

# Content Management and Replication in the SNSP:

## A Distributed Service-based OS for Sensor Networks

Jana van Greunen, Jan Rabaey  
Department of Electrical Engineering  
University of California, Berkeley  
Berkeley, CA, USA

**Abstract**—This paper presents data management and replication techniques used by the SNSP, a distributed operating system for sensor networks. Three replication schemes, a deterministic, a distributed probabilistic and an adaptive observation based scheme are compared. The first two were adapted from related work and the latter was developed for the SNSP. Results indicate that the distributed scheme has the best performance in terms of total cost per data access as well as increased data availability. When access cost without overhead is considered, the adaptive algorithm performs the best, but its overhead is higher because data items are replicated independently of accesses.

**Keywords**—File allocation; Distributed operating system

### I. INTRODUCTION

Widespread sensor network deployment has been hampered by lack of a standard hardware abstraction and software API. The hardware that comprises a sensor network is a collection of distributed nodes functioning as a whole. While there have been successful abstractions for the individual sensor node, e.g. IEEE's sensor interface standard [1] and mote abstraction [2], abstractions for the distributed network are only now emerging. This paper presents the Sensor Network Services Platform (SNSP), a distributed operating system for sensor networks, and focuses on the content management and replication.

The SNSP is a full-fledged operating system with memory management, location transparency, and resource allocation. The SNSP enables the creation of applications that can be mapped onto the network at runtime and allows the programmer to build up a library of reusable sensor network services.

The content management and replication problem, which is the focus of this paper, has been studied in the context of networked systems and many algorithms exist. However, these algorithms have largely been designed to operate in traditional networked environments, with high bandwidth connections and reliable nodes. The sensor network characteristics present these unique requirements and challenges:

- High rate of node failure / duty cycling
- Low Bandwidth
- Power and energy constraints
- Heterogeneous network

---

The authors wish to acknowledge the Berkeley Wireless Research Center, the National Science Foundation Infrastructure Grant No. 0403427, and the support of the Gigascale Systems Research Center, one of five research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program.

This paper presents the performance of these algorithms in the sensor network setting. The algorithms are aimed at reducing communication overhead and increasing data availability. The paper is organized as follows: The next section outlines a formal problem statement. Section 2 contains related work and is followed by Section 3, which presents details about the SNSP in which the content management operates. Three file allocation algorithms are outlined in Section 4. Section 5 details the simulation setup and results are presented in Section 6. The paper is concluded in Section 7

### A. Problem Statement

The content management and replication is also known as the file allocation problem (FAP) and has been extensively studied for traditional distributed databases. The FAP may be static or dynamic; the access patterns may be deterministic, probabilistic or unknown. All versions of the FAP are NP complete. A version of the static FAP, is formulated below.

Given:

- $n$  nodes and  $m$  files
- $l_j$  is the length of the  $j^{\text{th}}$  file for  $0 < j \leq m$
- $b_i$  is the available memory of the  $i^{\text{th}}$  node for  $0 < i \leq n$
- $c_j$  is the storage cost per unit at the  $j^{\text{th}}$  node
- $CA_i$  is the capacity of the  $i^{\text{th}}$  link

And a poisson traffic vector where:

- $u_{ij}$  is the query traffic originating from node  $i$  for file  $j$
- $v_{ij}$  is the update traffic originating at node  $i$  for file  $j$
- $u'_{ij}$  is the return traffic to node  $i$  from queries for file  $j$
- $v'_{ij}$  is the return traffic to node  $i$  from updates of file  $j$

Chu [3] formulated the file allocation problem as a zero-one integer-programming problem. The variable  $X_{ij}$  indicates that the  $j^{\text{th}}$  file is stored on the  $i^{\text{th}}$  node.

$$X_{ij} = \begin{cases} 1 & \text{if } j^{\text{th}} \text{ file is stored on the } i^{\text{th}} \text{ node} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Storage cost is given by:

$$Z_{\text{store}} = c^T X l \quad (2)$$

Where  $c = (c_1, c_2, \dots, c_n)^T$  and  $l = (l_1, l_2, \dots, l_m)^T$

Define  $q_j$  the replicas of file  $j$  in the network. The total traffic generated in the network is now:

$$Z_{comm} = \sum_{i=1}^n \sum_{j=1}^m [(u_{ij} + u'_{ij})(1 - X_{ij}) + (v_{ij} + v'_{ij})(q_j - X_{ij})] \quad (3)$$

For the SNSP, storage capacity and availability constraints can be formulated as follows:

$$C_{cap} = \sum_j X_{ij} L_j \leq b_i \quad \text{for } 1 \leq i \leq n \quad (4)$$

Analytical expressions for availability are hard to obtain, so an approximation, given in [4] is used and explained below. First, we define  $r_{ik}$  to be the probability that two nodes  $i$ , and  $k$  successfully communicate (these may not be independent for  $r_{mn}$ , with  $m, n$  not equal to  $i, k$  but it is assumed that they are for the approximation). Note, this probability includes the probability that node  $k$  is available. The availability  $a_{ij}$  of file  $j$  accessed by node  $i$  is

$$a_{ij} = 1 - \prod_{k=1}^n (1 - r_{ik} X_{kj}) \quad (5)$$

Let  $W_{ij}$  be the total traffic at node  $i$  for file  $j$ .  $W_{ij} = u_{ij} + u'_{ij} + v_{ij} + v'_{ij}$ . The traffic-weighted availability for each file is:

$$C_{avail}^j = \frac{\sum_{i=1}^n W_{ij} a_{ij}}{\sum_{i=1}^n W_{ij}} \quad (6)$$

The optimization problem given a target  $t$ , availability  $a$ , and parameter  $\lambda$ , is:

Find an allocation  $F(X)$  that minimizes

$$F(X) = Z_{store} + \lambda Z_{comm} \quad (\text{eqn 2, 3})$$

subject to

$$C_{cap} \quad (\text{eqn 4})$$

$$C_{avail}^j > a \quad \text{for } 0 < j \leq m \quad (\text{eqn 6})$$

## II. RELATED WORK

Content replication algorithms were studied extensively with the rise of networked computer systems in the late 60's and 70's. The optimal solution is NP complete. Techniques for FAP include branch and bound, randomization, predictive markov techniques, genetic algorithms and other heuristics. [5] gives a good overview of these difference techniques.

The content replication problem falls into a class of problems, called on-line problems, where the input pattern is not known in advance. Typically, competitive algorithms are used to solve/analyze online problems. Their performance on any sequence of requests is within a constant factor of the performance of any other algorithm (including the optimal). See [6] for more details. This work will compare algorithms that do not know the access pattern in advance to each other.

## III. SNSP

As an operating system, the SNSP manages the system resources e.g. memory and computation, allows multiple programs to co-exist, controls input/output devices and manages content. Further, the SNSP will execute on a physically distributed platform and thus implicitly supports communication, concurrency, and synchronization, providing

location transparency. As in traditional operating systems, a user's rights determine which programs, resources, and content they may read, write, and execute.

### A. SNSP Goals

The SNSP fits into the [7] goals that aim to seamlessly integrate disparate components into a functioning system. In addition, there are three goals that the SNSP aims to achieve in the sensor network setting:

- Enable applications to execute on different network configurations (agnostic of hardware/software)
- Shield the application (enable recovery) from node/hardware failures
- Optimize execution by mapping the application onto the network at runtime to save energy by reducing communication, increasing reliability and/or performance

As a general platform/middleware that must execute on all sensor network platforms, the SNSP may have access to only a few parameters from any given hardware platform and/or application. The following assumptions were made:

- Network consists of heterogeneous nodes, which are globally asynchronous and run SNSP middleware.
- The SNSP exposes geographic addressing.
- The SNSP knows the average throughput (Mb/s) between adjacent nodes.
- The hardware abstraction for each sensor node is:
  - MIPS number for computation
  - Dynamic memory in KB
  - Storage for data in MB
  - Hardware (sensors/actuators)

The SNSP lies above the communication medium so metrics like latency and throughput cannot be guaranteed. Instead, the SNSP adopts the *BASE* semantics: Basically Available, Soft State, Eventual Consistency. These were introduced in [8].

The SNSP programming model is supported by pre-defined services. A service-oriented model was chosen to separate the *content* of network services (i.e., what the services do) from their implementation. There are two kinds of services in the SNSP: (1) Operating-system services, and (2) User-defined services. Services can be considered a library of applications that conform to the SNSP's service API. Examples of the SNSP application model are outlined in [7].

### B. OS-level Services

OS services are the lowest level services that keep track of the system at the device and connectivity level. These functions have been divided into seven services, which are briefly treated in this report.

#### 1) Resource Discovery & Repository Service

In the SNSP we have chosen to do reactive resource discovery. Nodes send out a register message when they join the distributed system. The register message contains information about the resources present on the node, as well as

content and/or service code that may be present on the node. This information is stored as repository content by the content management and replication service.

### 2) Resource Utilization

The resource utilization service gives information about resources that are used by executing applications and services, in addition to the unallocated resources remaining in the sensor network. The mapper and content management service use this service.

### 3) Content Management and Replication

Section 4 presents and compares three heuristic solutions tailored for sensor networks.

### 4) Mapper

The mapper's task is to allocate resources to processes so that they can execute. This problem is closely related to the file allocation problem; the application is the set of files, and each file represents a service that the application utilizes. The processes will give the mapper additional constraints: namely, computation, dynamic memory, hardware, and location and bandwidth constraints. Considerations for the mapper are discussed more in [7]. Due to space limitations, heuristics for file allocation will be included in future publications.

### 5) Application Migration

When the mapper wants to relocate an application, it will request that the application checkpoints itself by creating a checkpoint object. The checkpoint object captures the applications' state; its current execution position, any local content that may have been generated as part of the process state, the services that the process are currently using and services/applications that have sent queries to the process (there are referred to as waiting services). This state is captured in the RemoteProcess object.

### 6) Fault Detection and Recovery

The SNSP supports three levels of fault tolerance: recoverable processes, fault detecting processes and no tolerance. Recoverable processes must create checkpoint objects. The checkpoint objects are stored and replicated by the content management service. Fault detecting processes will be re-started when a fault is detected, but are not guaranteed to preserve state.

### 7) Security Engine

As in [7], the security in the SNSP is based on persona and their access permissions. The implementation of this security is left as future work for the SNSP.

## IV. CONTENT REPLICATION ALGORITHMS

This section presents three heuristic algorithms: A deterministic central replication algorithm, a distributed probabilistic replication algorithm, and a learning-based algorithm. These algorithms are compared via simulation to a control case with no replication.

Additionally, content replication algorithms do not address *finding* the content. In the worst case, nodes will need to flood the network to locate information. In order to avoid flooding, an indexing system is used. Indices are placed at warehouse

nodes, which have higher availability and storage space than its peers. Warehouse nodes are selected via a clustering algorithm in a  $d$ -hop neighborhood. This work uses the clustering scheme presented in [9], which selects as cluster heads nodes within  $D$  hops with the highest id. This work uses a combination of the node's availability and storage as criteria for cluster head selection. The cost of maintaining the content indices is added as overhead cost. Note, the cost of accessing a warehouse will on average be  $D/2$ , the average distance a node is from its cluster head. Thus it is beneficial to reduce  $D$  to lower this cost. However, as  $D$  is decreased, the total number of cluster heads increases, which increases the cost to write a new location to all warehouses.

### A. Deterministic, central replication

This algorithm was adapted from [10]. The algorithm is  $O(\log n)$  competitive. For each data item  $p$  with size  $d$ , the algorithm maintains a list of  $L$  read requests and a tree  $T$  of the replicated data. The algorithm executes as follows:

- On read request  $r$ , find sphere  $S$  with radius  $k$  that contains  $d$  other reads
- If no copy exists within a sphere  $S'$  of radius  $\lambda k$  of  $r$ , the file is replicated to the highest availability node in  $S'$
- Read requests from nodes in  $S$  are deleted from  $L$
- After  $d$  writes, all copies in  $T$  are deleted except for the node with the highest availability.

### B. Distributed Alg

This algorithm was adapted from [11] to improve the availability of the data. It is  $O(\log^2 n / \log^2 d)$  competitive ( $d$  is the data size and  $n$  is the number of nodes). A tree  $T_p$  is kept for replicas of item  $p$ :

- On a read request from  $r$ , insert the node with the highest availability in  $r$ 's one-hop neighborhood in  $T_p$  with probability  $1/d$
- On a write request from  $w$ , with probability  $1/(\sqrt{3} * d)$ , delete all nodes in  $T_p$ , but the highest availability node.

### C. Adaptive observation based algorithm

This algorithm uses an observation period during which nodes observe the number of read/write requests. Replication decisions are made after this period.

Initial step:

- Observe 100 accesses of a data item, keep record of the location and whether it was a read/write.

Reallocation:

- After  $r$  reads
- Create  $k = \text{floor}(r/20)$  replicas of the data item
- Use  $k$ -means to divide the data requesters into  $k$  clusters and store one data item on the node within each cluster with highest availability.

Now that there are replicas in the network, writes can either come directly from a node (meaning that the replica was the closest copy to be written) or they can happen when data is written through for consistency.

Refinement steps (on each node that has a copy of  $d$ ):

- Observe 100 accesses of the data item: accesses are either reads  $r$ , direct writes  $w_d$ , or update writes  $w_u$ .
- If  $w_u > r + w_d$  remove the replica from the node
- If  $r > w_d$  create  $k = \text{floor}(r/20)$  replicas and use k-means to distribute the replicas onto a node with the highest availability in each cluster.

## V. SIMULATION SETUP

I used the discrete event simulator Omnet++ [12] with the Mobility Framework (MF) [13] extension. The simulation consists of 1000 nodes randomly placed in a 2000m x 2000m square. The nodes have a radio range of approximately 150m. The network's heterogeneity is also varied (these parameters were chosen to randomly to give a range of values), with a heterogeneous network featuring nodes with 3 amounts of storage: 10k, 50k, and 200k. Nodes with a large amount of storage are available 99.9% of the time, nodes with a medium amount of storage have a 95% availability and the smallest nodes have a 80% availability. In the homogenous network, nodes have 2 different amounts of storage, 50k and 100k, and they are available at rates 97% and 94% respectively.

When a node is available, it wakes and decides with probability 0.1 to read or write a randomly selected piece of data. Each simulation consisted on average of 39,000 queries. Nodes may also fail (according to their availability) when they wake up. All data items are the same size (100), but they have different read to write ratios of 1:1, 3:1, 15:1 and 100:1. Cluster head election occurred at the start of the simulation and the cluster heads did not change during the simulation. For the lower layers, a fading channel model with an Aloha MAC is used. At the network layer, geographic routing is used.

The following assumptions are made:

- Nodes know the id's of data items they want to access
- ID's are much smaller in length than the data item
- Data items are indivisible; original copies are not deleted
- There are no concurrent read/writes (few events in WSN)
- Node failures/deaths are modeled as an independent binomial process, independent of other nodes.
- Nodes know their own availability, i.e. the proportion of time that they are in functioning condition.

## VI. RESULTS

The first experiments examined the tradeoff between number of cluster heads, their availability, and the number of hops,  $D$ . Table 1 shows a summary of the number of cluster heads vs number of hops  $D$ . The results were similar across both topologies. All cluster heads were the highest availability nodes, even with  $D$  of 2. Further, there is a tradeoff between the number of hops  $D$  to reach a cluster head and the increase in clusters heads to keep synchronized when  $D$  gets small. Due to the relatively small change in number of clusters from 3 to 8 hops,  $D$  is chosen to be 3 for the remainder of the simulations.

Figure 1 shows the total cost, that is, the number of hops data was transported, per data access for each of the schemes. The solid portion of the bar represents the direct cost of the operation, for example, for a write that is the cost to transport data from the writing node to the closest replica. The top

portion of the bar is overhead to keep replicas consistent. Considering the data access cost, the adaptive algorithm performs the best, followed by the distributed algorithm, the deterministic algorithm and last the control algorithm.

TABLE I. NUMBER OF CLUSTER HEADS VS NUMBER OF HOPS  $D$

$D$	# Cluster Heads	Availability
2	10	99.9
3	6	99.9
4	5	99.9
8	3	99.9

However, considering total cost, the distributed algorithm outperforms the adaptive algorithm. This is because the distributed algorithm saves on replication cost by replicating to the nodes (or nodes very close to them) that make queries. While the adaptive algorithm has a better placement of replicas it incurs additional cost because it locates replicas independent of queries. The overhead of the deterministic algorithm makes it more costly than the control algorithm.

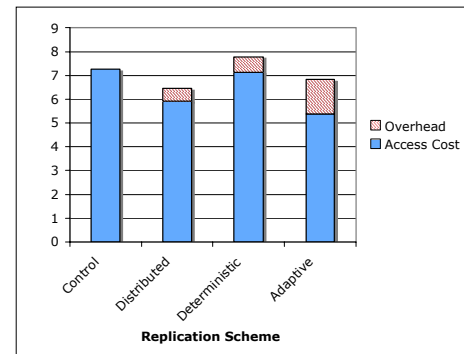


Figure 1. Comparison of data access cost and replication overhead.

Figure 2 shows the cost breakdown for the distributed and adaptive schemes by topology and data read/write ratio. Topology 1 is the heterogeneous topology and topology 2 is the homogeneous topology. On average, across all read/write ratio entries (for the same scheme) the two topologies differ by, 0.3 hops or roughly 4%.

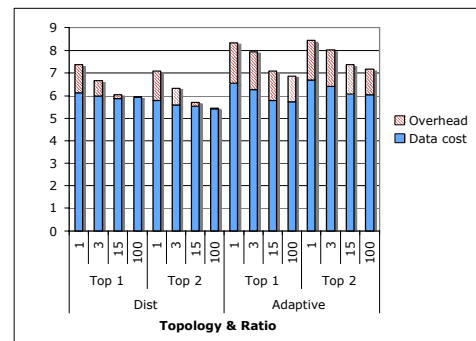


Figure 2. Comparison of data access cost vs topology and data read/write ratio.

The read/write ratio has a bigger impact on cost. In both schemes the overhead is much larger for data with lower read/write ratios. This overhead is negligible for data with a read/write ratio of 100 in the distributed algorithm. Intuitively, this is because data replication does not incur additional overhead, and even though there are many replicas, writes are so infrequent that they add little to the total cost. Thus, these schemes provide more benefit for data that is frequently read.

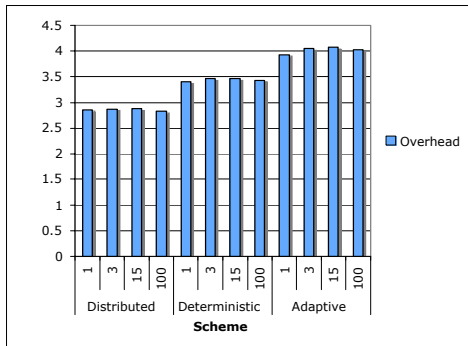


Figure 3. Comparison of control message overhead.

Figure 3 shows the control message overhead. The distributed algorithm has the lowest control message overhead. The data read/write ratio does not have a large impact on overhead cost because the overhead is dominated by the cost to find an item (there is not much difference in finding a location to read from or write to). Moreover, this overhead is less than the data cost shown in Figures 2 and 3. If data ids are a 10th the size of the data payload, then these number need to be divided by 10 to add them on the same scale as that of Figure 2, and on average, the distributed algorithm would cost 0.27 hops more, the deterministic algorithm would cost 0.34 more and the adaptive algorithm would cost 0.4 hops more. The control overhead becomes even more significant if the ratio between the data payload size and the data id size becomes larger.

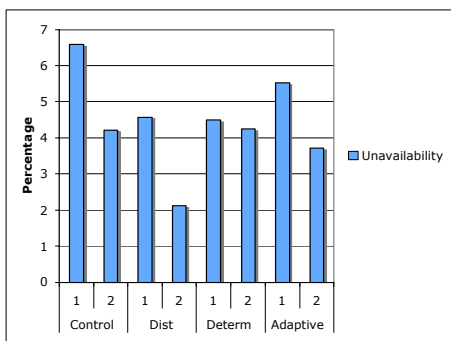


Figure 4. Percentage of unavailable data for different schemes & topologies

In addition to the cost of the replication scheme, the availability of data is also an important concern in the SNSP. Figure 4 shows the percentage availability of data for all schemes and topologies. Topology does have an impact on availability; the homogeneous topology has a higher

availability than the heterogeneous topology. The distributed replication improved reliability the most with a 33% improvement for topology 1 and a 51% improvement for topology 2.

## VII. CONCLUSION

This paper presented a formal problem definition of content replication and management. The problem was then investigated in the context of the SNSP, a distributed operating system, which dynamically maps data and programs onto the sensor network. Out of the three algorithms that were empirically compared, the distributed replication scheme performed the best with the lowest total cost and the highest availability. The adaptive algorithm developed for the SNSP performed the best when considering only data access cost, indicating that it had better placement than the other algorithms. However, it also had higher overhead.

Further the simulations showed that all replications schemes provided more benefit for data that is read more frequently than it is written. Finally, the simulations showed that although the replications schemes improved data availability, the sensor network topology still has an impact on the availability, with the homogenous network having roughly 2% more availability than the heterogeneous network. In conclusion, sensor networks can benefit from data replication to improve the cost of accessing data and increase the data's availability.

## REFERENCES

- [1] IEEE 1451 <http://iee1451.nist.gov/>
- [2] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, I. Stoica, A unifying link abstraction for wireless sensor networks Proc of the 3rd intl conf on Embedded networked sensor systems, p76-89, 2005.
- [3] W.W. Chu. Optimal File Allocation in a Multiple Computer System. IEEE Transactions of Computers, 18(10), 1969.
- [4] R. Tewari and N. R. Adam, "Distributed File Allocation with Consistency Constraints", Intl Conf on Distributed Computing Systems, pp. 408-415, 1992
- [5] L. W. Dowdy , D. V. Foster, Comparative Models of the File Assignment Problem, ACM CSUR, v.14 n.2, p.287-313, 1982
- [6] M. Manasse, L. McGeoch, and D. Sleator. Competitive algorithms for on-line problems Annual ACM Symp on Theory of Computing, 1988
- [7] C.R. Baker, Y. Markovsky, J. van Greunen, J. Rabaey, J. Wawrzynek, A. Wolisz, "ZUMA: A Platform for Smart-Home Environments," In Proc of IET Intelligent Environments, 2006.
- [8] A. Fox, S. D. Gribble, Y. Chawathe, E. A. Brewer, P. Gauthier, Cluster-Based Scalable Network Services, Symp on OS Principles, 1997
- [9] A. Amis, R. Prakash, T. Vuong, D. Huynh, Max-Min D-Cluster Formation in Wireless Ad Hoc Networks, Infocom, p 32-41, 2000
- [10] B. Awerbuch, Y. Bartal, and A. Fiat. Competitive Distributed File Allocation. In Proc of the 25th Ann. AMC Symp. On Theory of Computing, pp 164-173, 1993
- [11] Y. Bartal, A. Fiat, Y. Rabani. Competitive Algorithms for Distributed Data Management. 24th ACM STOC, 1992
- [12] A. Varga, "The OMNeT++ discrete event simulation system," in European Simulation Multiconference 2001.
- [13] W. Drytkiewicz, S. Sroka, V. Handziski, A. Köpke, H. Karl, A Mobility Framework for OMNeT++, 3rd Intl OMNeT++ Workshop, 2003.