INVITED PAPER

Scalability and Performance Issues in Deeply Embedded Sensor Systems

Prasanna Sridhar

Department of Electrical and Computer Engineering

University of New Mexico

Albuquerque, NM, USA

Email: prassi_hs@yahoo.com

Asad M. Madni

President and Chief Operating Officer (Retired), BEI Technologies, Inc.

Executive Managing Director and Chief Technical Officer, Crocker Capital

Los Angeles, CA, USA

Email: ammadni@yahoo.com

Abstract- The property of scalability for a given system indicates the ability of a system or a subsystem to be modified with changing load on the system. For a sufficiently large complex system, there are several factors that influence the ability of the system to scale. It is necessary to incorporate solutions to these factors (or bottlenecks) in the design for scalability of a given system. In this paper, we discuss such design principles to handle the key factors that influence the scalability of large complex systems. Specifically, we demonstrate design and implementation of simple, innovative, and relatively less expensive methodology to guarantee that a large complex system (such as network of sensors) is scalable under varying load conditions.

Index terms*:* Wireless sensor networks, scalability and performance, sensor calibration, data summarization and aggregation.

I. INTRODUCTION

A. Scalability

A system is said to be scalable if its availability and fault tolerance are within the acceptable thresholds when the load or subsystems is increased in the system [1-3]. Both parameters along

with the performance of the system signify the usability of the system. A system that scales well does not necessarily mean that it performs well. Ideally, if a system is scalable, then its performance increases (or remains same), and its availability increases. Performance of a system is based on its output behavior. Availability signifies that there is no single point of failure in the system. The system shows graceful degradation with an increase in fault tolerance.

A system can be,

- Geographically scalable – usability is still good no matter how geographically far the system is located from the users. However, usability depends on the system's sensitivity, range, power, and other parameters.
- Load scalable – usability is still good even with the increase in load on the system.

An important aspect of scalability may be best described in applications that require several complex large-scale systems to work in synergy in order to accomplish a common goal. Examples of such systems include sensor systems often seen in military and defense applications, to provide increased situational awareness (as shown in figure 1).The combination of two or more systems is entirely necessitated by the application under consideration. Therefore, a single system would not have accomplished the task efficiently as compared to multiple scalable systems working in tandem.

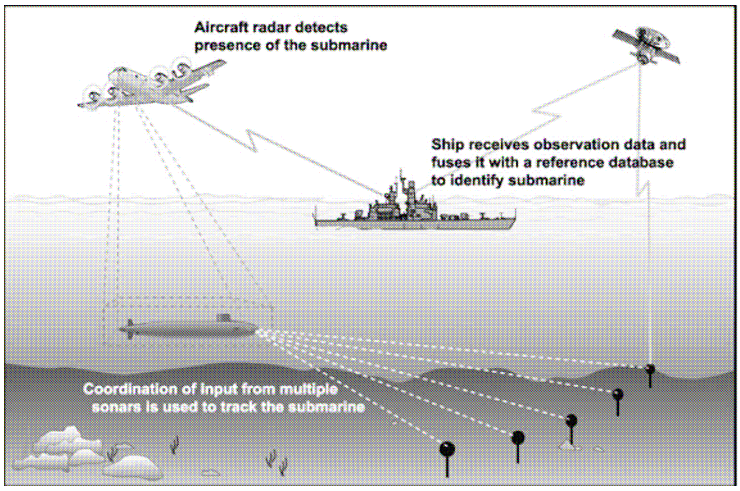

Figure 1. Sensory Systems Working in Tandem to Provide Increased Situational Awareness (Image courtesy of [4])

B. Scalability in Large Complex Systems

The importance of scalability as a design factor is due to the fact that such systems should ideally be able to add more systems or sub-systems without having to re-engineer the existing architecture. Consider for example an airport. Adding a redundant system (for example a new communication system) should ideally not degrade the performance, availability and usability of the existing system (airport). Generally, System Engineering (SE) [5] should provide such a systematic engineering methodology to deliver scalable systems and should incorporate the necessary means to guarantee scalability in the design principles.

II. SCALABILITY ISSUES IN SENSOR NETWORKS

Consider network of embedded sensors as yet another example of a system that is complex and is composed of heterogeneous, independent subsystems (or systems). This complex system is composed of sensor nodes each of which have multiple sensors on-board along with limited processing and storage capabilities. Therefore, they are sufficiently complex and operationally independent in nature. The hardware characteristics provided by these sensor nodes make it feasible to load software (and operating system) on these nodes. These software tools along with an operating system make the nodes managerially independent. The type of sensors on-board a given node determines if the node is functionally different from other nodes. Within a given sensor network, the sensor nodes can have different processors (ARM, Intel XScale, etc), different RF middleware technologies (ZigBee, 802.11, etc) and even different software layers on top of hardware. These characteristics make them architecturally different.

Having multiple sensor nodes clearly demonstrates the advantage of increased situational awareness. For example, multiple sensors deployed in a vast geographical area can provide better coverage than a single sensor. An increase in sensor node deployment will increase the coverage of the area (given an idealistic situation where nodes do not overlap). The availability and interval of failure of the entire network also increases due to increase in node density (as some nodes can act as redundant back-up nodes). This method of scaling is sometimes referred to as **scale-out**, where more load (nodes) is added to a given system [6]. In **scale-up**, one can add more resources (memory, processor power, low power usage etc.) to a single sensor node so as to increase its coverage which in turn can affect the scalability of the entire network. However, it is

important to note whether the *performance* of such a system would increase with the increase in sensor nodes.

There are two important metrics that can describe the performance of a given system – **throughput** and **latency** [7]. Throughput defines the amount of work processed by the system in a given unit time. The amount of time it takes to complete one given task is defined as latency. Ideally, a given system is said to be efficient in terms of performance if it has high throughput and low latency. In our previous example, increasing the sensor nodes (to achieve increased situational awareness) could decrease the performance of the overall system. Consider the task of delivering the sensed data from the sensor nodes to a processing node (such as base-station) and processing the sensed data to obtain a decision milestone. With an increase in sensor nodes, there is enormous amount of data being delivered to the central base-station. This results in network congestion which in turn induces higher latency before the data is actually delivered and processed. The throughput also suffers as a result of high utilization of the network and high processor usage at the base-station.

To this end, we will discuss in detail, two important features relating to performance in sensor networks. Specifically, we will focus on summarization of data in order to achieve cost effective means to optimize data transmission. The presented approach also proposes the use of spatially variant weights to reduce the significance of sensor readings taken near the boundary of the sensor range, in order to minimize potential corruption of summarized data. Although data summarization generally incurs delay, the proposed approach is a cost effective method (i.e., low computation utilization) resulting in decreased delay latency caused due to communication. Before summarization, the data in each sensor node can be locally calibrated to further enhance the correctness of data. This will also ensure one important factor of scalable design – tolerance to failure. We will discuss one such localized calibration technique in our next section.

A. Localized Calibration

The principle idea behind fault tolerance is the system's ability to perform or operate correctly even in the presence of faults. Fault-tolerant calibration scheme when performed at a local sub-system level (sensor node level) can have tremendous impact on a global system level (sensor network level). The problem of fault identification and isolation is generally a complex, non-trivial task in sensor networks due to the very nature of their construction and deployment.

Moreover, due to the low computation and communication capabilities of the sensor nodes, the fault-tolerant mechanism should have a very low computation overhead. There are several sensor data validation mechanism proposed in literature [8-10]. Most of these are limit checking mechanisms, in which, sensor reading is compared against pre-establish nominal window. If the reading is outside this window, the sensor is deemed faulty. Mechanisms such as Kalman filtering, particle filtering, and wavelet transforms [11, 12] for sensor data validation are not suitable simply because they are computationally intensive. A simple mechanism to detect and isolate faults in sensors is to use range tests. A sensor is deemed to be faulty if its reading exceeds the threshold limits (minimum and maximum values specified). Such tests are commonly used to capture "hard faults" that occur rapidly and are termed as Built-In Test (BIT) [13, 14].

The design of threshold limits for a given sensor should be meaningful. If the limits (min and max values) are tight, we obtain high false alarms. If the limits are relaxed, then we will capture very few faults. In our approach to fault detection, we assume that each sensor within a node is assumed to operate within a ***usable threshold window***. A Built-in Test (BIT) is said to have passed if the sensor reading is within this window. Every sensor is guaranteed to work "correctly" within a given operating range specified by the manufacturer. We call this operating range of sensor as ***guaranteed window***. This window is usually obtained from the sensor manufacturer. The usable threshold window is defined outside guaranteed window such that it will incorporate the guaranteed window for a given sensor. That is, lower threshold boundary of the usable threshold window will be lesser than the minimum range of operation defined by the guaranteed window and higher threshold boundary will be greater than the maximum operating range of the sensor. The sensor might still work outside this guaranteed window, however, with a much lesser accuracy. With limit checking or limit filtering, a sensor is deemed faulty when its reading is outside such guaranteed window. By using the concept of added usable window, however, we can capture intermittent sensor faults or sensor drifts that are seen only for a brief period of time. A simple methodology is to use a *bell-function* that decreases the weight exponentially as the reading deviates from the guaranteed window. At each time instance, the reading of the sensor (if outside the guaranteed window but within the usable window) can be weighted based on the weights obtained from the function:

$$w_{ij} = e^{-(r_{ij}/\varepsilon)^2}$$
$$r_{ij} = w_{ij} \times r_{ij} \quad (1)$$

where, r is the sensor reading, w is weighting factor, and ε is chosen appropriately such that 0<w<1.A detailed algorithm with limitations for this approach is discussed in [15]. A simple validation function or curve as described in (1) is dynamic in nature with the operating conditions of the sensor (since the weighting factor is influenced by the sensor reading). This means that the sensor reading gets discounted every time it moves away from the guaranteed towards usable threshold window.

Figure 2 shows the benchmarking result on aggregation of simulated sensor data from three temperature sensors. The guaranteed window is set to [+20°C, +120°C]. Reading from sensor-2 gradually approaches the usable window, thereby suggesting that there is high probability of failure. Our proposed windowing approach incorporates the degradation of sensor-2 and failure of sensor-1 during the aggregation process, and thus influences the effect of aggregation.

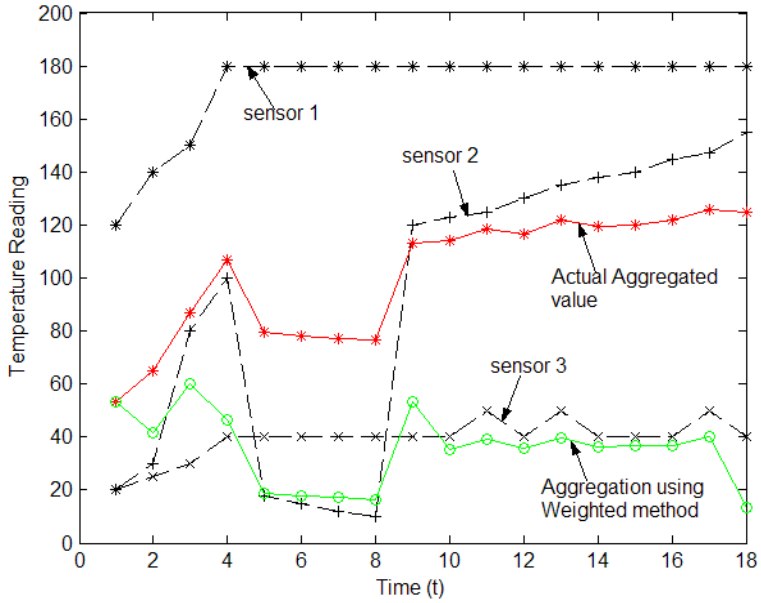

Figure 2. Windowing Effects on Sensor Data Aggregation

Threshold tests prove useful in detecting hard faults. By nature "soft faults" occur when sensors are working within the threshold limits but might still be deemed faulty. For example, when the sensor reading is outside the manufacturer's guaranteed window, it is generally difficult to classify whether the reading is an environmental stimuli captured by the sensor or if it's an

intermittent transient (short-lived oscillation due to sudden voltage or current change) or if it's an intermittent fault (soft fault). In order to detect such faults, we will take advantage of the nature of applications in which the sensor nodes are often deployed. These applications often require dense deployment of sensor nodes, thereby, making the sensor nodes ***spatially correlate*** with neighboring nodes within the same event region. This means that a given sensor would read the same event value (with minimal variations) as neighboring *k* sensors which are closely deployed. For example, consider three sensor readings a,b, and c from three redundant sensors. Two sensor readings are averaged ((a+b)/2, (a+c)/2, and (b+c)/2) at each predetermined time interval. The actual reading is set to the value at which the majority of the three averaged values agree upon – a *plurality voting principle* [16]. This helps to eliminate the faulty sensor in the group of the three sensors under consideration.

B. Data Summarization

A more comprehensive mechanism based on the plurality voting principle is to decrease the contribution of the faulty sensor and increase the contribution of the non-faulty sensors. This can be achieved by simple *weighting factor.*

Each sensor node has a weighting factor at any instance of time *t,* given by $w_i(t)$. In the event of sensor degradation, the proposed algorithm adaptively decreases the weight for sensors which demonstrate likelihood to fail. At the same time, weighting factors for the neighboring sensor nodes is increased. Hence, every reading from each sensor is weighted at each predetermined time interval *t*, and weight updates are computed as follows:

$$w_i(t+1) = w_i(t) \pm \Delta w_i(t) \qquad (2)$$

A coordinator in a large-scale complex system framework (a cluster-head in sensor network) can simply query the nodes for sensor reading in the event region. Based on a specific timeout, the fusion node performs weighted average aggregation based on the data it has received currently from the sensors.

In traditional neural networks, the change in weights $\Delta w_i(t)$ is a function of the error estimate [17], which is based on the difference between the expected reading and the actual reading. However, in sensor nodes, we do not know the expected or desired reading *a priori*. In order to

estimate $\Delta w_i(t)$, we use the concept of spatial correlation as explained earlier. In order to estimate $\Delta w_i(t)$, we propose the following model:

$$\Delta w_i(t) = |\tau_i| * \varepsilon \tag{3}$$

where, τ, the ***adaptation parameter*** is given by,

$$\tau_i = \frac{r_1 + r_2 \ldots r_{i-1} + r_{i+1} \ldots r_{k+1}}{k} - r_i \tag{4}$$

r_i is the reading from the i-th sensor, k is the number of neighboring sensors and ε, the ***scaling factor,*** is a small value $0 < \varepsilon < 1$ and is chosen appropriately for a given application. The scaling factor ensures that $0 < \Delta w_i(t) < 1$. A complete implementation can be found in [18].

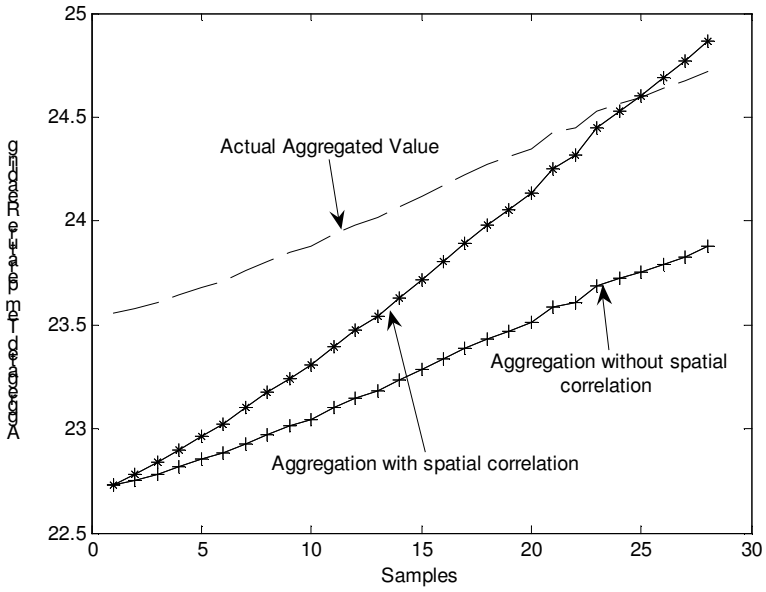

Figure 3. Comparison of Aggregation under Faulty and Normal Conditions

In order to validate our algorithm, we aggregated the data based on the obtained weighting factor. Figure 3 shows the comparison of data aggregation with and without spatial correlation. As the aggregated value approaches ground truth (actual aggregated value), the error in the algorithm performance decreases and eventually becomes zero (see figure 4).

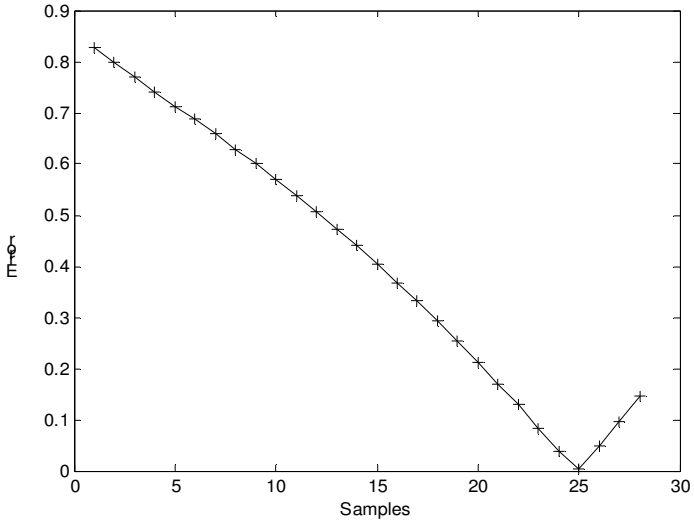

Figure 4. Error between Ground Truth and Proposed Approach

We also compared the weighted average data with a simple average of three sensory data with and without fault correction as shown in figure 5. Detailed real-world experimentation can be found in [19].

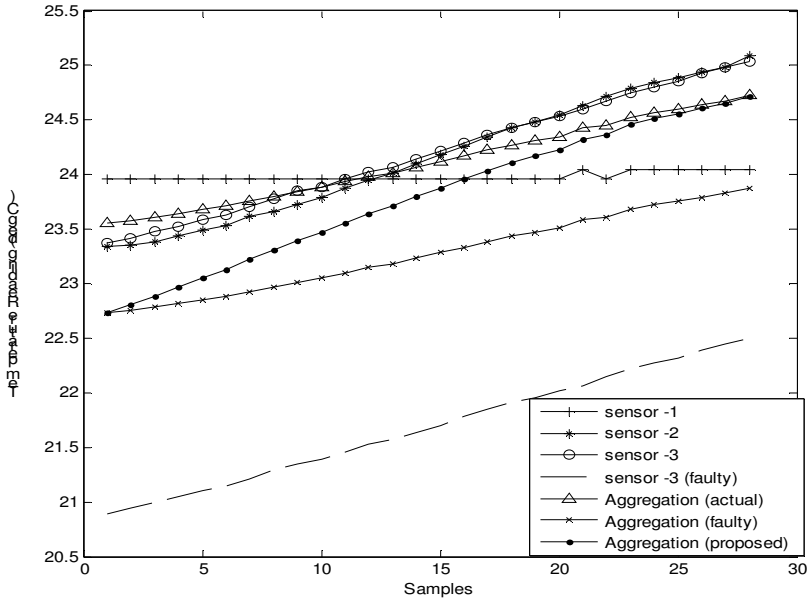

Figure 5. Aggregation with and without Fault Correction

By having such a fault-tolerant data aggregation, we have addressed the key factors that influence the scalability in large scale systems. The availability of the system (network) as whole is also guaranteed in general, since the deployment of sensor nodes is large to get better coverage.

Consolidating data by using time-varying weight adaptation method helps to reduce the data propagated from the sensor field to the base-station thereby having minimal effect on the utilization of the network.

B. Centralized Controllability and Experimental Verification

Extensive computation load should be handled by base-station which is assumed to have higher computation, power and storage capabilities compared to sensor nodes. This is our centralized implementation. Critical, faster, and less expensive computations (such as early fault detection, data summarization, etc.) should be handled at a node or cluster-head level, which is distributed in the area of interest. At the base-station level, in order to visualize data from different sensors as well as to propagate decision from base-station to all the nodes it is necessary to have a user-level visualization tool. Such visualization tool should also enable the user to control or pass messages (to control) to the deployed sensors. This forms a ***sensing-decision-actuation*** loop. We developed the visualization interface tool in National Instruments' LabView®. This interface allows the user to observe aggregated data and take decisions based on the observed data – such as enabling built-in test, disable aggregation for critical data, etc. A snap-shot of the interface developed is shown in figure 6.

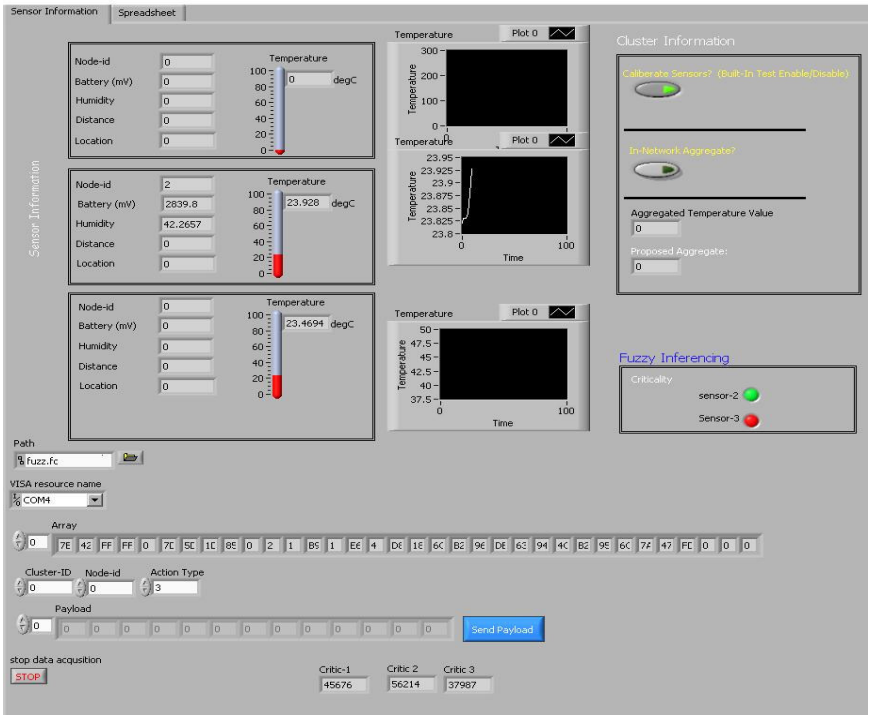

Figure 6. User Interface for Sensor Calibration and Data Summarization

In order for the user to control the parameters on a remote sensor through the interface it is necessary to devise a protocol (and a message format). The message or packet format will be understood by the remotely deployed sensor and takes specific action based on the *action type* in the message. The message structure is as shown figure 7.

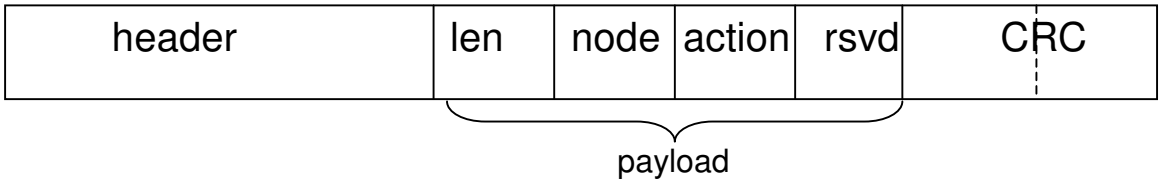

Figure 7. Custom Message Structure for Over the Air Programming

The description of fields (8-bit) in the message structure is as follows:

len: Length of the payload

node: node-id to send the message

action: action to be performed on the node

action type: 01 – Aggregate data
02 – Disable aggregation
03 – Enable Built-in Test
04 – Disable Built-in Test
05 – High Sleep Time
06 – Low Sleep Time
07 – Reset Sleep Time
08 – High Transmission Power (adjust potentiometer)
09 – Reset Transmission Power

rsvd: reserved field for future use.

For the remotely deployed sensor nodes to understand the commands from the interface, nodes need to decipher the packet (as shown in figure 7). The action selection at the sensor nodes is based on the action type in the message. Therefore, a middleware or service layer software component is embedded in these sensor nodes that can decipher the message [20].

III. CONCOLUDING REMARKS

Scalability is one of the important concepts necessary in more effectively implementing and analyzing large, complex, independent, heterogeneous and autonomous systems working cooperatively. When systems interact (and often cooperate and coordinate) with each other to address the defined high level objectives, the performance of the entire combination of such systems could degrade due to optimization, autonomy, etc. We can summarize our scalable design – a multi-tier architecture for efficient deployment of sensor nodes as shown in figure 8.

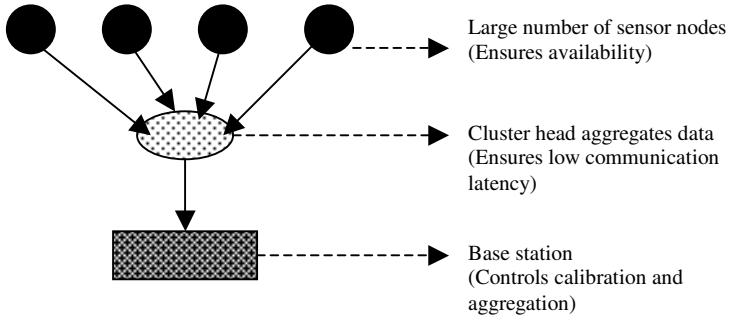

Figure 8. Multi-Tier Scalable Design

We have addressed the scalability issue in sensor networks as a case study by proposing an innovative solution to fault-tolerance and policy/decision making. The solution provided is generic in nature and can be extended to any complex systems applications. The coordinator takes critical managerial operations (for performance improvement) whereas each system takes local hard-real time operations (for fault isolation in this case study). Other factors that could also influence the performance of large-scale systems such as optimization can also be carried out at a coordinator level. For example, [21] addresses and formulates an optimization problem when system of sensors works cooperatively with system of robots with varying degree of autonomy for a specific application. Such analysis and modeling can be generalized to any complex heterogeneous system application.

REFERENCES

[1] Lee, L. C., Nwana, H.S., Ndumu, D.T., De Wilde, P., "The Stability, Scalability and Performance of Multi-agent Systems", BT Technology Journal, Vol. 16., pp. 94-103, Kluwer Acadamic Publishers, 1998

[2] Woodside C M, and Schramm C, "Scalability and performance experiments using synthetic distributed server systems", Distributed Systems Engineering, 3, Nos 2-8, 1996.

[3] Bondi, A., "Characteristics of scalability and their impact on performance", Proc. of the 2nd international workshop on Software and performance, pp. 195 - 203, 2000

[4] Hall, D., Llinas, J., "Handbook of Multisensor Data Fusion", CRC Press, 2001

[5] Sage, A., "Systems Engineering", Wiley IEEE Publishers, 1992

[6] Michael, M., Moreira, J.E., Shiloach, D., Wisniewski, R.W., "Scale-up x Scale-out: A Case Study using Nutch/Lucene", Proc. of the IEEE Parallel and Distributed Processing, pp. 1-8, 2007

[7] Friedman, R., Hadad, E., "Analyzing distributed-system performance - latency vs. throughput", IEEE Distributed Systems Online, Vol. 7, Issue 1, 2006

[8] F. Koushanfar, M. Potkonjak, A. Vincentelli, "Fault Tolerance Techniques for Wireless Ad hoc Sensor Networks", IEEE Sensors Journal, Vol 2., pp. 1491- 1496, 2002

[9] Elnahrawy, E., Nath, B., "Cleaning and Querying Noisy Sensors", Proc. of the Workshop on wireless sensor networks and applications, pp. 78 – 87, 2003

[10] Hereford, J., "Fault-Tolerant Sensor Systems Using Evolvable Hardware", IEEE Transactions on Instrumentation and Measurement, Vol. 55, No. 3, pp. 846-853, June 2006

[11] Wei, T., Huang, Y., Chen, P., "Particle filtering for adaptive sensor fault detection and identification", Proc. of the 2006 IEEE Robotics and Automation (ICRA 2006)
Volume , Issue , 15-19, pp. 3807 – 3812, May 2006

[12] Zhang, J.Q., Yan, Y., "Online validation of the measurement uncertainty of a sensor usingwavelet transforms", IEE Proc. of Science, Measurement and Technology, Volume 148, Issue 5, pp. 210 – 214, Sep 2001

[13] Madni, Asad M., Costlow, Lynn E., "Common Design Techniques for BEI GyroChip® Quartz Rate Sensors for both Automotive and Aerospace/Defense Markets", IEEE Transactions on Sensors Journal, Vol 3 No. 5, pp. 569-578, October 2003

[14] Chen, L., Dey, S., Sanchez, P., Sekar, K., Chen, Y., "Embedded Hardware and Software Self-Testing Methodologies for Processor Cores", 37th Design Automation Conference, pp. 625 - 630, 2000

[15] Madni, Asad M., Sridhar, Prasanna, Jamshidi, Mo, "Fault-Tolerant Data Acquisition in Sensor Networks", Proc. of the IEEE International Conference on System of Systems Engineering, San Antonio, 2007

[16] G. Sigletos, G. Paliouras, and C.D. Spyropoulos, "Combining information extraction systems using voting and stacked generalization," Journal of Machine Learning Research, Vol. 6, pp.1751-1782, 2005.

[17] Haykin, S., "Neural Networks: A Comprehensive Foundation", 2nd Edition, Prentice Hall, 1998.

[18] Sridhar, P., Madni, A. M., Jamshidi, M., "Hierarchical Data Aggregation in Spatially Correlated Distributed Sensor Network", Proc. of the World Automation Congress, 2006

[19] Sridhar, P., Madni, A. M., Jamshidi, M., "Hierarchical Aggregation and Intelligent Monitoring and Control in Fault-Tolerant Wireless Sensor Networks", IEEE Systems Journal (Inaugural Issue), Vol.1, No.1, pp.38-54, September 2007.

[20] Sridhar, P., "Hierarchical Aggregation and Intelligent Monitoring and Control in Fault-Tolerant Wireless Sensor Networks", Ph.D. Dissertation, Univ. of New Mexico, 2007.

[21] Azarnoush, H., Horan, B., Sridhar P., Madni, A M., Jamshidi, M., "Towards Optimization of a Real-World Robotic-Sensor System of Systems", Proc of the World Automation Congress, Budapest, Hungary, 2006