What dependability for networks of mobile sensors?

Carole Delporte-Gallet
Univ. Paris 7
Carole.Delporte@liafa.jussieu.fr

Hugues Fauconnier
Univ. Paris 7
Hugues.Fauconnier@liafa.jussieu.fr

Rachid Guerraoui
EPFL, Lausanne
Rachid.Guerraoui@epfl.ch

Abstract

What kind of dependability could be expected in a network of mobile sensors? This position paper illustrates the issues underlying this question using a scenario recently developed at Yale [1]. The original scenario considers failure-free sensors and we discuss how to cope with failures given the limited power of sensors. We distinguish crash, transient and arbitrary failures and present some preliminary answers and some definitive questions beyond the scenario considered.

The Failure-Free Scenario

The illustrative scenario we consider, developed at Yale [1], considers a flock of birds (penguins). Every bird is equipped with a sensor that can determine whether the bird has temperature. If at least 5 birds have temperature, a special alert signal should eventually be generated and stored at every sensor; say a specific bit is turned to 1.

The sensors are limited [2, 4, 6]. Each sensor has a constant number of bits of memory and can respond to a global start signal. Sensors are assumed to be massively produced without individual identities. The sensors are mobile and two sensors can only communicate when they get close to each other. Clearly, the sensors do not control the pattern of mobility of their associated birds. This pattern is however assumed to be fair in the sense that every pair of birds in the same flock repeatedly come sufficiently close to each other for their sensors to communicate.

If no sensor is prone to failure, the algorithm to guarantee that every sensor eventually has the correct outcome is pretty simple. At the global start signal, each sensor makes a measurement, resulting in a 1 in case of temperature or 0 otherwise. This value is stored in a counter that can hold values from 0 to 4. Then the algorithm is initiated. When two sensors communicate, the first to initiate the communication sets its counter to the sum of the two counters and the other sets its counter to 0. If two counters ever sum to at least 5, the sensors go to the special alert state, which is then copied by every sensor that encounters them. Eventually, the correct outcome can be retrieved from any of the sensors.

Dependability Questions

The algorithm just sketched, as well as all those given in [1], preclude the possibility of sensor failures. Clearly, there are all but good reasons for sensors attached to penguins to fail.[1]

Assume that one sensor can indeed fail, and it can do so by crashing (i.e., by stopping all its activities). Consider a scenario where 5 birds have temperature, and one of those birds, associated with a sensor denoted by b, moves in such a way that b turns its counter to the sum whenever it meets another sensor with temperature. If b crashes, then there is no way for the alert signal to be generated. Remember that we assume a scenario where exactly 5 birds have temperature, but we could have gone up to 9 birds for the counter example to hold. In this case, 9 birds would have temperature and the alert would not be generated.

Are there algorithms that tolerate sensor failures? It is obvious to see that even if we assume that only one sensor can fail, and it can only do so by crashing, then

[1] The excellent movie *The Walk of the Emperor* is full of such evidences.

no algorithm can ensure that an alert signal is generated if and only if 5 birds have temperature. One of these birds might be associated with the faulty sensor.

- But what if, still assuming that only one sensor can fail and it can only do so by *crashing*, we require that a signal has to be generated if 5 birds have temperature and can only be generated if at least 4 birds have temperature?
- What if failing does not only mean crashing, but might mean that the sensor starts its computation in an arbitrary state, i.e., the algorithm runs correctly but from any initial corrupted state? Such a *transient* form of failure is pretty common in practice.
- What if failing might mean that the algorithm within a sensor can deviate arbitrarily from its specification (called *arbitrary or Byzantine* failure)?

In the following, we address these particular questions [2] and pose the more general questions of how to tolerate more failures or perform more sophisticated computations than a simple aggregation (i.e., beyond the scenario we consider for illustration purposes). Indirectly, we explore the general issue of what kind of dependability can be achieved with minimal system support. Impossibility results and lower bounds would call for more support and provide guidelines for how to augment the basic sensors.

Crash Failure

Assume first that one sensor can fail, and it can do so by crashing. We sketch an algorithm that solves the following problem.

1. If at least 5 birds have temperature then all correct birds are in alert;
2. If any bird is in the alert state then at least 4 correct birds have temperature.

Not surprisingly, we obtain a 1-crash-tolerant algorithm by duplicating the failure-free algorithm. However, duplication here is different from applying state-machine or primary-backup techniques to sensors [5, 3]. To tolerate 1 crash, the idea is to run two algorithms within every sensor.

That is, every sensor performs two concurrent failure-free algorithms. The state of a sensor is now twofold and consists of two counters: one that captures the state according to the first algorithm and one that captures the state according to the second algorithm. When two sensors interact, one of them plays the role of the initiator in the first algorithm and the second the initiator for the second algorithm.

One of the algorithms might not terminate with the required outcome because of the crash of the sensor that "owns" the critical information (counter example used earlier). But in this case this sensor does not own any information in the second algorithm, except possibly its own initial value. If in any of the two algorithms, one sensor moves to the alert state, then its state is copied to any new sensor that it meets.

We discuss now how to generalize this algorithm to tolerate k failures. The idea is to run $k+1$ (failure-free) algorithms per sensor. A sensor now only owns the crucial information in one algorithm and its state captures a state for each algorithm, as well as a number from 1 to $k+1$ indicating the algorithm where this sensor possibly owns crucial information.

Let s_i be any sensor owning possibly crucial information in the a_i-th algorithm. Initially each sensor considers that it owns the crucial information in the first algorithm. Each time s_i and s_j, with $a_i = a_j \neq k+1$, interact, s_j gathers the crucial information on $a_j + 1$ after the interaction. When s_i and s_j interact, with $a_i \neq a_j$, s_i (resp. s_j) plays the role of the initiator in algorithm a_i (resp. a_j) while s_j (resp. s_i) plays the role of the responder for this algorithm. When s_i and s_j interact, with $a_i = a_j = k+1$, s_i and s_j run one step of the $(k+1)$-th algorithm.

If in any of the $k+1$ algorithms, one sensors moves to the alert state, then its state is copied to any new sensor that it meets.

Transient Failures

The algorithm just sketched does not tolerate a single transient failure. That is, if a single sensor can start from any arbitrary state, [3] in particular the alert state, then all sensors will go to the alert state.

It is interesting to view the problem as an information exchange problem between agents (sensors) in a game in which the agents begin in some start squares. Some agents can have been initially disrupted. The agents play the games correctly (i.e. according to the rules of the game) but do not start the game in a start square.

[2] We adopt an informal approach for the sake of the workshop. More details can be found in a technical report to be published soon [9].

[3] Say, because the associated bird had troubles with a sea lion.

The idea of an algorithm to tolerate 1−transient failure is to divide the sensors into 3 categories. Each sensor is put in one category such that there is at least one sensor in each category. The failure-free algorithm is now run 3 times. Each algorithm is executed by the sensors of a given category. Sensors in any given category take into account values of sensors from other categories only for their initial value.

In this way, if the sensor that exhibit a transient fault is in category α, it can only corrupt the algorithm executed by the sensors of the same category α. The only impact on sensors from other categories is the initial value. If in two of the three algorithms, the sensors go in the alert state then the actual alert is generated.

The dispatching in categories is performed as follows. First a sensor tries to find its category. If the sensor meets another sensor in the initial state (or in category 1), one of them becomes (or remains) in category 1, and the other knows that some sensor is in category 1. If a sensor that knows at least one sensor in category 1, meets another sensor that is not in category 1, then one of them is in category 0 and the other in category 2. With n sensors, it is possible that $n/2$ are in category 1, and $n/4$ in category 0 or 2. And thus, it is necessary to have at least 4 sensors to obtain 3 categories.

After having found its category, the sensor runs the corresponding failure-free algorithm. During this phase, a sensor of category C looks if a sensor in category $C+1$ or $C+2$ modulo 3 is in the alert state. If one sensor from each is in the alert state, then it also moves to the alert state.

Tolerating k failures is slightly more complicated and is doable with $2k+1$ categories and 2^{2k} birds.

Arbitrary Failures

In the following, we argue that if at least one sensor can exhibit an arbitrary behavior (Byzantine), and which we call a twisted sensor, then the problem is impossible. More precisely, it is possible to trigger an alert even if no bird has temperature.

Consider a flock of at least 6 birds where all birds have temperature except one equipped with sensor b_0. Sensor b_0 moves to the alert state after it meets a sequence y of sensors. As there is no identity, this sequence is characterized by the state of the other sensors and their roles in the meetings: initiator or responder. Now consider a flock where no bird has temperature. The twisted sensor meets b_0 following the sequence describe by y. At the end b_0 is in the alert state. We iterate this proceeding. At the i-th iteration, the sensor b_i meets the twisted sensor following the sequence describe by y. At the end of the n-th iteration (where n is the population of sensors) all the sensors are in the alert state. As this configuration never changes, the alert is generated even without any bird with temperature.

This impossibility can be generalized to any non-trivial operation (beyond sum).

Wider Perspective

Our transformation from a failure-free algorithm to a crash tolerant is generalizable to other forms of operations, i.e., beyond the simple *sum* of the bird scenario. In particular, it applies to *average*, *majority*, etc. Coming up with a precise characterization of the class of operations for which a crash-tolerant algorithm is possible is however left as an open problem. This is even less clear with transient faults.

Another interesting direction is a randomized one where we would assume for instance a certain probability for any two sensors to communicate, instead of assuming that any pair of correct sensors does eventually communicate. More generally, studying the actual colleration between the mobility pattern and the possible fault-tolerance is an interesting research topic.

As we argued in the paper, a single arbitrary failure is impossible to tolerate. In fact, even with a weak specification of the problem (say with a possible abort output if a sensor detects any failure) the problem is impossible. In fact, equipping the sensors with some notion of identity circumvents the impossibility but it is not clear which notion is strictly necessary. Clearly, if all sensors have different identities without masquerading, and can store some of them (5 for example), then the problem becomes solvable. But are these requirements necessary? At first glance not. What about a weaker form of storage capacity with some proof of authenticated meeting?

Concluding Remark

We hope to have conveyed the idea that mobile sensor networks open many research questions in dependable computing, and that their frugal nature opens many problems in the intersection between algorithms and systems.

References

[1] Dana Angluin, James Aspnes, Zoe Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors.

In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 290–299. ACM Press, 2004.

[2] Q. Fang, F. Zhao, and L. Guibas. Ligtweight sensing and communication protocols. In *MOBIHOC'04: Proceedings of the 4th ACM Symposium on Mobile Ad Hoc Networking and Computing*, pages 165-176. ACM Press, 2003.

[3] D. Johansen, K. Marzullo, F. B. Schneider, K. Jacobsen,and D. Zagorodnov. NAP: Practical Fault-Tolerance for Itinerant Computations. In *ICDCS'99: Proceedings of the 19th IEEE International Conference on Distributed Computation* 1999.

[4] S.R. Maden, M. J. Franklin, J.M Hellerstein, and W. Tong. TAG: a Tiny AGregation service for ad-hoc sensor networks. In *OSDI'02: Proceedings of the fifth ACM Symposium on Operating System Design and Implementation*, pages 56-67. ACM Press, 2002.

[5] K. Marzullo. Tolerating failures of continuous-valued sensors. In *ACM Trans. Comput. Syst.*, 8(4), pages 284-304. ACM Press, 1990.

[6] G. Pottie and W. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51-58.May 2000.

[7] T. Liu, C. M. Sadler, P. Zhang and M. Martonosi. Implementing software on resource-constrained mobile sensors: experiences with Impala and ZebraNet. In *MobiSYS '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 256–269, ACM Press.

[8] D.W. Welch, G.W. Boehlert, and B.R., Ward. POST-the Pacific Ocean Salmon Tracking Project. *Oceanologica Acta.* 25, pages 243-253.

[9] C. Delporte, H. Fauconnier and R. Guerraoui. Escorting the Walk of the Emperor. Technical Report. EPFL. To appear.