its neighbours of the new route information. In this way, it takes only a short time for the network to form. This reduction in self-organization and updating of the network is particularly effective during the initialization phase of the network formation. Just as it is necessary to disseminate good route information to neighbouring nodes, it is also desirable to update, in a timely manner, bad route information to neighbouring nodes. When a node does not receive a route beacon from its parent for more than three successive beacon periods or its transmission for ten successive packets fails, then the routing module begins the process of switching parents. If the process of switching parents fails, then the node's parent is set to TOS_BCAST_ADDRESS (which is the default upon initialization) and it proceeds to immediately send out a beacon with its infinite path cost. Additionally, if a node receives a beacon from its current parent while its grandparent is TOS_BCAST_ADDRESS, then it also sets its parent to be TOS_BCAST_ADDRESS and sends out a beacon message to notify the other nodes.

10.3.4.2 Reliable Broadcasting

Disseminating common information to all or a subset of the nodes in the network is often an integral part of many wireless network operations. Examples of situations that require reliable broadcasts include code updates, query of interested events, network maintenance and resource discovery. The reliability and speed of data dissemination are important aspects of protocol behaviour in wireless networks.

In environmental monitoring applications, absolute reliability in disseminating data, including both code updates and user command and control, throughout either the entire network or subsets of the network is required. Additionally, the delivery of both data and retasking commands must be accomplished within an acceptable period of time, as determined by the specific application. Due to the important role that each individual sensor plays in the overall network operations, it is vital that

all of the sensors receive all of the data within a relatively short period of time (in terms of cycles per period). For instance, in a volcanic monitoring application, scientists might want to change the agent software so that all of the seismic sensors sampling rates are updated from 100 to 120 Hz. It is vital that this command reaches all target stations in a short time period; otherwise it will result in a system failure, as the network is not satisfying the user's needs. Network code updates are another important application that demands reliable and fast data dissemination.

Data dissemination in wireless networks is achieved via each node broadcasting the data to its neighbours, thereby taking advantage of the shared wireless media. A straightforward approach for broadcasting is blind flooding, in which each node is required to rebroadcast the packet whenever it receives the packet for the first time. Blind flooding is, therefore, simple and easy to implement. However, it may generate many redundant transmissions, which can cause serious broadcast storm congestions. Furthermore, it cannot guarantee broadcast reliability. Reliability of dissemination can be defined as the percentage of all nodes that successfully receive the information. Data dissemination losses can result in performance problems for flooding, and ultimately in routing and applications. Unreliable flooding can result in slow setup for query-response applications, slow discovery of new routes, creation of inferior data collection trees and incomplete information for target tracking and signal processing. Additionally, unreliable data dissemination is unacceptable for code updates and user command and control operations. Thus, reliable unicasts in wireless networks are often implemented via link-layer automatic repeat-requests (ARQ) (via an RTS-CTS-data-ACK exchange) in order to protect unicast traffic from collisions and corruptions; however, ARQ for broadcasted traffic causes control traffic implosion.

Sink-initiated data dissemination usually carries the user's command and control or code updates. Hence, it is necessary to ensure 100% reliability in a short time period. Otherwise, from a user or external system viewpoint, it results in a system failure. Motivated by this need, we invented the cascades protocol as the first-known reliable and fast data dissemination protocol in sensor networks. The cascades protocol leverages portions of the existing data gathering routing protocols, such as multihopLQI and drain, by utilizing their data gathering tree structure to implement effective routing dissemination.

Cascade implements the parent-monitor-children analogy within its tree structure to ensure 100% reliability: each (parent) node monitors whether or not its children have received the disseminated messages through implicit listening (e.g., snoop child's broadcast) or explicit ACKs. Each parent will rebroadcast periodically until successful receipt of each of its messages is confirmed. Cascades is also a smart opportunistic routing protocol, and thus, data flows do not necessarily follow the routing tree structure and a node does not necessarily wait for disseminated messages from its parent. Instead, in order to speed up the dissemination process, a node may snoop a message from nearby nodes rather than wait to receive its own copy. For example, in Fig. 10.4, node a is the parent of nodes b and f. However, node a's data only reache node b. Then b rebroadcasts the data to its neighbours (also as an implicit ACK to node a). At the same time, node h receives data from

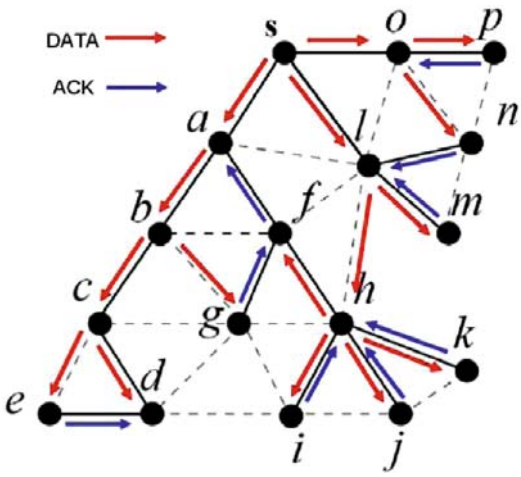

Fig. 10.4 Illustration of opportunistic broadcast flow in a tree with Cascades protocol. The *solid lines* sketch the tree structure, and the *dashed lines* show the other network links

node l (though l is not node h's parent), and rebroadcasts the data to its neighbours, which includes its parent, node f. Thus, the speed of the dissemination process is increased. It should be noted that the cascades protocol does not necessarily need a specific data gathering routing tree to be successfully implemented, rather it can be executed with any connected spanning tree. The reason that cascades does not require a data gathering routing tree is because it only utilizes the parent-children relationship within the tree to ensure reliability, and thus, any tree will suffice. Additionally, cascade's dissemination flow is opportunistic and nondeterministic for different sessions. Cascades is a unified solution for sink-initiated reliable data dissemination, including broadcast, multicast and unicast.

10.3.4.3 Pipelined Sending and Receiving Between Layers for Optimization

TinyOS is an even-driven system that does not support multithreading. As recommended by TinyOS, an upper-layer component should not send another message to a low-layer component until it receives a sendDone event indicating that the previous message has been received. The sendDone event lets the upper-layer know that the lower-layer has successfully received its message. Figure 10.5a illustrates one of the timing issues of this simplex approach, which can cause a large gap between two consecutive sends. We propose the pipelined sending approach to mitigate this gap. It is illustrated in Fig. 10.5b. Similarly, for message receiving, a low-layer component cannot receive another message before the upper-layer component returns the previous message (or swaps it for a free message space).

Our proposed pipelined sending/receive approach is realized through buffering messages in sending/receiving queues at each component. Pipelined sending/receiving helps to improve the performance of the communication stack, since

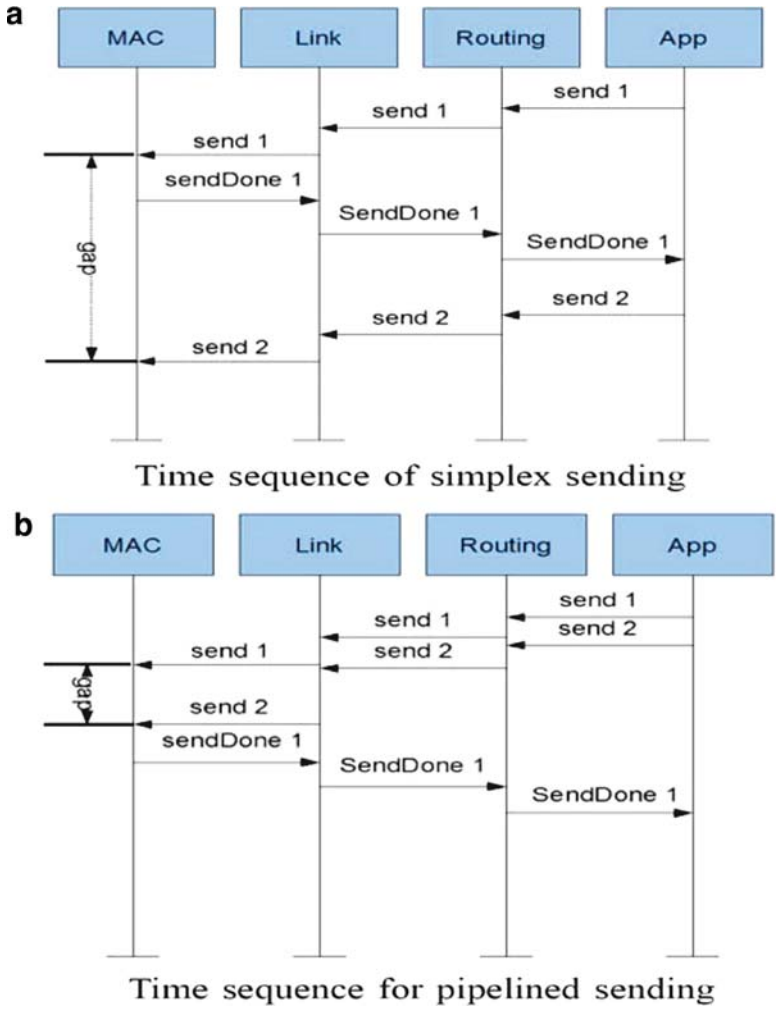

Fig. 10.5 TinyOS communication stack optimization: (**a**) Time sequence of simplex sending and (**b**) Time sequence for pipelined sending

when using a TDMA protocol, once a node gets allocated a time slot, the node should use the entire time slot efficiently, especially when high throughput is demanded. However, in the simplex approach, when the MAC layer sends a packet, the MAC module will signal up a sendDone message, usually via posting a task. Thus, only after the original sender processes the sendDone event, it will make another send request to MAC component. Because of this, the resource constrained channel's time is wasted. However, in our design, we overcome this problem by adding queues to each communication component. By doing this once the MAC module finishes sending out one packet, it will immediately fetch the next packet in the queue, instead of waiting for the upper-layers to send down a new packet.

For receiving, we added a link layer buffer pool and receiving queues in each communication component to support pipelined receiving and reduce the packet loss ratios.

10.3.5 Sensor Node Development

The staff at the USGS cascades volcano observatory have designed, prototyped and tested the data acquisition, sensors and communication hardware for the in situ sensor nodes. As part of this design process, early input from geodesists and seismologists was incorporated to ensure that the appropriate sensors and capabilities were included. Use of commercial off-the-shelf (COTS) embedded microcontroller modules with high level programmability enabled rapid development and application of an affordable platform. An expansion circuit and printed wiring board were designed and produced to include power conditioning and control, sensor input multiplexing, signal conditioning, signal digitization and communications interfaces. COTS sensors include an L1 GPS receiver for timing and deformation monitoring, seismic accelerometer, microphone or microbarograph for infrasonic detection of explosions and emissions, lightning detector for ash cloud detection. Telemetry between nodes is either IEEE 802.15.4 or 900 MHz ISM spread spectrum given the specific link requirements.

10.3.6 Smart Sensing

Our sensing module is the first step in acquiring data from the environment. When working in harsh environments such as volcanic monitoring, the long-term continuous running of the network and the high frequency sampling scenarios make it particularly challenging for the sensing module which, if not designed properly, can become the weak point of the entire system. A broken sensing module will cause the network to do unnecessary work resulting in useless data transmissions. In order to satisfy the requirements of the seismologists of USGS and NASA JPL (jet propulsion laboratory), the in situ network as a part of our OASIS project was designed with a sensing module that supports real-time high frequency data monitoring for an extended period of time. Below we will discuss our design principles and implementation in the development of our sensing module.

10.3.6.1 Sensing Module Challenges

Due to the nature of our project, many of the challenges that we faced are unique to our project. For example, until recently in traditional wireless sensor networks, the limited resources of the sensor devices have prohibited both high sampling rates

and real-time data streaming. However, through recent advances in technology this challenge is now overcome. Thus, we have implemented high sampling rates and real-time data streaming into our sensing module. Now we will delve further into the design principles that we employed when we faced the following five major obstacles: high sampling rates and real-time data streaming, synchronized sampling, event detection, multiple sensors and a harsh environment.

High Sampling Rates and Real-Time Data Streaming

In contrast with low-data-ratio applications, we have the requirement of sampling at 100 Hz (a very high data rate) with 16-bits of data resolution. This brings a high workload to the system. High frequency sampling also faces competition for processor resource from real-time data streaming transmissions. Additionally, sensor sampling and data processing (analyzing, saving and packaging) must be finished within a short period.

Synchronized Sampling

Synchronized sampling is necessary in order for seismologists to make comparisons across the whole network. It is the desire of the seismologists, and thus, one of the requirements of our project to have all of the nodes samples with a particular sensor, say infrasonic, at the same time. In order for the comparisons to be accurate, the difference between timesamples must be less than 1 ms. Although each node is equipped with GPS for time synchronization, we still cannot guarantee that all nodes will perform synchronized sampling due to multiple time delays that occur within the network. We did not adopt a time synchronization protocol into our system because in addition to time synchronization, geologists also wanted to be able to obtain the position information of each node in order to monitor the movement on different portions of the volcano. Since we are already implementing real-time data transmissions, it is necessary for the system to reduce other types of network communications, and adding a time synchronization protocol to our system would greatly increase network communications. Instead, we have implemented low power consuming GPS sensors and a high quality battery supply, which can guarantee long-term system health and extend the lifetime of the network.

Time Synchronization

In this section, we describe the implementation of our time synchronization protocol as one key component of our smart sensing module. Our OASIS time synchronization protocol is a GPS receiver based generic protocol. It can be implemented on any sensor node equipped with a GPS receiver. In general, GPS based synchronization removes the uncertainty problem that arises with radios; however, we cannot claim that the high accuracy synchronization is achieved unless we address the uncertainty

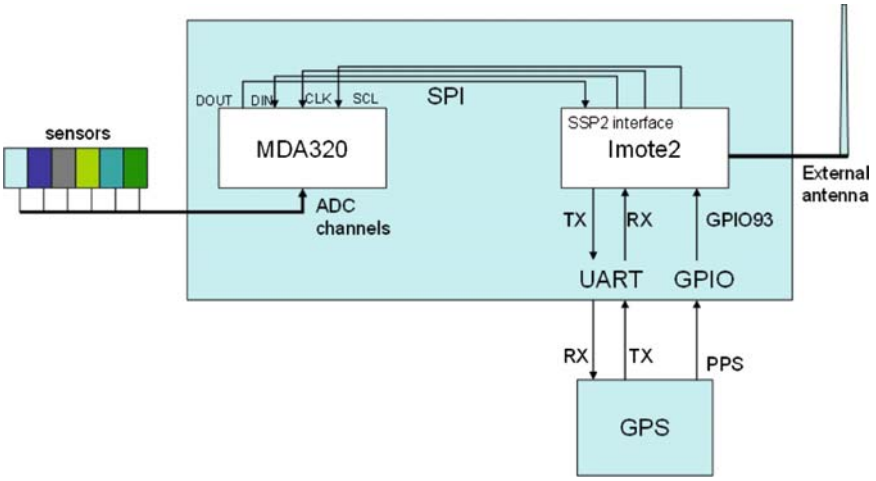

Fig. 10.6 Hardware components connection

resulted from interrupt handling, clock accuracy, mote clock drift, long-term running and environmental changes. In addition, OASIS's time synchronization protocol was not developed solely for time stamping, rather it is also used as the timer driver for the sensing module, as shown below in Fig. 10.6.

In order to fully comprehend this architecture, it is necessary to have a brief overview of the construction of an Imote2 sensor node. The Imote2 sensor node is a newly developed sensor node, which contains a powerful CPU and a relatively large memory space. Each sensor node is capable of running either the TinyOS operating system or a Linux operating system. The processor of Imote2 is an Intel PXA27x processor that provides multiple timer channels and general purpose input/output (GPIO) resources. Both the operating system timer and the GPIO interface are necessary for the implementation of time synchronization. The operating system timer, PXA27x, provides eleven clock channels in which the clock resolution can be adjusted from 1 μs to 1 s. In order to generate a low granularity clock, we use the OS Timer channel 10 as clock source, with the resolution set to 1 μs. Although there are many GPIO interfaces provided through the PXA27x, we can only use three of them for our customized application due to the design of the Imote2 sensor node. In our implementation, we use the GPIO 93, shown in Fig. 10.6, to capture PPS (Pulse Per Second) signal.

Now that we have an understanding of the architecture of the Imote2 sensors, we will discuss the design of our OASIS time synchronization protocol, shown below in Fig. 10.7. Our protocol is a three phase synchronization protocol. In phase 1, the GPS readings and coordinated universal time (UTC) preparation are performed. Phase 2 consists of real-time setting. Finally, phase 3 is responsible for ensuring that the errors associated with the time are self-keeping.

The time synchronization protocol is composed of two components, the real-time timer and the GPS sensor. The realtime timer is a timer that is fired every one

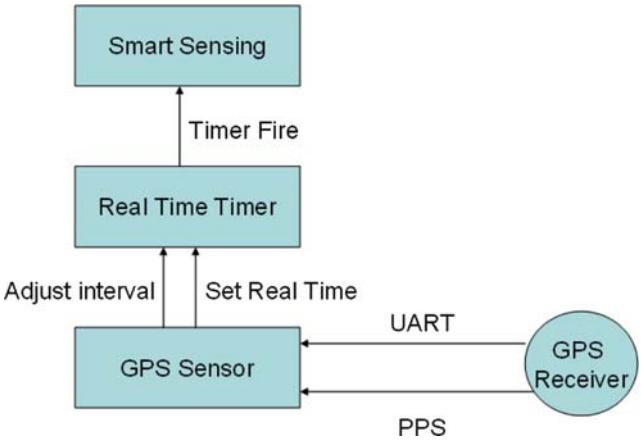

Fig. 10.7 Synchronization process

thousand microseconds. After each fired event, the UTC time, maintained by the realtimetimer module, will be updated. It should be noted that UTC time is formatted hour:minute:second:millisecond. In addition to updating the UTC time, the timer interval may also be adaptively changed according to the offset value in the time fire event. The GPS sensor acts as the GPS data receiving agent and PPS capturing agent. GPS data are configured to arrive every 10 s and the PPS signals are configured to arrive every 1 s. The PPS signals are nanosecond accurate. GPS data arrive several milliseconds later then PPS. Thus, it is not wise to immediately set the UTC time after getting the GPS data. After the first PPS event is detected, the application moves into a synchronous waiting state. The next incoming GPS data packet will only be processed if the application is in the synchronous waiting state. After the GPS data processing has taken place, when each successive PPS event arrives, the UTC time in the Realtime Timer module will be incremented by 1 s.

Event Detection

Event detection is crucial to the overall system performance since, at times in order to reduce the network traffic, it is necessary for the system to stop real-time data transmissions for a short period of time. However, during this period, the system still needs to detect and transmit all of the big events back to the base station. In order to accomplish this, our event detection algorithm was implemented. Within our algorithm, it is necessary to achieve fast in-node data analysis to accomplish event detection, while helping to reduce the data transmission by only sending out the data from big events and remaining silent when insignificant events take place.

Initially, we decided to use the following method to implement event detection. Every time an RSAM value was sampled, we would compare it to the average of the last 400 samples. If the ratio was over some event threshold, we would consider

it to be an event. However, this schema requires lots of computation and disobeys our fast sampling principle, thus we decided to use another approach that employs hardware support. In order for this approach to be realized, we first had to introduce a new sensor, the long-term RSAM (LRSAM) sensor. This sensor is composed of one short term RSAM sensor and some additional delay components. Readings from this sensor give us an average sensing value for each 20 s period. By simply reading this value and comparing it to the short-term RSAM value, we are able to greatly reduce the necessary calculation. LRSAM is read every second, and SRSAM is read every 10 ms. Therefore, we define the occurrence of an event, either large or small, using two lemmas:

- Lemma 1 Small Event: SRSAM is not greater than twice the LRSAM value.
- Lemma 2 Large Event: SRSAM is greater than twice of the LRSAM value.

Note that if a large event is detected, one period of data must be sent. After detecting this event, the application sends out the timestamped sensed data plus the two previous seconds worth of data. This additional 2 s ensure that the data being transmitted capture the entire earthquake.

Multiple Sensors

OASIS requires the simultaneous sampling of multiple sensors. Our sensing module is required to process multiple sensors data without any data fusion, which introduces resource competition for memory allocation, timer allocation and the sequence of processing and transmitting the acquired data. For OASIS, the following sensors and sampling rates are required:

1. Seismic sensor: 100 samples per second (100 Hz sampling rate), 16 bytes per sample (200 Bps)
2. Infrasonic sensor: 100 samples per second (100 Hz sampling rate), 16 bytes per sample (200 Bps)
3. RSAM sensor: 1 sample per second (1 Hz sampling rate), 2 bytes per sample (1 Bps)
4. GPS sensor: 1 sample per 10 second, 200 bytes per sample (20 Bps)
5. Lightning sensor: 10 samples per 1 second, 2 bytes per sample (20 Bps)

Thus, the total size of the raw data being sampled and transmitted is 440 bytes per second, not including network message headers.

In order to accommodate the multiple sensors, we will sample synchronically from all of the required sensors. Additionally, the earth scientists will have easy interactive control of the network, which will be achieved through a centralized command and control centre. This will allow the user to adjust the sampling rate and priority of individual sensors as well as temporarily turn sensors on and off.

Harsh Environmental Conditions

The harsh environment produced from the networking being deployed on an active volcano provides an additional challenge for the sensing module. Due to the harsh conditions, it is not unusual for sensors to be destroyed. Additionally, debris from broken sensors can cause interference. Thus, it is necessary to implement an error detection scheme and a self-healing scheme in order to reduce these adverse effects. If an error is detected, depending on the type of error, either the self-healing scheme needs to be executed or else a report of the errors must be sent to the base station.

Error Detection

In order to accomplish error detection, one timer will be dedicated to periodically checking the sensor data. This timer will continually check the last fifty data samples and get following information: the maximum value, the minimum value, the average value, the maximum deviation and the average standard deviation. The combination of these five values will indicate the type of error that has occurred. There are four main types of errors that we will detect: a sensor board disconnection error, a sensor disconnected from the sensor board error, a broken sensor error and a battery offset error.

In order to detect that a sensor board has been disconnected, the maximum and minimum values are examined. If both of these values are the same, and more specifically 0xffff, then it is determined that the sensor board has become disconnected. If we detect that the sensor board is disconnected, we will turn off the node and send an event report packet to the control centre.

It can be determined that a sensor node has become disconnected from its sensor board if the ADC channel is sampling noisy data, which will be in accordance with the current time, and whether or not the volcano is experiencing activity. However, the noisy data must also follow a pattern. The maximum deviation value and the maximum average deviation value must both exceed the maximum threshold. Additionally, the maximum and minimum sampling values must also exceed the normal range. When it is detected that a sensor is disconnected from its sensor board, the node will be turned off and an event report packet will be sent to the control centre.

If a sensor is broken, the value will either jump to an extreme value, high or low, because of the destroyed resister or the error will be similar to that experienced when a sensor-disconnected error occurs. Additionally, if a sensor is broken, its average value may drop below the minimum threshold value. If a broken sensor is detected, the sensor will not be closed immediately, instead we will change the value of the detected count and then, if this count goes over the predefined threshold, the sensor will be closed and a report will be sent to control centre. However, even if a broken sensor error is detected in a node, it is still possible that it is a false positive detection. Thus, centralized detection calculations will also be added after analyzing the data. Note that if the scientists decide to continue running that sensor and later errors are detected, the sensor cannot be turned off again.

Battery offset detection cannot be detected in-node because the drop will also affect the reference voltage in the node. This error can only be detected in the control centre by comparing the sensed data with other neighbouring nodes' sensed data.

10.4 Space Component

One of the primary goals of OASIS is to implement a two-way communication channel between the ground and space components. The motivation for this two-way communication channel is optimal allocation of the limited power and bandwidth ground resources, while also providing smart management for the competing demands. Just as we would like to have unlimited power and bandwidth resources available for the ground sensors, it would be ideal to have space assets available to meet every demand and desire of the Earth scientists, such as providing a continuous streaming of data products from space assets. However, given the multiple demands on satellites, this is not feasible. Thus, it is crucial that resources are used strategically. Some of the questions asked by Earth scientists requiring satellite data are:

- What is happening?
- Where is it happening?
- What is the magnitude of the event?
- What is the change, or rate of change?

The space component consists of JPL sensorweb ground software and the EO-1 satellite, shown below in Fig. 10.8.

The EO-1 satellite is the first mission of NASA's new millennium program earth observing series, managed from Goddard Space Flight Center. Designed as a testbed for the next-generation of advanced land imaging instruments, EO-1 was launched on a Delta 7320 from Vandenberg Air Force Base on 21 November 2000 into a 705 km circular sunsynchronous orbit at a 98.7° inclination.

EO-carries three instruments: the advanced land imager (ALI), the hyper-spectral Hyperion Imager and the atmospheric corrector (AC). The ALI combines novel wide-angle optics with a highly integrated multispectral and panchromatic spectrometer to demonstrate spatial and spectral resolution comparable or improved from Landsat at substantial mass, volume, and cost savings. The Hyperion is a high-resolution imager capable of resolving 220 spectral bands (from 0.4 to 2.5 μm) with a 30-m spatial resolution. The instrument typically images a 7.5 km by 42 km land area per image, and provides detailed spectral mapping across all 220 channels with high radiometric accuracy (ASE uses the products from the Hyperion instrument for onboard science processing). Finally the EO-1 AC provides the first space-based test of an AC that is – designed to compensate for atmospheric absorption and scattering, allowing for increased accuracy of surface reflectance estimates.

The requirements of the space component are twofold. First, it is responsible for responding to inquiries regarding general capabilities. These are inquiries regarding generic sensor capabilities such as providing information on the data pedigree, the

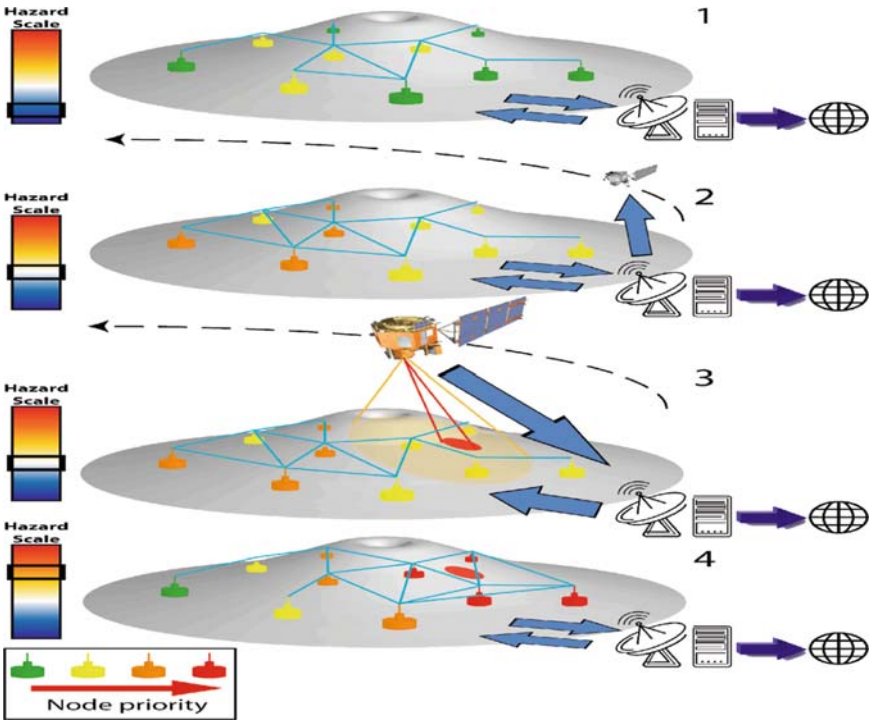

Fig. 10.8 Optimized autonomous apace in situ sensorweb

number of frequency bands, the spectral resolution of the image, the spatial resolution of the image, the wavelength range of the data and the signal-to-noise ratio of the data. Second, it is also responsible for tasking all requests made by Earth scientists for data acquisition, processing the data and generating any alerts derived from the data. Specifically, the space component receives alerts from the in situ sensors. Upon receipt of an alert, the space component attempts to issue a request to EO-1 to acquire data of the region of interest. Observation contention is resolved by first determining if the priority of the new request is greater than the existing observation requests in the schedule. If the priority is greater, the new request is uploaded to EO-1 for execution. Once the data has been acquired onboard EO-1, analysis searching for features of interest, e.g., thermal activity, is executed. The results of the analysis are later downlinked in an engineering telemetry stream, while onboard they may have generated a follow-up observation activity.

In order to support these requirements, we have adopted the open geospatial consortium sensor web enablement initiative to develop generalized service orient interfaces. These include the following:

1. The sensor planning service (SPS) used to determine if the sensor is available to acquire requested data. For example, using the SPS, an observation request to EO-1 can be issued to acquire science data, determine the status of an existing request and cancel a previous request.

2. The sensor observation service (SOS) used to retrieve engineering or science data. This includes access to historical data as well as data requested and acquired from the SPS.
3. The web processing service (WPS) used to perform a calculation on the acquired remote sensing data. This includes processing the raw data into derivative products such as vegetation indices, soil moisture, burn areas, lava flows and effusions rates, etc.
4. The sensor alert service (SAS) used to publish and subscribe to alerts from sensors. Users register with this service and provide conditions for alerts. When these conditions are met by the acquired data, alerts containing the data along with the time and location of the events are automatically issued to the user.
5. A description of the sensor and their associated products and services using the SensorML. SensorML provides a high level description of sensors and observation processes using an XML schema methodology. It also provides the functionality for users to discover instruments on the web along with services to task and acquire sensor data (such as the SPS, SOS, SAS and WPS).

10.4.1 Science Products

In conjunction with the standard instrument data collected, EO-1 has the capability of analyzing the data acquired onboard the spacecraft. Custom Hyperion data classification algorithms were uploaded to EO-1 in late 2003, which include thermal, snow/water/ice/land, flood, cloud and sulfur detection (cite each classifier author). In the context of OASIS, the thermal detection algorithm analyzes all acquisitions of Mount St. Helens. This algorithm is able to generate an onboard thermal summary products containing the radiance values of each hot pixel at 12 specific wavelengths. A hot spot is classified as an area of intense heat that is emitted from the volcano. Additionally, the hot and extreme pixels are another important product. Integrated hot spot energy, which is the summation of all of the hot pixels within the cluster, provides a useful summary data to the Earth scientists for evaluation.

Full data products nominally are downlinked 6–15 h after data acquisition, depending on when the ground contact was scheduled. However, a thermal summary product can be available with a couple of hours of the acquisition. This thermal summary product identifies both the line and the sample of thermally-active pixels; however, due to downlink constraints; its size is limited to 20 KB.

These summary products are used to generate alerts that are broadcast to interested entities, completing two-way communications with sensor peers. For OASIS, the alerts from the results of the thermal detection algorithm will automatically be ingested by the in situ command-and-control centre and autonomously trigger ground-based data collections.

10.5 Conclusion and Discussion

We have designed and are currently implementing and testing OASIS. Once our field tests are complete, we will deploy the sensor network on Mount St. Helens where it will run continuously for 1 year. Our deployment will be the first to integrate space and ground components into a continuous feedback loop, which will be constantly updating and optimizing the network's performance in order to obtain the maximum possible performance.

Acknowledgment This work is partially supported by NASA AIST Grant #106269, NSF-ITR grant IIS-0326505 and NSF-ITR grant IIS-0324835.

References

1. Pottie G, Kaiser W (2000) Wireless sensor networks. Commun ACM 43(5):51–58
2. Asada G, et al. (1998) Wireless integrated network sensors: low power systems on a chip, ESSCIRC
3. Eynde FO, et al. (2001) A fully-integrated single-chip SOC for Bluetooth, ISSCC 2001, vol. 446, pp. 196–197
4. Cerpa A, Elson J, Estrin D, Girod L, Hamilton M, and Zhao J (2001) Habitat monitoring: application driver for wireless communications technology. ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean
5. Mainwaring A, Polastre J, Szewczyk R, and Culler D (2002) Wireless sensor networks for habitat monitoring. ACM Workshop on Sensor Networks and Applications
6. Estrin D, Culler D, Pister K, and Sukhatme G (2002) Connecting the physical world with pervasive networks. IEEE Pervasive Comput 1(1):59–69
7. Fang Q, Zhao F, and Guibas L (2003) Lightweight sensing and communication protocols for target enumeration and aggregation. ACM MobiHoc, pp. 165–176
8. Werner-Allen G, Johnson J, Ruiz M, Lees J, and Welsh M (2005) Monitoring volcanic eruptions with a wireless sensor network. In Proc. Second European Workshop on Wireless Sensor Networks (EWSN'05)
9. Delin KA, Jackson SP, Johnson DW, Burleigh SC, Woodrow RR, McAuley JM, Dohm JM, Ip F, Ferre TPA, Rucker DF, and Baker VR (2005) Environmental studies with the sensor web: principals and practice. J Sens 5:103–117
10. Kok T, Seah WKG, and Wong WC (2006) An Energy Efficient Topology Management Scheme for Underwater Acoustic Sensor Network Using Connected Dominating Sets. OCEANS 2006 – Asia Pacific, pp. 1–7
11. Yarvis M, Ye W (2004) Tiered architectures in sensor networks. In Mohammed Ilyas (ed.), Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems. CRC Press, Boca Raton, FL, USA
12. Kottapalli V, Kiremidjian A, Lynch JP, Carryer E, Kenny T, Law K, and Lei Y (2003) A two-tier wireless sensor network architecture for structural health monitoring. In Proceedings of SPIE's 10th Annual Symposium on Smart Structures and Materials, San Diego, CA
13. Govindan R, Kohler E, Estrin D, Bian F, Chintalapudi K, Gnawali O, Rangwala S, Gummadi R, and Stathopoulos T (2005) Tenet: An Architecture for Tiered Embedded Networks. CENS Technical Report 56
14. Bluetooth SIG, Inc. http://www.bluetooth.org Accessed 28 May 2008
15. Rodoplu V, Meng TH (1999) Minimum energy mobile wireless networks. IEEE JSAC 17(8):1333–1344

16. Intanagonwiwat C, Govindan R, and Estrin D (2000) Directed diffusion: a scalable and robust communication paradigm for sensor networks. In Proceedings of the International Conference on Mobile Computing and Networking
17. Hwang I-S, Hwang B-J, Ku L-F, and Chang P-M (2008) Adaptive bandwidth management and reservation scheme in heterogeneous wireless networks. IEEE International Conference on Sensor Networks, Ubiquitous and Trustworthy Computing, SUTC '08, pp. 338–342
18. Zhu M, Reid A, Finney S, and Judd, M (2008) Energy scavenging technique for powering wireless sensors. International Conference on Condition Monitoring and Diagnosis, pp. 881–884
19. Woo A, Culler D (2001) A transmission control scheme for media access in sensor networks. In Proceedings of Seventh Annual International Conference on Mobile Computing and Networking (MobiCom). ACM, Rome, Italy, pp. 221–235
20. Ye W, Heidemann J, and Estrin D (2002) An energy-efficient mac protocol for wireless sensor networks. In IEEE INFOCOM
21. Heinzelman W, Chandrakasan A, and Balakrishnan H (2000) Energy-efficient routing protocols for wireless microsensor networks. In Proceedings of Hawaii International Conference on System Sciences
22. Sankarasubramaniam Y, Akan OB, and Akyildiz IF (2003) Esrt:event-to-sink reliable transport in wireless sensor networks. In The 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing(MobiHoc)
23. Clement B, Barrett A (2003) Continual coordination through shared activities. 2nd International Conference on Autonomous and Multi-Agent Systems (AAMAS 2003), Melbourne, Australia
24. Chien S, Cichy B, Davies A, Tran D, Rabideau G, Castano R, Sherwood R, Mandel D, Frye S, Shulman S, Jones J, and Grosvenor S (2005) An Autonomous Earth-Observing Sensorweb. Intelligent Systems, IEEE, May–June, 20(3):16–24
25. Song W-Z (2005) Real-time data gathering in wireless sensor networks. Technical Report 2005-S001. Washington State University, Vancouver
26. Alex H, Kumar M, and Shirazi B (2005) Collaborating agent communities for information fusion and decision making. International Conference on Knowledge Integration and Multi Agent Systems (KIMAS 05)
27. Murray TL, Ewert JW, Lockhart AB, and LaHusen RG (1996) The integrated mobile volcano-monitoring system used by the Volcano Disaster Assistance Program (VDAP). Monitoring and Mitigation of Volcano Hazards
28. Tai S, Benkoczi RR, Hassanein H, and Akl SG (2007) QoS and data relaying for wireless sensor networks. J Parallel Distribution Computing, pp. 715–726
29. Karp B, Kung HT (2000) GPSR: Greedy perimeter stateless routing for wireless networks. ACM Mobicom
30. Seada K, Helmy A (2005) Efficient and robust geocasting protocols for sensor networks. Elsevier Computer Commun J, Special Issue on Dependable Wireless Sensor Networks