FORMING LOGICAL NODE GROUPINGS IN SENSOR NETWORKS

Martin J Colley and Robert P Stacey
University of Essex, UK

ABSTRACT

Sensor network hardware has advanced to the point where cheap fully wireless nodes are available at commercially viable prices. Deployment of these nodes on any scale requires that they communicate with each other to aggregate and move data around the network. As many applications require the data to be combined from a particular set of nodes we propose a set of methods that make the logical groupings required by these applications possible. Our research focuses on the machine condition monitoring domain and how we can use the data generated by the sensor nodes to detect significant events in the time series data. We then describe a set of methods that can match these events and use the information to form logical groups based on the similarity of the events experienced. We then describe a simulation system that is being developed to investigate these methods along with the initial results and conclusions from our research so far.

INTRODUCTION

With current technology allowing the fabrication of cheap low powered truly wireless devices, new domains for sensor networks have been opened up. Micro sensor networks for habitat monitoring such as in Mainwaring et al. 2002 (1) and Cerpa et al. 2001 (2), networks for security Pottie and Kaiser 2000 (3), networks for machine condition monitoring, intelligent environments and indeed many other domains can now deploy far more sensor nodes increasing the amount and granularity of the data moving towards the level of ubiquitous computing mentioned in Weiser 1991 (4). With this increase in the number of nodes comes the increase in the complexity of the system and the need for automatic or self configuration for the ratio of nodes to users and the dynamics of these systems prohibit preset configurations as stated in Bulusu et al. 2001 (5).

The increase in the quantity of data has led to a variety of aggregation techniques, Heinzelman et al. 2000 (6), Estrin et al. 1999 (7), Heidemann et al. 2001 (8), Khan et al. (9), Krishnamachri et al. 2001 (10) and applications, Niculescu and Nath 2003 (11), Zhao et al. 2002 (12) that aim to reduce network load, save power and increase the accuracy of the data that is provided by the system. The aggregation techniques used by sensor networks mainly use node clustering techniques to group nodes around a cluster head that performs the information aggregation. The clusters formed in these techniques are arbitrary and selection of the cluster heads forms the main part of the algorithms, this selection can be based on randomness, energy levels, processing power or combinations of these factors. The remaining nodes then select a cluster head based on radio signal strength or some other arbitrary factor. For many domains this technique works well but some domains require that the data to be aggregated by a cluster should be from a particular set of nodes that have something in common.

The machine condition monitoring domain is an area that requires a different approach. This domain requires that the data to be aggregated comes from the same machine or component. By using techniques that employ arbitrary methods to form node clusters it cannot be guaranteed that the clusters formed will include all such nodes or even that the nodes will be based on the same machine. This limitation is also apparent in other domains, for example in habitat monitoring inside buildings, the data can only be aggregated within room boundaries otherwise the data may be meaningless.

Each node in a sensor network has the capability to sense its environment and so can use this information rather than radio signal to form node clusters. For different domains the sensors available will vary but by identifying distinct characteristics of the environment that can then be matched with other nodes meaningful clusters or logical groups can be formed. In this paper we focus on the machine condition monitoring domain and the sensors that will be available, namely vibration and temperature sensors.

Self Configuring Nodes

In systems where the number of nodes greatly outnumbers the users and where the location of deployment is not known beforehand, self configuration is not a luxury but a requirement. Sensor networks are an example of these systems as for an entire network of nodes there is typical only a few users if any. The dynamic nature of sensor networks also requires that the nodes are dynamically configurable to ensure that they
of the data must come from the particular machine or part of machine that is being monitored. If the data from a sensor mounted on another machine was used in the aggregation it could corrupt the data.

Current grouping or clustering techniques have no way of stopping this happening as they simply form arbitrary groups based on the selected cluster heads. This domain requires that logical groups are formed that relate to a machine of part thereof. Once these groups are formed cluster heads can be selected from within this logical set to communicate with the wider network and the standard techniques for moving data out of sensor networks can be used.

Micro sensors in the condition monitoring domain tend to be equipped with vibration/acceleration and temperature sensors that can record information about the machine they are situated on. This information about the machine must be collected and logged periodically to allow the data mining and predictive engineering applications and analysis to function. The other important aspect of the condition monitoring domain is that each machine will have alarm bands that readings should not move outside of. Should these limits be reached, alarms will be signalled from the system, whilst trying to avoid false alarms for problems such as sensor errors.

The next sections describe how events can be detected in the environment and the how these events can be compressed, matched and used to form logical clusters. We then move on to describe the simulation methods we have used and the experimentation that we have carried out so far. The paper ends with conclusions on the work we have carried out and a summary of the further work and aims of the research.

**DETECTING EVENTS IN THE ENVIRONMENT**

The environment of a node includes all the sensor readings to which it has local access, remote sensors located on other nodes could be situated in a different environment and so cannot be used for detecting local events.

The type of event we are attempting to detect can be described as an “interesting” sequence of readings that are somehow different from the normal or most recent values being read from a sensor. Figure 1 shows an example of such readings, the highlighted area of the graph shows the sequence of readings that we consider being an event. These events need not have a definite time scale as some sensors may not be able to sense high frequency changes in the environment and it would impair the accuracy of the system to constrain all sensors to the limits of the lowest frequency that can be measured in the system. The granularity of the events

---

1 Machine Condition Monitoring refers here more accurately to online condition monitoring where data is continuously obtained from the machinery being monitored rather than condition monitoring which can be a calendar based survey of machinery.

---

**Figure 1** – This figure shows real accelerometer data from an electric motor, the period highlighted shows the type of ‘interesting event’ that we are seeking. This part of the graph corresponds to the motor shutting down for a short period of time.

---

In Proceedings of the 2nd International Conference on Intelligent Environments, 5-6 July, Athens, Greece.
will also change dependant upon the application and the frequency of change inside the domain. For example the frequency of data and therefore events available from the temperature domain is lower than that of the vibration domain.

The events or “interesting” values we are using are periods of change in a sequence of readings. This will correspond to changes in the environment the sensor is in and hence the running of the machine it is situated on. These events will include start ups, run downs, speed changes, bump tests, moving through resonant frequencies or one of many other variations in the normal signal. The detection of these events requires that the node monitors its sensors more often than normal for a sensor network. However this will only need to be done during setup phase of the system.

Several researchers have used a variety of techniques to detect events in time series data as in Guralnik and Srivastava 1999 (15) and Li et al. 2002 (16). The techniques include attempting to detect changes in the data model and attempts to detect changes directly in the real data. As we need a method that is fast to execute, techniques that try to determine the data model are unsuitable as they involve significant computation. Methods that attempt to detect changes in the real signal can be used however as they generally use a simple detection model and can be used in an incremental or online fashion. In our research we have been using the simple amplitude change detection and this has allowed us to detect significant events in real time series data.

As events will occur in the system over a time period and are logged by the nodes, the data from the events will be compacted and stored in a specific format. This can then be transmitted to the other nodes nearby that are candidates for the same logical group. Individual nodes will therefore also receive events that have been detected by other nodes in the local area.

As the most common sensors available in the machine monitoring domain are vibration and temperature we have decided to focus on the vibration sensors for our initial research as they produce the most distinct reading transients and provide higher frequency data than temperature sensors.

**COMPRESSING EVENTS**

Once a node has observed an event the data sequence must be compressed so as to conserve memory space and message size and in doing so power. The event itself essentially contains a time sequence that has to be compressed into an expressive form that saves space and can easily compared to one another. Most compression algorithms either compress the data into an unreadable form that must be decompressed and/or are too computationally expensive for use in sensor networks, Roverso 2002 (17) provides techniques that may be suitable for ours needs but go beyond the capabilities of sensor network nodes by using wavelet transforms and transient matching.

The method used must allow for a direct comparison using the compressed format to allow the nodes to store events and compute the comparison results in good time. The speed of the compression is less important than the comparison speed as the node will only have to perform the compression on the local event but may have to make several comparisons with remote events in the same time period. As there is no requirement for a reverse transform in the data compression lossy techniques are acceptable for use.

As the data is a changing sequence of readings the technique used must preserve these changes and the sequence of these changes. The amplitude of the signal however, is not as important as different nodes may detect the same events with different levels of attenuation on the signal dependant on their location on the machine in question and hence will record at varying amplitudes. The compression algorithm therefore needs not ensure that the amplitude is preserved, although relative amplitudes will be important. The events recorded may vary in length and so the compression should preserve the idea of time within the event and standardise timing information, this storage format should also allow for partial signals to be matched.

In terms of the properties of the data compression required we have identified the main areas the algorithm and format should address;

- Changes in signal
- Relative Amplitudes (not absolute)
- Timing
- Partial Matching

The algorithm to be used is an area of current research and we are working on an algorithm and event format that will achieve these aims. Our system will initially use a mixture of signal averaging and information tagging where the set of compressed data is tagged with a timestamp and some statistical information about the data set giving an informative yet compact event format.

**MEASURING SIMILARITY OF EVENTS**

For nodes to work out the logical group they belong to, they must perform a similarity test or matching between events experienced locally and the remote events it receives. The test should give a result that specifies the likelihood that the events are the same. This measure will give a definite value to the confidence that two events are in fact the same event recorded from different perspectives.
The technique used should output a confidence factor that represents how similar the events are. The nodes may need to make many of these comparisons in a short space of time because if the environment is changing quickly remote nodes will be generating many events. As nodes need compare events from many different remote devices they should be able to compute the similarity of events quickly using a computationally inexpensive matching algorithm. Several algorithms exist that can form confidence measures for similarity but of these none are suitable as they are too complex and are unfeasible for implementation on sensor network nodes as they use clustering techniques or are not compatible with the format of events that we are using.

The tagged data format that we are using allows an algorithm to compare some features of the data set before even looking at the compressed data so there will be two stages to the matching algorithm.

**Matching Parameters**

An event will have many features that can be compared before the compressed measurements are looked at. These features or tags can be compared first as they can provide a fail fast method to avoid unnecessary calculation.

Timing is an important piece of information on an event as no matter how similar events seem, if they occurred at different times there is no possibility they can be the same event. This type of matching means that the nodes must have access to a clock that is kept relatively accurate. In the condition monitoring domain the existence of a clock can be assumed as each reading that is recorded must be time-stamped to ensure that the database of readings for a particular machine is accurate and can be used by data mining or predictive engineering applications. These same clocks can be used to timestamp events as they occur and hence give a first indication as to the similarity of events. If two events do not overlap in terms of their timing accounting for the allowed clock drift in the system then the two events cannot have occurred at the same time and therefore will have a similarity of zero.

Information on the event length and some statistics can also be provided if required and will form an initial similarity measure. If this value shows the events are not similar enough the next stage need not be completed. If this measure shows that the events may still be the same then the next phase of the algorithm should compare the compressed data values.

**Matching Compressed Readings**

As the data measurements recorded will be compressed before they are matched we will be trying to compare values that have already been normalised and so a direct comparison is possible. A simple method of event comparison is to subtract one event from another, if the results show an area where the readings fall to zero or near zero then it is likely that the events are the same in this area. Figure 2 shows an example of this matching where similar and dissimilar events are compared. The match will succeed where the resulting values are zero or thereabouts for the entire sequence. The sum of the resulting data normalised over length will provide the similarity of the events where the higher the value the less similar the events are.

Where the events are of differing lengths the algorithm must be modified to account for the fact that a match will no longer occur along the whole of the longer event. This can be solved by performing a series of matches using the smaller event and sections of the larger event. With the smallest resulting value being the

In Proceedings of the 2nd International Conference on Intelligent Environments, 5-6 July, Athens, Greece.
value returned as the similarity of the events. This will provide satisfactory results as if the events are of a significantly different length the fast fail mechanism will have already rejected the two events.

Once the events have been matched the nodes can begin to build up a record of which nodes are experiencing similar events.

FORMING LOGICAL CLUSTERS

As the nodes have started to collect information on the similarity of the events experienced by themselves and other nodes they can keep track of the likelihood that another node is in the same logical group. If a number of events can be matched then it is more likely that the two nodes are observing the same sequence of data and are in the same environment.

With information on the likelihood that other nodes are based in the same environment each node can form a personal view on which group they should be in. As there are no guarantees to the number of nodes that should be in each group or that the groups should be the same size or even the number of groups each node must decide on its group individually. The nodes however have a simple outlook on the groupings, either a node is in the same group or not.

Each node can use the information that it receives to decide upon the group it thinks it is in, as nodes are either in or out then this can be represented as a bit string.

As each node can decide on a set of other nodes that will form its group there must be a way of forming a consensus of opinion amongst the nodes. The solutions that the nodes decide on will in general be the same or very similar therefore a simple voting system should identify the most likely groups. Each node would vote towards the particular solution it had found and the solution with the most votes would be used. This speculation remains to be tested but seems a reasonable initial approach to the problem.

Choosing cluster heads and communicating with the wider network

Once local nodes have decided that they are in the same group they can decide upon a cluster head using a method based power levels, processing power, voting systems, random timers or any other information as any data that is generated from the cluster will now come from the same logical group of nodes. Any application that uses groupings can now use the clusters to perform its function.

To ensure that this logical grouping improves performance we intend to test an application with and without the clustering. To this end we have begun to build a simulation system that has been used to perform the initial experimentation into the methods proposed above.

SIMULATION SYSTEM

The simulation system is based in Java and uses a multi threaded model to enable both machines and nodes to be simulated simultaneously. This also allows the simulation to be distributed easily in the future. The simulation works by using a clock to increment the time in discrete units that are configurable on a per simulation basis. The nodes however are free running threads and so have asynchronous access to a machine and its sensor readings. The simulated machines use the time given by the clock to set the current parameters based on duty cycles that are specified in the simulation configuration.

Simulating Machines

Machines are simulated by the system by using a data set specified in the configuration that specifies the duty cycles of the machine. This data is currently simulated data but we are beginning to use real data in the simulations. Each machine is linked to the nodes that are placed on it and provides them all with the current values of vibration and temperature.

Simulated Sensing

The sensors form the interface between the machines and the nodes. Each sensor has models to add noise to the perfect readings obtained from the machines to ensure that two nodes will not obtain identical readings. The sensors are configured so that they can modify the sensor readings directly from the machine as they see fit, this enables the use of perfect sensors and the use of very noisy sensors in a simple plug and play manner. The sensors available to each node are specified in the simulation configuration file.

Node Simulation

The simulation models each node as an agent and the type of agent can be specified in the simulation configuration file. This enables the system to run as a heterogeneous agent system. The advantages of this are that agents with different sensor suites can be simulated in the same system, this is a more realistic scenario as
Situated nodes in the real world would carry different sensor suites dependant on where and on which type of machine they were positioned.

**Simulation Configuration**

The simulation configuration file allows all the parameters of the simulation to be specified as they loaded into the simulation. The simulation engine also supports configuration files with multiple simulations and will run them in sequence for performing large runs unattended. This method ensures that simulations can be repeated to confirm results for runs with identical parameters.

**Logging**

The simulation engine supports logging of a variety of types in xml. Events, messages and machine updates can all be logged in the current system with no limitations to what can be recorded in the future. The motivation behind xml is to support easy access to the information contained in the log and to ensure that all of the information is properly described. This method of logging allows many different visualization techniques to be used to view the information.

**Visualization**

To view the log file information for a simulation a variety of visualization techniques have been developed. Each of these has been designed to view different aspects of the simulation and allow the detection of different features. Currently only a single visualization has been completed and is shown in Figure 3 but as the data is formatted with xml there are many more that could be implemented.

**SIMULATED EXPERIMENTS**

To assess the viability of the methods proposed for determining clusters it must first be confirmed that nodes can indeed detect the “interesting” events that occur in the environment. To this end we have developed the simulation we have described above that models the duty cycles of machines and provides the nodes with noisy data on the machines vibration and temperature. Using this information the simulated nodes were configured to try and detect the events given the readings they have access to and using the methods we have described. Each simulated node is placed on a virtual machine inside the simulation and can only receive information about the machine it is linked to.

As simulation logs the events that the each of the nodes produces, if the nodes are capable of detecting similar events they will log events at the same time. Figure 3 shows a visualisation of these log files. It shows when each node created a log entry for an event that has been detected. As can be seen the nodes that are based on the machine produce events at approximately the same time. This shows that the nodes are capable of detecting the events in the simulated environment.

**FURTHER WORK**

This research is in the early stages of experimentation which has confirmed the detection of events. Further work is needed to determine the minimum amounts of information that is needed to match events and then to allow nodes to form personal cluster solutions. The work can then be extended to allow the nodes to form a consensus on these groupings and enable higher level functionality to draw on data collected from a logical group of nodes.

These techniques will not only be useful in the condition monitoring domain but in other areas where the information collected from a sensor network requires that the groupings are some logical entity such as a machine, room, field or compartment. This type of configuration would be useable in intelligent environments and mobile computing as mobile devices would be able to query local situated nodes and compare recent events enabling them to discover their location. It would be beneficial in habitat monitoring applications where data could be collected on specific areas and the technique could be used in agriculture for determining the grouping of nodes distributed across the farm. These techniques would allow the applications to draw on data that had been collected from specific

Figure 3 – This figure is the initial visualisation of the simulation data showing the occurrence of events inside the simulation. As can be seen each node produces a series of events from the data obtained.

In Proceedings of the 2nd International Conference on Intelligent Environments, 5-6 July, Athens, Greece.
groups of nodes that had not been configured at compile
time and that need no additional configuration once
deployed in the field.

Real World Realisation

To ensure that the techniques we develop function in a
real world situated environment we plan to develop an
experimental test bed of nodes on which we will
implement the ideas presented here. This test bed will
allow us to validate the results of the simulation and
show that the techniques are applicable to a variety of
domains.

CONCLUSIONS

The research carried out so far indicates that the
proposals we have made can logically group nodes and
are feasible and realistic. We have shown that events
can be detected inside the environment and we have
presented methods to compare these events with each
other. The next steps in our research have been outlined
above and we believe that these directions will make
sensor networking techniques more applicable to
domains where the composition of node groupings is
important such as condition monitoring, habitat
monitoring and intelligent environments.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the
contributions of Global Maintenance Technologies Ltd,
Chelmsford, Essex, UK and the Engineering and
Physical Sciences Research Council (EPSRC) for their
support in this project.

REFERENCES

Networks for Habitat Monitoring. ACM International
Workshop on Wireless Sensor Networks and
Applications, Atlanta, GA.
Technology. Proceedings of the 2001 ACM SIGCOMM
Workshop on Data Communications in Latin America
and the Caribbean.
Integrated Network Sensors." Communications of the
ACM 43(5).
Century. Scientific American.
Coordination for Wireless Sensor Networks: Self-
Configuring Localization Systems. In Proceedings of
the Sixth International Symposium on Communication
Theory and Applications Ambleside, Lake District, UK.
Efficient Communication Protocol for Wireless
Microsensor Networks. 33rd International Conference
on System Sciences, Hawaii.
Challenges: Scaleable Coordination in Sensor
Wireless Sensor Networks with Low-level Naming. In
Proceedings of the Symposium on Operating Systems
Principles, Alberta, Canada.
Clusters, Data Aggregation, and Security in
Microsensor Networks.
Data Aggregation in Wireless Sensor Networks.
Technical Report 01-750, University of Southern
California.
Positioning System (APS) Using AOA. In Proceedings
of INFOCOM, San Francisco, CA.
Scan for Monitoring Sensor Networks. IEEE Wireless
Communications and Networking Conference, Orlando,
FL.
Unattended Ground Sensor Technologies and
Applications, Orlando, FL.
Topologies. IEEE INFOCOM.
detection from time series data. Proceedings of the fifth
ACM SIGKDD, San Diego, California.
16. Li, D., et al. (2002). Detection, Classification
and Tracking of Targets in Distributed Sensor
Networks. IEEE Signal Processing Magazine 19(2): 17-
30.
Transient Classification: The ALADDIN Approach.
IJIS(Special Issue on Intelligent Systems for Plant
Surveillance and Diagnostics).