PERGAMON Journal of the Franklin Institute 338 (2001) 655–668

Journal *of* The Franklin Institute

www.elsevier.com/locate/jfranklin

Distributed sensor networks—a review of recent research ☆

Hairong Qi[a,*], S. Sitharama Iyengar[b], Krishnendu Chakrabarty[c]

[a] *Department of Electrical and Computer Engineering, The University of Tennessee, 319 Ferris Hall, Knoxville, TN 37996-2100, USA*

[b] *Department of Computer Science, 298 Coates Hall, Louisiana State University, Baton Rouge, LA 70803, USA*

[c] *Department of Electrical and Computer Engineering, Duke University, 130 Hudson Hall, Box 90291, Durham, NC 27708, USA*

Received 1 May 2001

Abstract

Advances in sensor technology and computer networks have enabled distributed sensor networks (DSNs) to evolve from small clusters of large sensors to large swarms of micro-sensors, from fixed sensor nodes to mobile nodes, from wired communications to wireless communications, from static network topology to dynamically changing topology. However, these technological advances have also brought new challenges to processing large amount of data in a bandwidth-limited, power-constraint, unstable and dynamic environment. This paper reviews recent developments in DSNs from four aspects: network structure, data processing paradigm, sensor fusion algorithm with emphasis on fault-tolerant algorithm design, and optimal sensor deployment strategy.

Keywords: Distributed sensor networks; Data processing paradigm; Sensor fusion; Fault tolerant algorithm

1. Introduction

Distributed sensor networks (DSNs) have recently emerged as an important research area [1–5]. This development has been spurred by advances in sensor

☆ This research was supported in part by DARPA under Grant N66001-001-8946.

*Corresponding author. Tel.: +1-865-974-8527; fax: +1-865-974-5483.

E-mail address: hqi@utk.edu (H. Qi).

0016-0032/01/$20.00
PII: S0016-0032(01)00026-6

technology and computer networking. International Society of Information Fusion (ISIF) maintains a comprehensive list to related publications [6], including books, edited books, journals, conference/workshop/symposium proceedings, special issues and sections. Even though it is economically feasible to implement DSNs, there are several technical challenges that must be overcome before DSNs can be used for today's increasingly complex information gathering tasks. These tasks, across a wide spectrum of both civilian and military applications, include environment monitoring, scene reconstruction, motion tracking, motion detection, battlefield surveillance, remote sensing, global awareness, etc. They are usually time-critical, cover a large geographical area, and require reliable delivery of accurate information for their completion.

Fig. 1 is a block diagram, illustrating different components in a DSN from functionality point of view. The ultimate goal of DSNs is to make decisions or gain knowledge based on the information fused from distributed sensor inputs. At the lowest level, individual sensor node collects data from different sensing modalities on-board. An initial data processing can be carried out at the local node to generate local event detection result. These intermediate results will then be integrated/fused at an upper processing center to derive knowledge and help making decisions.

Research issues associated with this diagram can be summarized into three questions: where to fuse? what to fuse? and how to fuse? With the size of sensors getting smaller and the price getting cheaper, more sensors can be developed to achieve quality through quantity. On the other hand, sensors typically communicate through wireless networks where the network bandwidth is much lower than for wired communication. These issues bring new challenges to the design of DSNs: First, data volumes being integrated are much larger; Second, the communication bandwidth for wireless network is much lower; Third, the power resource on each sensor is quite limited; Fourth, the environment is more unreliable, causing unreliable network connection and increasing the likelihood of input data to be in faulty.

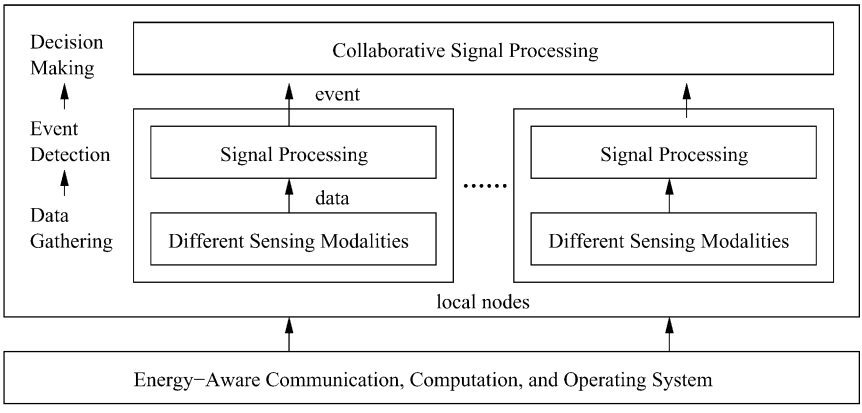


Fig. 1. Block diagram of a DSN from functionality point of view.

There have been a few review papers published since late 80s [7–11], summarizing research in DSNs from different aspects. This paper tries to present recent developments in DSNs from four aspects: the network structure (Section 2), the data processing paradigm (Section 3), the fault-tolerant sensor fusion algorithm design with an emphasis on fault-tolerance integration (Section 4), and the optimal sensor deployment strategy (Section 5).

2. Network structure

DSN research in this aspect started in the early 80s. Wesson et al. [5] were among the first to propose network structures that can be used to design a DSN. Later, Iyengar et al. [1] made some important improvement to the original design.

We first define a general DSN structure and explain the terminologies used in the structure. A general DSN (Fig. 2) consists of a set of *sensor nodes, a set of processing elements* (PEs), and a *communication network* interconnecting the various PEs [1]. One or more sensors is associated with each PE. One sensor can report to more than one PE. A PE and its associated sensors are referred to as a *cluster*. Data are transferred from sensors to their associated PE(s), where the data integration takes place. PEs can also coordinate with each other to achieve a better estimation of the environment and report to higher level PEs.

2.1. Previous work

Two structures were analyzed in the initial work of Wesson et al. [5]: the anarchic committee (AC) structure and the dynamic hierarchical cone (DHC) structure as illustrated in Fig. 3. AC can be viewed as a fully interconnected network without hierarchy, where each node can communicate with any other node, thus coordination between nodes is straightforward. Although easy for communication, AC structure is expensive to implement and also hard to extend. On the other hand, DHC provides a hierarchical structure, also called a tree structure. It only allows communications between nodes in adjacent layers, but not within the same layer.

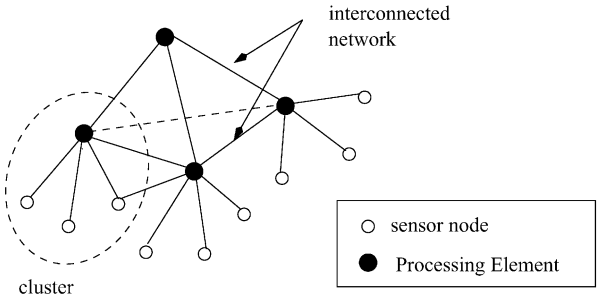


Fig. 2. The architecture of a general DSN.

Compared to AC, DHC is easier to extend, but is more vulnerable since a faulty node can disconnect an entire subtree.

In order to overcome the drawbacks in both AC and DHC, a hybrid structure which is called flat tree network was proposed in [2,4]. The nodes in this network are organized as many complete binary trees, and the roots of which are completely connected, as illustrated in Fig. 4. Even though flat tree structure improves DSN from both hierarchical and robustness aspects, integration errors of the lower nodes will be accumulated as the information goes up the hierarchy. One way to overcome this problem is to interconnect nodes in the lower levels of this network. Iyengar et al. [1] proposed to use deBruijn graph (DG) [12] to connect nodes at each level as shown in Fig. 4. DG provides several advantages over AC, DHC, and flat tree structures, such as simple routing schemes, better fault tolerant capabilities, better extensibility (the diameter of the network grows only logarithmically with the number of nodes).

2.2. Ad hoc wireless sensor networks

Advances in sensor technology and wireless communication have made ad hoc wireless sensor networks (AWSNs) a reality. Unlike traditional wired networks, the

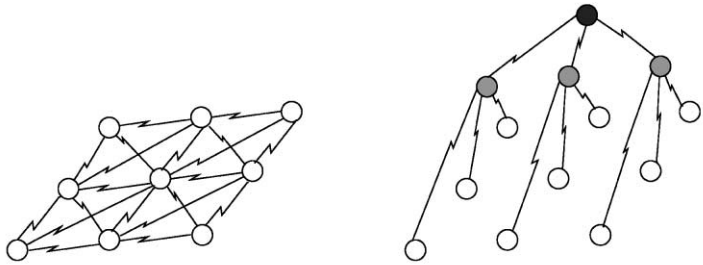


Fig. 3. AC and DHC structures from Wesson et al. [5].

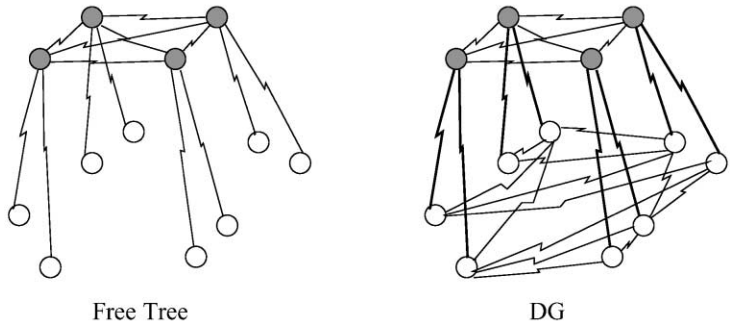


Fig. 4. Flat tree and DG network.

connection between sensor nodes in AWSNs is dynamically changing. A short-lived network is set up only for the communication needs of the moment [13].

Take the example of battlefield which provides daunting challenges to sensor fusion networks. Lightweight, inexpensive, highly specialized sensors are usually deployed with irregular patterns in a hostile environment. Each individual sensor node may come and go, and they may also suffer intermittent connectivity due to high error rate of wireless link, and it can be further deteriorated by environmental hazard. Therefore, an effective sensor fusion network must be able to provide robust communication infrastructure and survivability to cope with node failures, connectivity failures, and individual service failures (e.g., one type of specialized sensor node stops functioning).

Lim [14] proposed the distributed service concept for information dissemination in self-organizing sensor networks. In a dynamic network environment, adaptive methods need to be used to control the system. Specifically, he proposed three fundamental services: service lookup, sensor node composition, and dynamic adaptation.

Wang [15] presented an architecture based on smart sharing space concept, where each sensor node and the services are represented as "resource objects" and registered with a central "lookup" service repository, which are visible to all the sensor nodes within the domain or range of transmission power. Information sharing, coordination and fusion are achieved through smart management of resources object. Robustness is achieved through automatic fail-over when failed sensor node can no longer maintain its service.

Estrin et al. [16] designed directed diffusion—a localized algorithm to establish flexible, efficient data delivery paths in AWSN. The basic idea is inspired by biological systems, application simple systems proposed by Van Jacobson. The communication primitives here is not expressed in terms of nodes generating or requesting data, but in terms of the named data. That is, consumer of data will initiate interests in data with certain attributes. Nodes then will diffuse the interests towards producers via a sequence of local interactions. This process sets up gradients in the network which channel the delivery of data. Even though the network status is dynamic (caused by dynamic operating conditions, dynamic availability of resources, and dynamic tasks), the impact of dynamics can be well localized.

3. Data processing paradigm

No matter how different the network structure is, the current data processing approaches tend to use a common network computing model: the client/server model. Client/server model has been supporting many distributed systems, such as remote procedure calling (RPC) [17], common object request broker architecture (CORBA) [18,19], etc. In this model, the client (individual sensor) sends data to the server (processing element) where data processing tasks are carried out. This model, however, has several drawbacks [20] which can be summarized as follows: First, the client/server model usually requires many round trips over the network in order to

complete one transaction. Each trip creates network traffic and consumes bandwidth. In a system with many clients and/or many transactions, the total bandwidth requirements may exceed available bandwidth, resulting in poor performance of the system. Second, the client/server model also requires the network connection to be alive and healthy, the entire time a transaction is taking place. If the connection goes down, the transaction has to restart if it can at all. Third, the design of a client/server-based system requires precise consideration of the network traffic, the number of clients and servers, transaction volumes, etc. If the estimates are inaccurate, the performance of the system will suffer. Unfortunately, once implemented, it is hard to make any modification to the system.

3.1. Mobile-agent-based DSN

Mobile agent paradigm was proposed in [21] to respond to the above challenges. The corresponding DSN is referred to as mobile-agent-based DSN (MADSN). MADSN adopts a new computation paradigm: data stay at the local site, while the integration process (code) is moved to the data sites. By transmitting the computation engine instead of data, MADSN offers the following important benefits:

- Network bandwidth requirement is reduced. Instead of passing large amounts of raw data over the network through several round trips, only the agent with small size is sent, This is especially important for real-time applications and where the communication is through low-bandwidth wireless connections.
- Better network scalability. The performance of the network is not affected when the number of sensor is increased. Agent architectures that support adaptive network load balancing could do much of a redesign automatically [20].
- Extensibility. Mobile agents can be programmed to carry task-adaptive fusion processes which extends the capability of the system.
- Stability. Mobile agents can be sent when the network connection is alive and return results when the connection is re-established. Therefore, the performance of MADSN is not much affected by the reliability of the network.

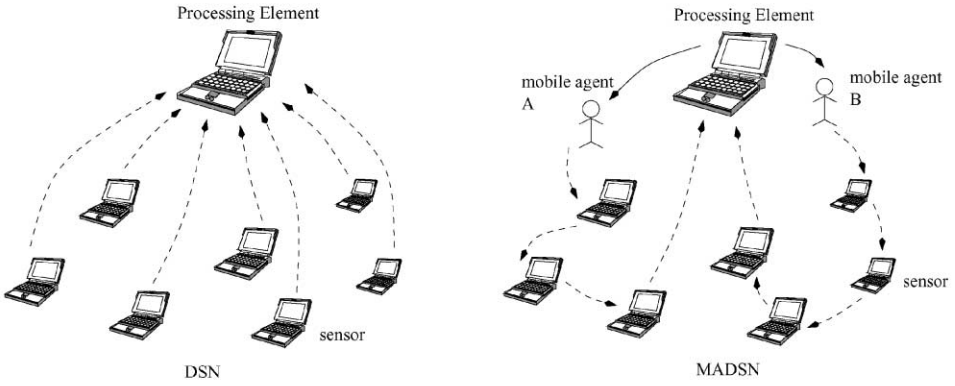
Generally speaking, mobile agent is a special kind of software which can execute autonomously. Once dispatched, it can migrate from node to node performing data processing autonomously, while software can typically only execute when being called upon by other routines, Franklin and Graesser provided a formal definition of agent [22]. Lange listed seven good reasons to use mobile agents [23], including reducing network load, overcoming network latency, robust and fault-tolerant performance, etc. Although the role of mobile agents in distributed computing is still being debated mainly because of the security concern [24,25], several applications have shown clear evidence of benefiting from the use of mobile agents. For example, mobile agents are used in networked electronic trading [26], where they are dispatched by the buyer to the various suppliers to negotiate orders and deliveries, and then return to the buyer with their best deals for approval. Instead of having the

buyer contact the suppliers, the mobile agents behave like representatives, interacting with other representatives on the buyer's behalf, and alert the buyer when something happens in the network that is important to the buyer. Another successful example of using mobile agents is distributed information retrieval and information dissemination [27–30]. Agents are dispatched to heterogeneous and geographically distributed data bases to retrieve information and return the query results to the end-users. Mobile agents are also used to realize network awareness [31] and global awareness [32]. Network-robust applications are of great interest in military situations today. Mobile agents are used to be aware of and reactive to the continuously changing network conditions to guarantee successful performance of the application tasks.

Fig. 5 provides a comparison between DSN and MADSN from both feature and architecture points of view.

Features	DSN	MADSN
Elements moving on the network?	Data	Computation
Bandwidth consumption?	High	Low
Scalable?	No	Yes
Extensible?	No	Yes
Affected by network reliability?	Yes	No
Fault-tolerable?	Yes	Yes

(a)



(b)

Fig. 5. Comparison between DSN and MADSN: (a) feature, (b) architecture.

4. Sensor fusion algorithms

Information/decision fusion problems are centuries old, dating back to the works of Condorcet on democracy models in 1786 and Laplace on composite methods in 1818. In engineering and systems, early works in this area are due to von Neumann, who showed that a reliable system can be built using unreliable components by employing simple majority fusers.

Over the past decade, there has been a dramatic increase in the application areas in which information/decision fusion methods have been employed, and sensor fusion is one of the most active application areas. As larger amount of sensors are deployed in harsher environment, it is important that sensor fusion techniques are robust and fault-tolerant so that they can handle uncertainty and faulty sensor readouts. Here, the redundancy in the sensor readouts are used to provide error tolerance in fusion. The review in this section focuses on the development of fault-tolerant fusion algorithms. Four simple functions were developed representing four milestones: the M function from Marzullo [33], the S function from Schmid and Schossmaier [34], the multi-resolution analysis (MRA) of the Ω function from Prasad et al. [35], and the N function from Cho et al. [36].

All the functions assume to process the readouts from abstract sensors. An *abstract sensor* is defined as a sensor that reads a physical parameter and gives out an abstract interval estimate which is a bounded and connected subset of the real line. Abstract sensors can be classified into *correct sensors* and *faulty sensors*. A *correct sensor* is an abstract sensor whose interval estimate contains the actual value of the parameter being measured. Otherwise, it is a *faulty sensor*. A faulty sensor is *tamely faulty* if it overlaps with a correct sensor, and is *wildly faulty* if it does not overlap with any correct sensor.

Fig. 6 shows the original readout from four sensors I_1, I_2, I_3, I_4, and the fusion results using different functions. I_2 is deliberately vibrated to I_2' to compare the robustness of the four functions. Let n be the number of sensor readouts, f be the number of faulty sensors, in this case, $n=4$ and $f=1$.

$M([I_1, I_2, \ldots, I_n])$ is defined to be the smallest interval that contains all the intersections of $n-f$ intervals. It is guaranteed to contain the true value provided the number of faulty sensors is at most f. However, M function exhibits an unstable behavior in the sense that a slight difference in the input may produce a quite different output. This behavior was formalized as violating Lipschitz condition with respect to a certain metric on intervals [37].

S function is able to return a closed interval $[a,b]$, where a is the $(f+1)$th left end point of the intervals and b is the $(n-f)$th right end point of the intervals. S function is proved to satisfy Lipschitz condition but generates a larger output interval. In another word, S function sacrifices accuracy to satisfy Lipschitz condition.

The Ω function is also called the overlap function. It returns $\Omega(x)$ gives the number of intervals overlapping at x. Multi-resolution analysis provides a hierarchical framework for interpreting the overlap function. Based on the prior knowledge that tamely faulty sensors cluster around correct sensors and create high and wide peaks in the profile of $\Omega(x)$, and that wildly faulty sensors do not overlap with correct

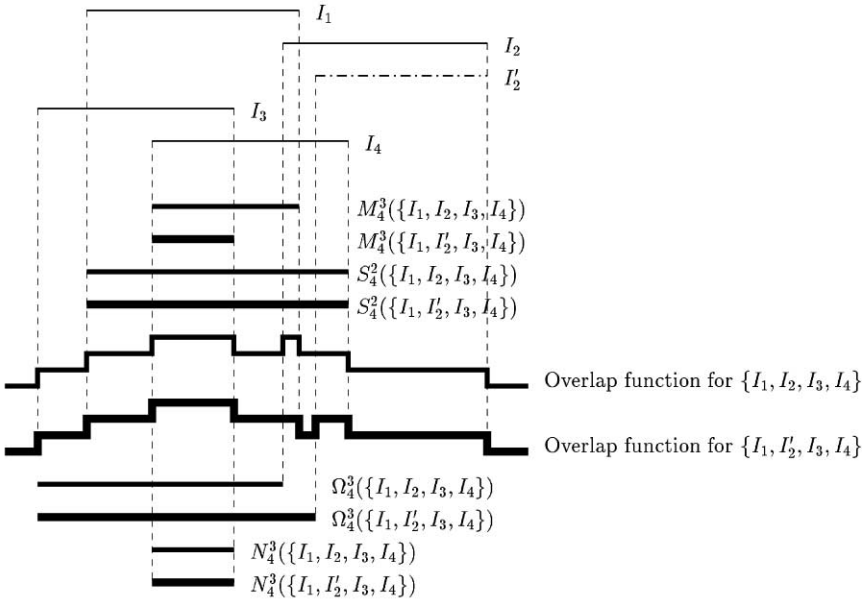


Fig. 6. Comparison of robustness of different functions to small vibrations in the sensor readout.

sensors, and therefore contribute to smaller and narrower peaks, Ω function results in an integration interval with the highest peak and the widest spread at a certain resolution. The Ω function is also robust, satisfying Lipschitz condition, which ensures that minor changes in the input intervals cause only minor changes in the integrated result. However, it also generates wider intervals than the M function.

The N function improves the Ω function to only generate the interval with the overlap function ranges $[n-f, n]$. It also satisfies Lipschitz condition. But the biggest advantage of this function is that it is able to reduce the width of the output interval in most cases and produce a narrower output interval when the number of sensors involved is large, which is the case for distributed sensor network in general.

5. Sensor deployment

Until now, most work in DSNs has concentrated exclusively on efficient sensor communication [16,38] and sensor fusion [39,40] for a given sensor field architecture. However, as sensors are used in greater numbers for field operation, efficient deployment strategies become increasingly important. Indeed, it is fair to state that the extensive research in this area has not yet led to a firm grasp of sensor deployment strategies for target location. This lack of understanding is not altogether surprising because the sensor deployment combines the hitherto unexplained interaction of target location with optimal placement of sensors.

5.1. Optimal sensor placement

In a typical scenario, battlefield commanders or surveillance authorities have several different types of sensors available which can be appropriately placed in the sensor field. These sensors differ from each other in their monitoring range, detection capabilities, and cost. Clearly, sensors that can accurately detect targets at longer distances have higher cost. However, the use of these expensive, long-range sensors may be prohibitive in terms of total deployment cost. On the other hand, if only small-range sensors are used, effective surveillance can only be achieved with a large number of these sensors. Therefore, optimal sensor deployment strategies are necessary to minimize cost and yet achieve mandated levels of surveillance accuracy. Fig. 7 shows a sensor field in which grid points (circles) are at distances of 100 m and two sensors are shown with different costs and range of coverage.

The sensor placement problem for target location is closely related to the alarm placement problem described in [41]. The latter refers to the problem of placing "alarms" on the nodes of a graph G such that a single fault in the system (corresponding to a single faulty node in G) can be diagnosed. The alarms are therefore analogous to sensors in a sensor field. It was shown in [41] that the alarm placement problem is NP-complete for arbitrary graphs. However, Chakrabarty et al. [42] shows that for restricted topologies, e.g. a set of grid points in a sensor field, a coding theory framework can be used to efficiently determine sensor placement.

In [42], the sensor field is represented as a grid (two- or three-dimensional) of points (coordinates), and sensors are selectively placed on a subset of these grid points. An integer linear programming (ILP) model is developed for minimizing the cost of sensor deployment under the constraint of complete coverage of the sensor field. For two-dimensional sensor fields with a given number ($p=8$) of grid points in each dimension using two types of sensors (a type-A sensor with cost \$150 and range 100 m, and a type-B sensor with cost \$200 and range 200 m), complete coverage for using 20 sensors (sensor density = 0.31) can be obtained. This involves using four sensor fields with 16 grid points each, and is shown in Fig. 8.

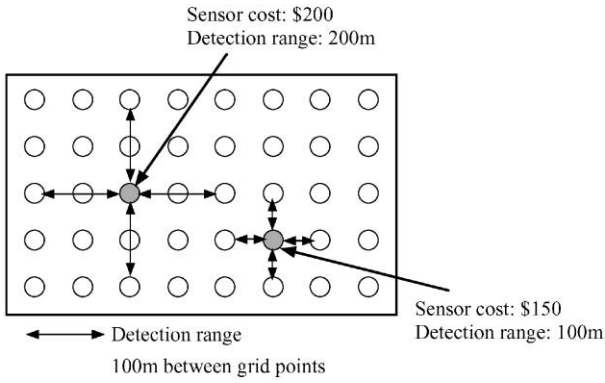


Fig. 7. An example of a two-dimensional sensor field.

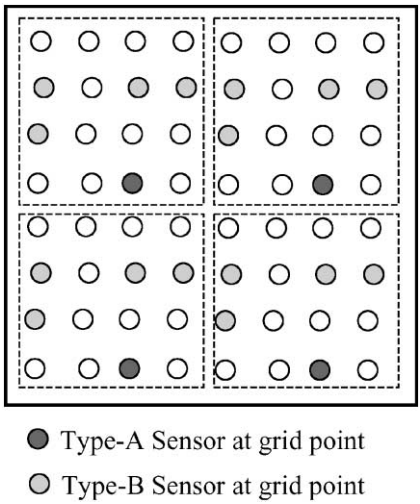


Fig. 8. Sensor placement for $p=8$ using a divide-and-conquer approach.

5.2. Sensor placement in target location

Another associated problem in sensor networks is that of target location. If the sensor field is represented as a grid (two- or three-dimensional) of points (coordinates), target location refers to the problem of pin-pointing a target at a grid point at any point in time. For enhanced coverage, a large number of sensors are typically deployed in the sensor field, and if the coverage areas of multiple sensors overlap, they may all report a target in their respective zones. The precise location of the target must then be determined by examining the location of these sensors. In many cases, it is even impossible to precisely locate the target (within the granularity of a single grid point). Alternatively, target location can be simplified considerably if the sensors are placed in such a way that every grid point in the sensor field is covered by a unique subset of sensors. In this way, the set of sensors reporting a target at time t uniquely identify the grid location for the target at time t. The trajectory of a moving target can also be easily determined in this fashion from time series data. Chakrabarty et al. [42] also provided coding-theoretic bounds on the number of sensors and presented methods for determining their placement in the sensor field.

For a two-dimensional sensor field with $p=13$, we need a total of 65 sensors for 169 grid points (sensor density = 0.38), which is slightly greater than the lower bound of 57 predicted, as shown in Fig. 9.

Even though Chakrabarty et al. [42] assumes that all sensors in the sensor field are fixed, extensions to mobile sensors are straightforward and can easily be incorporated. The detection range of mobile sensors is greatly enhanced—this can be incorporated into the ILP framework by modeling the fact that a mobile sensor can cover a large number of grid points. Mobile sensors are often desirable since they

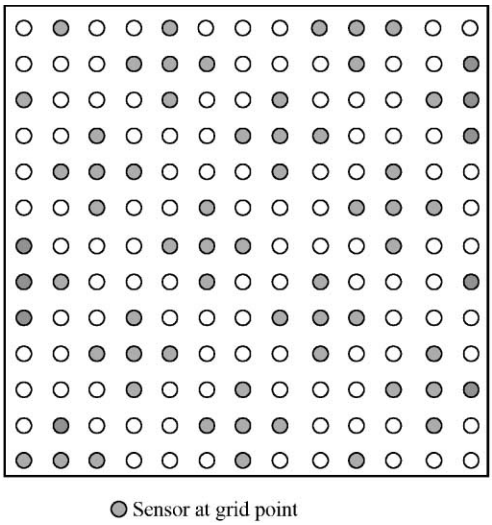


Fig. 9. A checkerboard placement of sensors.

can patrol a wide area, and they can be re-positioned for better surveillance. Mobile sensor can also be concealed when vulnerable. Therefore, mobility enhances sensor performance and survivability. An advantage of the ILP framework is that in addition to modeling the advantages of mobile sensors, it can also model the fact that mobile sensors are more expensive than fixed sensors. It therefore facilitates trade offs between cost, performance, and survivability.

6. Summary

This paper briefly discusses recent developments in the study of distributed sensor networks (DSNs). Advances in sensor technology and wireless communication have brought new challenges to this field. We reviewed innovative approaches from four aspects: the network structure design for traditional DSNs and for wireless ad hoc sensor networks (WASNs), the comparison between two data processing paradigm—client/server model and mobile-agent-based DSNs, the sensor fusion algorithm with a performance evaluation among four fault-tolerant integral integration functions, and the sensor placement strategy with an application example on target location.

References

[1] S.S. Iyengar, D.N. Jayasimha, D. Nadig, A versatile architecture for the distributed sensor integration problem, IEEE Trans. Comput. 43 (2) (1994) 175–185.

[2] D.N. Jayasimha, S.S. Iyengar, R.L. Kashyap, Information integration and synchronization in distributed sensor networks, IEEE Trans. Systems, Man, Cybernet. SMC-21(21) (1991) 1032–1043.
[3] A. Knoll, J. Meinkoehn, Data fusion using large multi-agent networks: an analysis of network structure and performance, In: Proceedings of the International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), Las Vegas, NV, October 2–5 1994, IEEE, pp. 113–120.
[4] L. Prasad, S.S. Iyengar, R.L. Kashyap, R.N. Madan, Functional characterization of sensor integration in distributed sensor networks, IEEE Trans. Systems, Man, Cybernet. SMC-21 (5) (1991) 1082–1087.
[5] R. Wesson, F. Hayes-Roth, J.W. Burge, C. Stasz, C.A. Sunshine, Network structures for distributed situation assessment, IEEE Trans. Systems, Man, Cybernet. SMC-11(1) (1981) 5–23.
[6] ISIF, International society of information fusion—publication list, http://www.infofusion.org/isif/publications.html.
[7] I. Bloch, Information combination operators for data fusion: a comparative review with classification, IEEE Trans. SMC—Part A: Systems and Humans, 26 (1) (1996) 52–67.
[8] R.S. Blum, S.A. Kassam, H.V. Poor, Distributed detection with multiple sensors: Part ii—advanced topics, Proc. IEEE, 85 (1) (1997) 64–79.
[9] M. Kokar, K. Kim, Review of multisensor data fusion architectures and techniques, Proceedings of the International Symposium on Intelligent Control, Chicago, IL, August 1993. IEEE, pp. 261–266.
[10] R.C. Luo, M.G. Kay, Multisensor integration and fusion in intelligent systems, IEEE Trans. Systems, Man, Cybernet. 19 (5) (1989) 901–931.
[11] R. Viswanathan, P.K. Varshney, Distributed detection with multiple sensors: Part i—fundamentals, Proc. IEEE 85 (1) (1997) 54–63.
[12] M.R. Samantham, D.K. Pradhan, The debruijn multiprocessor network: a versatile parallel processing and sorting network for vlsi, IEEE Trans. Comput. 38 (4) (1989) 576–581.
[13] C.E. Perkins, Ad Hoc Networking, Addison-Wesley, Reading MA, December 2000.
[14] A. Lim, Distributed services for information dissemination in self-organizing sensor networks, J. Franklin Institute 338 (6) (2001) 707–727.
[15] F. Wang, Smart management of resource objects in distributed sensor networks, 2001, in preparation.
[16] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, Next century challenges: scalable coordination in sensor networks, ACM/IEEE International Conference on Mobile Computing and Networks, 1999, pp. 263–270.
[17] A.D. Birrell, B.J. Nelson, Implementing remote procedure calls, ACM Trans. Comput. Systems 2 (1) (1984) 39–59.
[18] S. Baker, Corba implementation issues, IEE Colloq. (Digest) (007) (1994) 5/1–5/3.
[19] A. Watson, Omg (object management group) architecture and corba (common object request broker architecture) specification, IEE Colloq.—Digest, (007) (1994) 4/1.
[20] Todd Sundsted, An introduction to agents, Java World, June 1998.
[21] H. Qi, S.S. Iyengar, K. Chakrabarty, Multi-resolution data integration using mobile agents in distributed sensor networks, IEEE Transactions on SMC: C (2000), submitted.
[22] S. Franklin, A. Graesser, Is it an agent, or just a program?: a taxonomy for autonomous agents, In: J.G. Carbonell, J. Siekmann, Third International Workshop on Agent Theories, Architectures, and Languages. Springer, Berlin, 1996, http://www.msci.memphis.edu/franklin/AgentProg.html.
[23] D.B. Lange, M. Oshima, Seven good reasons for mobile agents, Commun. ACM, 42 (3) (1999) 88–89.
[24] C.G. Harrison, D.M. Chess, A. Kershenbaum, Mobile agents: are they a good idea? Technical Report RC 19887, IBM Thomas J. Watson Research Center, March 1995, http://www.research.ibm.com/massive/mobag.ps.
[25] D. Milojicic, Mobile agent applications, IEEE Concurrency 7 (3) (1999) 80–90.
[26] P. Dasgupta, N. Narasimhan, L.E. Moser, P.M. Melliar-Smith, Magnet: mobile agents for networked electronic trading, IEEE Trans. Knowledge Data Eng. 11 (4) (1999) 509–525.
[27] M. Hattori, N. Kase, A. Ohsuga, S. Honiden, Agentb-based driver's information assistance system, New Generation Comput. 17 (4) (1999) 359–367.
[28] J. Kay, J. Etzl, G. Rao, J. Thies, Atl postmaster: a system for agent collaboration and information dissemination, Proceedings of the Second International Conference on Autonomous Agents, Minneapolis, MN, 1998, ACM, pp. 338–342.

[29] T. Oates, M.V.N. Prasad, V.R. Lesser, Cooperative information-gathering: a distributed problem-solving approach, IEE Proc.—Software Eng. 144 (1) (1997) 72–88.
[30] J.S. Wong, A.R. Mikler, Intelligent mobile agents in large distributed autonomous cooperative systems, J. Systems Software 47 (2) (1999) 75–87.
[31] W. Caripe, G. Cybenko, K. Moizumi, R. Gray, Network awareness and mobile agent systems. IEEE Commun. Mag. July 36 (7) (1998) 44–49.
[32] K.N. Ross, R.D. Chaney, G.V. Cybenko, D.J. Burroughs, A.S. Willsky, Mobile agents in adaptive hierarchical bayesian networks for global awareness, Proceedings of the IEEE International Conference on Systems, Man and Cybernet, IEEE, 1998, pp. 2207–2212.
[33] K. Marzullo, Tolerating failures of continuous-valued sensors, ACM Trans. Comput. Systems 8 (4) (1990) 284–304.
[34] U. Schimd, K. Schossmaier, How to reconcile fault-tolerant interval intersection with the lipschitz condition? Technische Universitat Wien, Department of Automation, Vienna, 1999, submitted for publication.
[35] L. Prasad, S.S. Iyengar, R.L. Rao, Fault-tolerant sensor integration using multiresolution decomposition, Phys. Rev. E 49 (4) (1994) 3452–3461.
[36] E. Cho, S.S. Iyengar, K. Chakrabarty, H. Qi, A new fault tolerant sensor integration function satisfying local lipschitz condition, 2000, submitted for publication.
[37] L. Lamport, Synchronizing time servers, Technical Report 18, Digital System Research Center, 1987.
[38] J.M. Kahn, R.H. Katz, K.S.J. Pister, Mobile networking for smart dust, ACM/IEEE International Conference on Mobile Computing and Networks, 1999.
[39] R.R. Brooks, S.S. Iyengar, Multi-Sensor Fusion: Fundamentals and Applications with Software. Prentice-Hall PTR, Upper Saddle River, NJ, 1998.
[40] S.S. Iyengar, L. Prasad, H. Min, Advances in Distributed Sensor Technology, Prentice-Hall, Englewood Cliffs, NJ, 1995.
[41] N.S.V. Rao, Computational complexity issues in operative diagnosis of graph-based systems, IEEE Trans. Comput. 42 (1993) 447–457.
[42] K. Chakrabarty, S.S. Iyengar, H. Qi, E. Cho, Coding theory framework for target location in distributed sensor networks, IEEE International Conference on Information Technology: Coding and Computing, June 2001.