Crash-resilient cooperative mobile robots with asynchronous communications *

Rami Yared

JAIST, School of Information Science
Japan Advanced Institute of Science and Technology
Email: {r-yared}@jaist.ac.jp

Abstract

The paper discusses different approaches that ensure the liveness of collision prevention system of cooperative mobile robots with asynchronous communications, in presence of the crash of some robots. We compare different techniques and discuss the trade off between the strength of the properties of the failure detector, and the performance of the collision prevention system in terms of the number of requests cancelation.

Each robot in the system knows the composition of the group, and can communicate with all robots of the group.

1 Introduction

Many research in distributed systems study problems in which hosts are mobile and their physical location cannot be abstracted. While most efforts are still aimed at mobile ad hoc networks and sensor networks, there is also a gradual realization that cooperative robotics raises many interesting new challenges with respect to distributed systems, and particularly in relation to mobility. Indeed, unlike traditional distributed systems and even more so than ad hoc or sensor networks, mobility becomes an essential part of the problems to address.

Many interesting applications are envisioned that rely on groups of cooperating mobile robots. Tasks may be inherently too complex (or impossible) for a single robot to accomplish, or performance benefits can be gained from using multiple robots [2].

As a simple illustration, consider the following example. A distributed system composed of cooperative autonomous mobile robots cultivating a garden. Cultivating a garden requires that mobile robots move in all directions in the garden sharing the same geographical space. A robot has no prior knowledge about neither the paths of other robots, nor their speeds.

Context and problem We consider that the robots have the ability to communicate wirelessly and also that they can query their own position according a common referential, as given by a positioning system. However, the robots do not have the ability to detect each other's position in the environment, and they are not synchronized. In addition, communication delays are unpredictable, and actual robot motion speed is unknown.

A robot is based on its local motion planning facility to compute a path between the current location and the goal. This path avoids the collisions with fixed known obstacles.[1]

Problem The robots are moving in different directions sharing the physical space, thus collisions between mobile robots can possibly occur. It is very important to focus on the problem of *preventing* collisions between mobile robots, and thus ensure a *safe* motion, such that no two robots ever collide regardless of the tasks of the robots. The *safety* of the system must be guaranteed independently of timeliness properties of the system, and even in the event of unexpected timing errors in the environment. However, the *performance* of the system may possibly degrade as the result of badly unstable network characteristics or erratic robot speed.

Thus, it is essential to provide a safe motion platform on which mobile robots can rely for their motion. This platform guarantees that no collision between robots can occur, regardless of the timeliness guarantees of the underlying environment.

Contribution We focus on the collision prevention problem in presence of crash of some robots. We propose fault-tolerant approaches that enable our collision prevention protocol to handle the crash of some robots, and to ensure the liveness of the system of robots in presence of crash.

*Work supported by MEXT Grant-in-Aid for Young Scientists (A) (Nr. 18680007).

[1] The robots are the only moving entities in the considered applications.

Structure of the paper The rest of the paper is organized as follows. Section. 2 presents the related work. Section. 3 describes the system model and terminology. In Section. 4, we propose several crash-resilient approaches that enable a collision prevention system to handle the crash of some robots, and maintain the liveness of the robotic system. Section. 5 concludes the paper.

2 Related work

Martins et al. [7] demonstrated the avoidance of collisions between three cars, elaborated in the CORTEX project. They rely on the coexistence of two networks, as defined in the Timely Computing Base of Veríssimo and Casimiro [13]. One network, the payload network, is asynchronous and carries the information payload of the application. The second network, the control network or wormhole, enforces strict real-time guarantees and is used sparingly by the protocol.

Their approach differs from ours in several aspects. The major difference with our protocol presented in this paper is that our protocol tolerate the crash of robots in the system, while the approach in [7] is not fault-tolerant.

Nett et al. [10] presented a protocol for cooperative mobile systems in real-time applications. They considered a traffic control application in which a group of mobile robots share a specified predetermined space. Communication is done through WiFi (802.11) with a base station. All robots can communicate directly with each other, and the system assumes the existence of a known upper bound on communication delays. Needless to say that the protocol relies on the strict enforcement of timing assumptions, and it des not tolerate the crash of robots.

We have recently developed a simpler version [15] which is a time-free collision prevention platform for a group of asynchronous cooperative mobile robots. Our protocol presented in [15] guarantees that no collision occurs between robots, independently of timeliness properties of the system, and even in the presence of timing failures in the environment. However, our protocol in [15] does not consider the crash of robots.

In [14], we have presented a time-free collision prevention protocol which relies on ad hoc communication and supports dynamic groups of mobile robots. In a dynamic group of mobile robots the total composition of the system, of which robots have a partial knowledge, can change dynamically, our protocol presented in [14] does not consider the crash of robots.

Clark et al. [4] presented a collision avoidance based on motion planning framework by combining centralized with decentralized motion planning techniques. When robots become within communication range of each other, they establish dynamically a network. Their protocol ensures that at any time, robots in each network share a common world model by accessing sensing information of all other robots in the same network. Robots avoid collisions by re-planning their paths. Their approach relies on proper timing of communications and robots speed.

Jager et al. [5] presented a decentralized collision avoidance mechanism based on motion coordination between robots. When the distance between two robots goes below a certain threshold, they exchange information about their respective planned paths and determine whether there is a risk of collision. If a collision is possible, then they monitor each other's movements and may change their speed to avoid the collision. The approach is highly dependant on the proper timing of communication and, to some extent, to the proper control of robots speed.

Similarly, Azarm et al. [1] presented an on line distributed motion planning. When a conflict is detected between two robots, they exchange their information and determine their respective priorities. The robot with the highest priority keeps its original path while other robots must re-plan their motion.

The problem of robots collision avoidance has also been handled using different strategies which are sensor-based motion planning methods. The detailed information about motion planning strategies is inspired from [8].

Minguez et al. [8] compute collision-free motion for a robot operating in dynamic and unknown scenarios. Motion planning algorithms compute a collision-free path between a robot's location and its destination. Robots involve sensing directly within the motion planning by sensing periodically at a high rate.

Some of these approaches (e.g., [9]) apply mathematical equations to the sensory information and the solutions are transformed into motion commands. Another group of methods (e.g., [12]) computes a set of suitable motion commands to select one command based on navigation strategy. Finally, other methods (e.g., [8]) compute a high-level information description (e.g., entities near obstacles, areas of free space) from the sensory information, and then apply several techniques simplifying the difficulty of the navigation to obtain a motion command in complex scenarios.

Sensor-based approaches depend on real-time guarantees for processing the sensory information. Furthermore, the information provided by proximity sensors is unreliable and much more limited in range than most wireless network interfaces.

3 System model and terminology

3.1 System model

We consider a system of n mobile robots $S = \{r_1, \ldots, r_n\}$, moving in a two dimensional plane. Each

robot has a unique identifier. The total composition of the system is known to each robot.

Robots have access to a global positioning machine that, when queried by a robot r_i, returns r_i's position with a bounded error ε_{gps}.

The robots communicate using wireless communication such that a robot r_i can communicate with all robots of the system. Communications assume retransmissions mechanisms such that communication channels are reliable.

The system is asynchronous in the sense that there is no bound on communication delays neither between the robots nor between a robot and the positioning machine, processing speed and on robots speed of movement.

We consider that robot can fail by crash and that crash is permanent. A *correct* robot is defined as a robot that never crashes. A *faulty* robot is a robot that might crashes. A robot r_i is provided with a failure detector. We assume that the majority of the robots are correct. The number of faulty robots f ($f < \lceil \frac{n}{2} \rceil$), where n is the total number of robots.

3.2 Failure detectors

There exists several classes of failure detectors, depending on how unreliable the information provided by the failure detector can be. Classes are defined by two properties, called *completeness* and *accuracy*.

We distinguish four classes of failure detectors, $\mathcal{P}$ (perfect), $\Diamond\mathcal{P}$ (eventually perfect), $\mathcal{S}$ (strong), and $\Diamond\mathcal{S}$ (eventually strong). The four classes have the same property of completeness, but differs in their accuracy property. [3].

- STRONG COMPLETENESS Eventually every faulty process is permanently suspected by all correct processes.
- STRONG ACCURACY No process is suspected before it crashes. [class $\mathcal{P}$]
- EVENTUAL STRONG ACCURACY There is a time after which correct processes are not suspected by any correct process. [class $\Diamond\mathcal{P}$]
- WEAK ACCURACY Some process is never suspected. [class $\mathcal{S}$]
- EVENTUAL WEAK ACCURACY There is a time after which some correct process is never suspected by any correct process. [class $\Diamond\mathcal{S}$]

4 Crash-resilient collision prevention approaches

Crash-free collision prevention We have presented a crash-free collision prevention protocol in [15]. The idea of the crash-free protocol is explained intuitively as follows. It is essentially a mutual exclusion on geographical zones. Our crash-free protocol consists of a distributed path reservation system, such that a robot must reserve a zone before it moves. When a robot reserves a zone, it can move *safely* inside the zone.

All robots run the same protocol. When a robot wants to move along a given chunk of a path, it must reserve the zone that surrounds this chunk. When this zone is reserved, the robot moves along the chunk. Once the robot reaches the end of the chunk, it releases the zone except for the area that the robot occupies. When moving along a path, the robot repeats this procedure for each chunk along the path.

The protocol is based on the state machine approach of Lamport [6]. Briefly, the idea is as follows. Each robot maintains a copy of the reservation queue and a protocol ensures that all requests/releases are delivered in the same sequence. With all replicas starting in the same state, they evolve consistently with no need for further synchronization.

The wait-for graph is a directed acyclic graph that represents the wait-for relations between robots. A node represents a robot and a directed edge represents a wait-for relation between the corresponding robots.

System liveness in presence of robots crash In the crash-free model, the liveness property of the system is guaranteed since a robot eventually release the zone that has reserved. However, in the presence of crash of some robots, the protocol cannot guarantee the liveness of the system. If a robot r_j has crashed while it reserves Z_j, then a robot r_i that waits for r_j starves because the requested zone Z_i intersects with Z_j which would be kept infinitely under the reservation of r_j. Since the system is asynchronous, then it is impossible for the robot r_i to distinguish whether r_j is very slow or r_j has actually crashed. Therefore, the robot r_i that waits for the crashed robot r_j is blocked. After some period of time, if a robot r_k requests a zone Z_k that intersects with Z_i or with Z_j, then r_k would be also blocked and so on. This *Snowball* effect may eventually invoke all the robots in the system, and thus the whole system will be blocked.

We provide techniques that guarantee the liveness of the system in presence of crash of some robots. We present three algorithms based on different classes of failure detectors. At first, we present a solution relies on a perfect failure detector (class $\mathcal{P}$), a second algorithm relies on an Eventual Strong Accurate failure detector (class $\Diamond\mathcal{P}$) and a third algorithm relies on an Eventual Weak Accurate failure detector (class $\Diamond\mathcal{S}$).

4.1 Approach with a perfect failure detector (class $\mathcal{P}$)

A perfect failure detector of class $\mathcal{P}$ has the Strong Accuracy property, such that no process is suspected before it crashes. The intuition of this approach is as follows. A robot r_i that waits for a robot r_j. If the failure detector FD_i of r_i suspects that r_j has crashed, then r_i handles r_j as a fixed obstacle and r_i considers that Z_j is released. Hence r_i removes the node that represents r_j and its related edges. When r_i does not wait for any robot, then r_i is granted Z_i.

4.2 Approach with an Eventual Strong Accurate failure detector (class $\Diamond\mathcal{P}$)

A failure detector of the class $\Diamond\mathcal{P}$ has the property that eventually correct processes are not suspected by any correct process. The intuition of this approach is as follows. A request of a robot r_i is preempted if it is considered as a crashed robot by the majority of robots, and only if r_i is not granted Z_i. When the request (r_i, Z_i) is preempted, r_i restarts a new request of Z_i later, if r_i has not really crashed.

If the robot r_i is considered as a crashed robot by the majority of robots in the system after r_i is granted Z_i, then Z_i is considered as a blocked zone and it is granted to r_i. If r_i has not really crashed, then r_i eventually releases Z_i. If r_i has really crashed, then Z_i remains a blocked zone, and is granted always to r_i.

In the preemptive protocol, the suspicion of a robot and granting a zone to a robot occur using Total Order Broadcast, to ensure that all the robot in the system decide consistently that a robot r_i is suspected by the majority of the robots and whether r_i is granted Z_i or not, after being suspected by the majority of robots in the system.

According to the Eventual Strong Accuracy, all correct robots are not suspected by any correct robot, so eventually a correct robot r_i will be granted Z_i because eventually the request (r_i, Z_i) is not preempted. Therefore, the liveness of the system is ensured.

4.3 Approach with an Eventual Weak Accurate failure detector (class $\Diamond\mathcal{S}$)

A failure detector of the class $\Diamond\mathcal{S}$ has the property that eventually some correct process is never suspected by any correct process. The intuition of the non preemptive protocol using a $\Diamond\mathcal{S}$ failure detector is the following. If a robot r_j waits directly for a robot r_i (because Z_j intersects with Z_i), and if r_j suspects that r_i has crashed, then r_j cancels its request (r_j, Z_j) and requests an alternative zone that does not intersect with Z_i. In this approach a request of a robot is not preempted if the robot is suspected to be crashed.

4.4 Other approach

We discuss an approach based on leases and synchronized clocks to tolerate the crash of robots and to guarantee the liveness of the system. A lease is an abstraction of a contract, such that its holder is given a tenure for a limited period of time. The time period of a lease is called *term* of the lease.

The intuition of this approach is the following. Each robot is provided by a local physical clock, such that a robot r_i can observe time using its clock. The values of the clock of r_i is related to real-time and can be read and written by r_i. The clocks of the robots have a bounded value of drift, so a clock synchronization is required to achieve and maintain a known bounded drift between the clocks.

A lease is given to a robot r_j such that the term of the lease equals to the time required by r_j to move along its requested chunk of path and to release Z_j. The robot r_j reads its local clock to detect the expiry of its lease taking in consideration the maximal drift between the clocks. If r_j does not complete moving along its requested chunk of path, then it stops within a time delay Δ and behaves as if it were a fix obstacle. (This requires bounds on processing and movement speed of robots to stop within a delay Δ). r_j releases Z_j and performs a new request to complete moving along its previous chunk of path.

When the lease of r_j expires according to the local clock of the robot r_i, then r_i waits for a time delay Δ in addition to the value of the maximal drift between the clocks. After that, r_i considers the zone Z_j as a released zone.

This approach is simple, however it relies on clock synchronization. Deterministic software clock synchronization algorithms requires a known bound on communication message delay. There exist some hardware techniques for clock synchronization which require a dedicated network to connect the physical clocks (other than the network of the application). (Shin and Ramanathan [11]).

4.5 Comparison and discussion

The described algorithms guarantee the liveness of the system. There is a trade off between the strength of the failure detector and the performance of the system in terms of the number of requests cancelation. Request cancelation implies that a robot requests an alternative zone, and this may cause inconvenience for a robotic system, particularly when the density of robots is high. A solution using a failure detector of class $\Diamond\mathcal{P}$ generates less request cancelation than a solution relies on a $\Diamond\mathcal{S}$ failure detector, although the classes of these failure detectors differ in the Accuracy property (some correct process are not suspected versus all correct process are not suspected). A solution relies on a perfect failure detector (class $\mathcal{P}$) is ideal, but a perfect fail-

ure detector relies on strong assumptions that requires the knowledge of a fix upper bound on communication delays. Also, a solution relies on leases with synchronized clocks requires the knowledge of a fix upper bound on communication delays.

5 Conclusion and future directions

We have presented different algorithms and techniques to ensure the liveness of a collision prevention system of cooperative mobile robots with asynchronous communications, in presence of crash. We provided different approaches each of which uses a specific class of failure detector. We discussed the trade off between the strength of the assumptions and the properties of the failure detector and the performance of the collision prevention protocol in terms of the number of requests cancellation.

We presented also an approach based on leases with synchronized clocks, which is equivalent to a solution based on a perfect failure detector. In both cases, the knowledge of a fix upper bound on communication delays is required.

A quantitative study of the trade off between the different approaches is in progress using simulations in different scenarios and models. In particular, the trade off between a $\Diamond\mathcal{P}$ based approach and another based on $\Diamond\mathcal{S}$ failure detector.

In the future we intend to further investigate and optimize the performance of the system, according to different parameters.

References

[1] K. Azarm and G. Schmidt. Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation. In *Proc. IEEE Int'l Conf. Robotic and Automation, ICRA'97.*, 1997.

[2] Y. Cao, A. Fukunaga, and A. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–27, 1997.

[3] T. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM: Journal of the ACM*, 43(2):225–267, 1996.

[4] C. Clark, S. Rock, and J.-C. Latombe. Motion planning for multiple mobile robots using dynamic networks. In *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA'03)*, Taipei, Taiwan, Sept. 2003.

[5] M. Jager and B. Nebel. Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots. In *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS'01)*, 2001.

[6] L. Lamport. The implementation of reliable distributed multiprocess systems. *Computer Networks*, 2:95–114, 1978.

[7] P. Martins, P. Sousa, A. Casimiro, and P. Veríssimo. A new programming model for dependable adaptive real-time applications. *IEEE Distributed Systems Online*, 6(5), May 2005.

[8] J. Minguez and L. Montano. Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios. *IEEE Trans. on Robotics and Automation*, 20(1):45–59, 2004.

[9] L. Montano and J. Asensio. Real-time robot navigation in unstructured environments using a 3D laser rangefinder. In *IEEE/RSJ Conf. on Intelligent Robots and Systems*, pages 526–532, 1997.

[10] E. Nett and S. Schemmer. Reliable real-time communication in cooperative mobile applications. *IEEE Trans. Computers*, 52(2):166–180, 2003.

[11] K. Shin and P. Ramanathan. Transmission delays in hardware clock synchronization. *IEEE Trans. Computers*, 37(11):1465–1467, Nov. 1988.

[12] R. Simmons. The curvature-velocity method for local obstacle avoidance. In *IEEE/RSJ Conf. on Intelligent Robots and Systems*, pages 3375–3382, 1996.

[13] P. Veríssimo. Uncertainty and predictability: Can they be reconciled? In *Future Directions in Distributed Computing*, pages 108–113, 2003.

[14] R. Yared, J. Cartigny, X. Défago, and M. Wiesmann. Locality-preserving distributed path reservation protocol for asynchronous cooperative mobile robots. In *8th IEEE Intl. Symp. on Autonomous Decentralized Systems ISADS'07*, 2007.

[15] R. Yared, X. Défago, and M. Wiesmann. Collision prevention using group communication for asynchronous cooperative mobile robots. In *21st IEEE Intl. Conf. on Advanced Information Networking and Applications AINA'07*, 2007.