The Power of Lights: Synchronizing Asynchronous Robots using Visible Bits

Shantanu Das*, Paola Flocchini†, Giuseppe Prencipe‡, Nicola Santoro§, Masafumi Yamashita¶
*BGU & Technion-Israel Institute of Technology, shantanu@tx.technion.ac.il
†EECS, University of Ottawa, flocchin@site.uottawa.ca
‡Dipartimento di Informatica, Università di Pisa, prencipe@di.unipi.it
§School of Computer Science, Carleton University, santoro@scs.carleton.ca
¶ Department of Informatics, Kyushu University, mak@inf.kyushu-u.ac.jp

***Abstract*—In this paper we study the power of using lights, i.e. visible external memory, for distributed computation by autonomous robots moving in Look-Compute-Move (LCM) cycles. With respect to the LCM cycles, the most common models studied in the literature are the *fully-synchronous* (FSYNC), the *semi-synchronous* (SSYNC), and the *asynchronous* (ASYNC). In this paper we introduce in the ASYNC model, the weakest of the three, the availability of visible external memory: each robot is equipped with a light bulb that is visible to all other robots, and that can display a constant numbers of different colors; the colors are persistent, that is they are not automatically reset at the end of each cycle.**

We first study the relationship between ASYNC with visible bits and SSYNC. We prove hat asynchronous robots, when equipped with a constant number of colors, are strictly more powerful than traditional semi-synchronous robots. We also show that, when enhanced with visible lights, the difference between asynchrony and semi-synchrony disappears; this result must be contrasted with the strict dominance ASYNC <SSYNC between the models without lights.

We then study the relationship between ASYNC with visible bits and FSYNC. We prove that asynchronous robots with a constant number of visible bits, if they can remember a single snapshot, are strictly more powerful than fully-synchronous robots. This is to be contrasted with the fact that, without lights, ASYNC robots are not even as powerful as SSYNC, even if they remember an unlimited number of previous snapshots. These results demonstrate the power of using visible external memory for distributed computation with autonomous robots. In particular, asynchrony can be overcome with the power of lights.

I. Introduction

A. The Framework

The computational capabilities of a team of autonomous mobile entities, usually called *robots* or *agents*, have been the object of extensive research in a variety of fields. In particular, in the last few years, a large amount of work in distributed computing has been devoted to the study of models of autonomous mobile robots operating in *Look-Compute-Move* (LCM) cycles. During a cycle, a robot obtains a snapshot of the environment (*Look*); executes the protocol, the same for all robots, using the snapshot as an input (*Compute*); and moves towards the computed destination, if any (*Move*). After each cycle, a robot may be inactive for some time. The main goal of the research efforts has been to understand the relationships between the capabilities of the robots and their power to solve common tasks. With respect to the LCM cycles, the most common models used in these studies are the *fully synchronous* (FSYNC), the *semi-synchronous* (SSYNC), and the *asynchronous* (ASYNC). In the *asynchronous* (ASYNC) model [17], the robots are activated independently, and the duration of each cycle is finite but unpredictable. As a result, the robots do not have a common notion of time, robots can be seen while moving, and computations can be made based on obsolete observations. On the opposite side of the spectrum, in the *fully synchronous* (FSYNC) model [24], the activations of the robots can be logically divided into global rounds; in each round, all the robots are activated, obtain the same snapshot, compute and perform their move. As a result, no robot can be seen while moving, and no information is out-of-date. Note that, this is computationally equivalent to a fully synchronized system in which all robots are activated simultaneously and all operations are instantaneous. The *semi-synchronous* (SSYNC) model is like the fully synchronous model where however not all robots are necessarily activated in each round [24]. Since there are problems that can be solved in FSYNC but not in SSYNC (e.g. [24]), and problems that can be solved in SSYNC but not in ASYNC (e.g. [22]), the relationship between the computational power of the models is strict: FSYNC $>$ SSYNC $>$ ASYNC.

In this paper we introduce in the ASYNC model, the weakest of the three, a simple form of direct and explicit communication: each robot is equipped with a light bulb that is visible to all other robots, and that can display a constant numbers of different colors; the colors are persistent, that is they are not automatically

reset at the end of each cycle. In other words, we equip the robots with a constant number of visible and persistent bits of memory. We denote this model as $\text{ASYNC}^{O(1)}$, and study the computational power of robots so endowed, with respect to the traditional models SSYNC and FSYNC.

Peleg [21] first suggested the use of external lights to enhance the capabilities of mobile robots. To our knowledge, no results have been presented so far on a set of mobile robots that have available some kind of visible memory (such as the visible lights used in this paper); the only reference to the use of visible lights can be found in [14], which provided the earliest indication that incorporating in the robot model some simple means of signaling might positively affect the power of the team. This paper explores this research direction.

B. Main Contributions

In this paper we introduce the $\text{ASYNC}^{O(1)}$ model and show that with just six colors it is at least as powerful as the traditional SSYNC. We do so constructively: we present a ASYNC^{6} protocol that, for any given SSYNC protocol $\mathcal{P}$, produces a semi-synchronous execution of $\mathcal{P}$. We then solve, in ASYNC with four colors, the gathering of two oblivious robots, a problem that is not solvable in SSYNC. In other words, we prove that asynchrony with a constant number of colors is strictly more powerful than semi-synchrony: $\text{ASYNC}^{O(1)} > SSYNC$.

We also show that, when enhanced with visible lights, *the difference between asynchrony and semi-synchrony disappears*; in fact we prove that $\text{ASYNC}^{O(1)} \equiv \text{SSYNC}^{O(1)}$. This result must be contrasted with the strict dominance in absence of lights: ASYNC $<$ SSYNC.

Next, we investigate whether ASYNC robots with lights can match the power of FSYNC. To this end, we consider augmenting ASYNC robots with internal (i.e., not visible) persistent memory of k previous snapshots, a model denoted as ASYNC_k. This enhancement has been already considered in the literature (e.g., see [6], [24]). There are however problems solvable in FSYNC that are not solvable in ASYNC_∞, i.e., even if the asynchronous robots are endowed with enough memory to store an unbounded number of snapshots [22]. We show that with the simultaneous use of both external lights and internal snapshots, ASYNC becomes at least as powerful as FSYNC. In fact, we demonstrate that if ASYNC robots with only three colors can remember a single snapshot, they can solve any problem solvable in FSYNC. This proof is also constructive: we present a ASYNC_1^3 protocol that allows to produce a fully synchronous execution of any given protocol. On the other hand, we show the existence of problems solvable with ASYNC robots having three colors and one snapshot, but not solvable in FSYNC, without these additional powers. In other words, we prove that asynchrony with a constant number of colors and a single snapshot is strictly more powerful than full synchrony: $\text{ASYNC}_1^{O(1)} >$FSYNC.

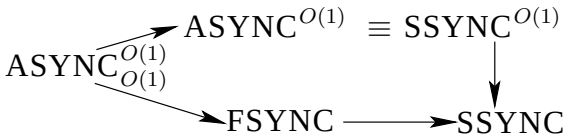

Figure 1. Relationship between models.

These results, summarized in Figure 1, demonstrate the power of using lights i.e. visible external memory for distributed computation with autonomous robots. In particular, they show that asynchrony can be overcome with the power of lights.

C. Related work

The main effort in the study of autonomous mobile robots has been to understand their limitations and their power for solving basic coordination tasks. In a seminal work [24], the authors have compiled a comprehensive study of the computational capabilities of oblivious robots in FSYNC and in SSYNC, characterizing the *Arbitrary Pattern Formation Problem*, where the robots are required to form a given pattern. This problem has been also studied in the ASYNC model, introduced in [15]; in particular, in [17] the solvability of the problem has been characterized based on the various levels of agreement on a common coordinate system; another interesting study on pattern formation in the semi-synchronous model can be found in [25], which shows that oblivious robots can form any pattern that non-oblivious robots can form. A number of studies have been devoted to the analysis of a particular pattern formation problem, the *circle*; here, the robots are required to place themselves uniformly on the border of a circle (e.g., [10], [11]).

Another basic coordination problems that has been studied in the literature in all models is the *gathering*, where the robots are required to meet in a point of the plane not fixed in advance. In particular, in [24] the problem has been tackled in the semi-synchronous model, with oblivious robots having unlimited visibility; in the same model, a study with limited visibility has been presented in [2]. The gathering problem has also been studied in the asynchronous model, in both the unlimited ([7]) and limited visibility setting ([16]). One interesting result is that, in both the semi-synchronous and asynchronous model, the problem

is not solvable when $n = 2$ (if we assume that the robots cannot *bump* into each other), as shown in [24]. Also, in all the available solutions, an implicit necessary condition as that the robots have the ability of *multiplicity detection*, that is a robot can dustinguish whether a given position on the plane is occupied by one or more than one robot. In [6] the problem is solved dropping this condition; however, the solution requires an unbounded amount of memory available to each robot.

Other important problems studied for these teams of robots include *scattering* (e.g., [3], [13]), where the robots are required to scatter on the plane where they operate; leader election, where the robots have to elect one of them as the leader of the team (e.g., [12]); and *flocking*, where the robots have to follow one of the robots while keeping a formation, like a flock of birds (e.g., [5], [18]). Also studied has been the problem of communicating the local coordinate systems (e.g., [4]). Finally, studies have also been conducted on the fault-tolerance of a distributed system composed by a set of autonomous mobile robots, such as in [1], [9], [19], [20], [23].

As mentioned earlier, the use of external signals or lights to enhance the capabilities of mobile robots was first suggested by Peleg [21]. The use of visible identities has been investigated in [8]. The use of visible lights for signaling has been proposed in [14] in the context of partitioning a swarm of anonymous mobile robots.

II. The model

The system is composed of a team of mobile entities, called *robots*, each modelled as a computational unit provided with its own local memory and capable of performing local computations.

The robots are placed in a spatial universe, here assumed to be $\mathbb{R}^2$, and they are viewed as points in $\mathbb{R}^2$. Each robot has its own local coordinate system; however, the local coordinate systems of the robots might not be consistent with each other. A robot is endowed with sensorial capabilities and it observes the world by activating its sensors, which return a snapshot of the positions of all other robots with respect to its local coordinate system.

The robots are *identical*: they are indistinguishable by their appearance and they execute the same protocol. The robots are *autonomous*, without a central control. The robots are *silent*, in the sense that they have no means of direct communication (e.g., wireless) of information to other robots.

Each robot is endowed with motorial capabilities, and can freely move in the plane. A move may end before the robot reaches its destination, e.g. because of limits to its motion energy. The distance traveled in a move is neither infinite nor infinitesimally small. More precisely, there exists an (arbitrarily small) constant $\delta > 0$ such that if the destination point is closer than δ, the robot will reach it; otherwise, it will move towards it a distance of at least δ. Note that, without this assumption, an adversary would make it impossible for any robot to ever reach its destination, following a classical Zenonian argument. The quantity δ might not be known to the robots.

At any point in time, a robot is either *active* or *inactive*. When *active*, a robot r executes a *Look-Compute-Move* (LCM) cycle performing the following three operations, each in a different state:

(i) **Look:** The robot observes the world by activating its sensor, which returns a snapshot of the positions of all robots with respect to its own coordinate system (since robots are viewed as points, their positions in the plane are just the set of their coordinates).

(ii) **Compute:** The robot executes its algorithm, using the snapshot as input. The result of the computation is a destination point.

(iii) **Move:** The robot moves towards the computed destination; if the destination is the current location, the robot stays still (performs a *null movement*).

When *inactive*, a robot is idle. All robots are initially inactive. The amount of time to complete a cycle is assumed to be finite, and the *Look* is assumed to be instantaneous.

The robots may or may not have distinct identities; if they are without identifiers that can be used during the computation they are said to be *anonymous*. The robots may or may not have a finite but *persistent* memory, that is memory whose content is preserved from one cycle to the next; they are said to be *oblivious* if they do not, in which case they start each cycle without any information on the past.

We denote by $\mathcal{R}$ the set of all teams of robots satisfying the above assumptions (i.e., they are identical, silent, autonomous, and operate in LCM cycles), and denote by $R \in \mathcal{R}$ a team of robots having identical capabilities (e.g., persistent storage, anonymity, etc.). We will specifically denote by $\mathcal{R}_o \subset \mathcal{R}$ the set of all teams of oblivious robots.

With respect to the activation schedule of the robots and their *Look-Compute-Move* cycle, the most common *models* are the fully-synchronous, the semi-synchronous, and the asynchronous. In the *asynchronous* (ASYNC) model, the robots are activated independently, and the duration of each *Compute*, *Move* and inactivity is finite but unpredictable. As a result, the robots do not have a common notion of

time, robots can be seen while moving, and computations can be made based on obsolete observations. On the opposite side of the spectrum, in the *fully-synchronous* (FSYNC) model, the activations of all robots can be logically divided into global rounds; in each round, the robots are all activated, obtain the same snapshot, compute and perform their move. Note that this is computationally equivalent to a fully synchronized system in which all robots are activated simultaneously and all operations are instantaneous. The *semi-synchronous* (SSYNC) model is like the fully-synchronous model where however not all robots are necessarily activated in each round. Based on the fairness of the activation scheduler, sub-models can be obviously defined.

Given a model X and a team of robots $R \in \mathcal{R}$, let $Task(X, R)$ denote the set of problems solvable by R in X. Given two models X and Y, we say that X is computationally not less powerful than Y, denoted by $X \geq Y$ if $\forall R \in \mathcal{R}, Task(Y, R) \subseteq Task(X, R)$. If $X \geq Y$ and $\exists R \in \mathcal{R}, Task(X, R) \setminus Task(Y, R) \neq \emptyset$, we say that X is computationally more powerful than Y, denoted by $X > Y$. If $X \geq Y$ and $Y \geq X$, X and Y are said to be computationally equivalent, denoted by $X \equiv Y$. For simplicity of notation, let $\mathcal{A}(R)$, $\mathcal{S}(R)$, and $\mathcal{F}(R)$ denote $Task(\text{ASYNC}, R)$, $Task(\text{SSYNC}, R)$, and $Task(\text{FSYNC}, R)$, respectively. Trivially we have:

$$\text{FSYNC} \geq \text{SSYNC} \geq \text{ASYNC}. \tag{1}$$

There are problems that are solvable in SSYNC but not in ASYNC (e.g. [22]); that is,

$$\exists R \in \mathcal{R}, \mathcal{S}(R) \setminus \mathcal{A}(R) \neq \emptyset \tag{2}$$

Similarly, there are problems that are solvable in FSYNC but not in SSYNC (e.g. [24]); that is,

$$\exists R \in \mathcal{R}, \mathcal{F}(R) \setminus \mathcal{S}(R) \neq \emptyset \tag{3}$$

Thus, from (1), (2) and (3), we have the following relationship between the computational power of the three basic models:

$$\text{FSYNC} > \text{SSYNC} > \text{ASYNC}. \tag{4}$$

In this paper we augment the ASYNC model by providing some additional capabilities to the robots. Each robot in addition to its capabilities, has a light bulb that is visible to all the robots when they perform their *Look* operation. The light associated with a robot can assume different colors (from a finite set) and can be updated by a robot during the *Compute* operation. The light is persistent; i.e., while the robots might be oblivious forgetting all other information from previous cycles, their lights are not automatically turned off at the end of a cycle. Thus, it constitutes a form of external persistent memory.

The second capability we consider is the ability to remember a constant number of snapshots from previous cycles. More precisely, for some integer constant $j > 0$, the robot is allowed to store in its internal memory at most j snapshots from previous *Look* operations (the robot may choose which snapshots it stores).

We denote these two additional abilities using a subscript and a superscript representing the number of snapshots and the number of external colors respectively; that is, ASYNC^i_j denotes the ASYNC model when each robot is augmented by a visible light with $i > 0$ colors and by a persistent memory of $j > 0$ past snapshots, and $\mathcal{A}^j_i(R)$ denotes the class of problems solvable in this model by $R \in \mathcal{R}$.

III. Asynchrony with Visible Lights versus Semi-Synchrony

A. $\text{ASYNC}^{O(1)}$ *is at least as powerful as* SSYNC

In this section we show that asychronous systems equipped with a light colorable with $O(1)$ colors are at least as powerful as semi-synchronous systems without lights. More precisely, we have:

Theorem III.1. $\forall R \in \mathcal{R}$, $\mathcal{S}(R) \subseteq \mathcal{A}^6(R)$.

The proof is constructive: we present a ASYNC^6 protocol SIM that produces a semi-synchronous execution of any SSYNC protocol $\mathcal{P}$.

The lights used by SIM can have six colors: `T`(rying), `M`(oving), `S`(topped), `F`(inished), `W`(aiting), `N`(ext). At the beginning, all lights are set to `T`. The protocol is a sequence of Mega-Cycles, each of which starts with all robots trying to execute protocol $\mathcal{P}$ (color `T`) and ends with all robots finishing the Mega-Cycle having executed $\mathcal{P}$ once (color `F`). All robots with light `F` then eventually turn their lights to `T`; when this process is completed, a new Mega-Cycle starts.

During a Mega-Cycle every robot executes $\mathcal{P}$ once. Each Mega-Cycle is composed of a sequence of *stages*: at each stage, some robots are allowed to execute $\mathcal{P}$, and protocol SIM ensures that they have the *same* view of the world (i.e., they observed the same snapshot). Each stage starts when a robot observes all the other robots with their light either `T` or `S` (and starts the execution of $\mathcal{P}$ eventually turning their light `M`), and it ends when at least a robot has changed light from `M` to `S` and all other robots are again in `T` or `S`. In particular, at the beginning, all the robots that during their *Look* phase see only robots with light `T` are allowed to enter the first stage by turning their own light to `M` before executing $\mathcal{P}$. Any other robot with

State *Look*

Take the snapshot of the positions of the robots, that returns for all robots $r \in R$:

- `Pos[r]`, the position on the plane of robot r (according to my coordinate system);
- `Light[r]`, the color of the light of robot r.

(**Note**: I am robot x)

State *Compute*

$p :=$ `Pos[x]`.
Case `Light[x]`:

- `T`
  If $\forall r \neq x$, `Light[r]` $\in \{$`T`, `S`$\}$ **Then**
  Execute $\mathcal{P}$.
  $p :=$ computed destination.
  `Light[x]` $:=$ `M`.
  If $(\exists r \neq x |$ `Light[r]` $\in \{$`M`$\})$ **Then**
  `Light[x]` $:=$ `W`.
- `M`
  If $\forall r \neq x$, `Light[r]` $\in \{$`M`, `W`, `S`$\}$ **Then**
  `Light[x]` $=$ `S`.
- `S`
  If $\forall r \neq x$, `Light[r]` $\in \{$`S`, `F`$\}$ **Then**
  `Light[x]` $=$ `F`.
- `F`
  If $\forall r \neq x$, `Light[r]` $\in \{$`F`, `T`$\}$ **Then**
  `Light[x]` $=$ `T`.
- `W`
  If $\forall r \neq x$, `Light[r]` $\in \{$`W`, `N`, `S`$\}$ **Then**
  `Light[x]` $=$ `N`.
- `N`
  If $\forall r \neq x$, `Light[r]` $\in \{$`S`, `N`, `T`$\}$ **Then**
  `Light[x]` $=$ `T`.

State *Move*

`Move(`p`)`.

Figure 2. Protocol SIM

light `T` that perform its *Look* operation when some robots' lights are `M` (and thus the robots are potentially moving), will be prevented from entering the current stage, loses its turn changes color to `W` and waits for another turn. The robots with light `M`, after executing $\mathcal{P}$, will turn their own lights to `S`.

Only after all robots that entered the current stage turn their light to `S`, the robots waiting for their turn, i.e., with light `W`, will be given a chance to enter the next stage. In particular, they will turn their lights to `N` and eventually to `T` to try to execute $\mathcal{P}$.

Essentially, the transition of lights from `T` to `W`, to `N`, and back to `T` corresponds to a queue where robots that failed to enter the current stage wait for their turn.

In the following, we will prove that Protocol SIM provides a fair and correct execution of any semi-synchronous protocol $\mathcal{P}$.

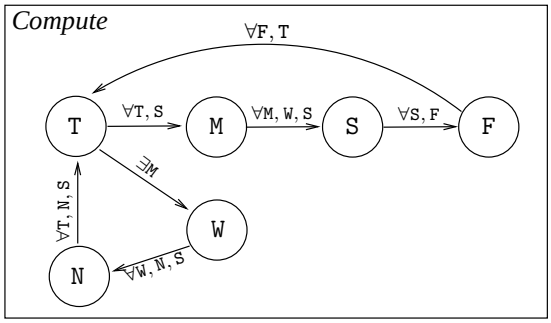

Figure 3. The transition diagram of the SIM protocol. The label in the nodes represent the value of the light of the executing robot (i.e. `Light[x]`). The label of an edge expresses a condition that must be satisfied on the light of all the other robots.

Lemma III.1. *In each Mega-Cycle, each robot executes $\mathcal{P}$ exactly once; in each stage, all robots executing $\mathcal{P}$ have the same snapshot.*

Proof: (Sketch) First observe that a robot can perform a non-null move in the *Move* phase only if it executed $\mathcal{P}$ in the *Compute* phase.

Let t be any time instant such that all robots are colored `T`, and let t_0 be the latest time $t_0 \leq t$ when they all became colored `T`. Let us call a robot *active* when it is not in `S`. By definition the configuration at time t_0 is the same as the one at time t.

By construction, it is easy to observe the following facts:

1) From time t_0, at least one robot in phase *Look* observes all other robots with light `T`, and thus executes $\mathcal{P}$ and turns its light to `M`.
2) Let $t_1 \geq t_0$ be the first time since (and including) t_0 when a robot turns its own light to `M`. That is all robots that *Look* from (and including) time t_0 to (and excluding) t_1 eventually turn their lights to `M`. Since a robot can perform a non-null move only after changing its own light to `M`, all robots with light `M` have observed the same configuration in their *Look*.
3) All robots with light set to `M` do not change color as long as some robots' light is `T`.
4) The robots that *Look* after time t_1 (inclusive) change color to `W` and keep that color until no robot has light colored `T` or `M`. That is, there is a time instant when all active robots have light either `M` or `W`, and at least one robot's light is `M` (see case 1 above).
5) Since all robots with light `W` did not execute $\mathcal{P}$ and thus did not move, and all robots with light `M` have observed the same configuration (the one at time t_1), then their movements are based on the configuration at time t_1.

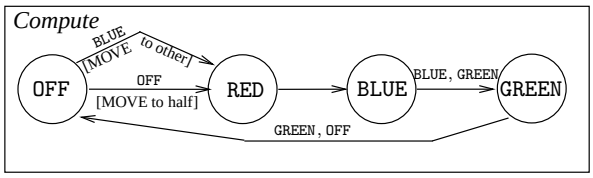

Figure 4. The transition diagram of the TWOGATHERLIGHT protocol.

6) All robots with light M eventually change their color to S (Case Light[x] = M in SIM). The robots with light W can turn their lights to N only after all robots in M have turned their lights to S. Eventually, they will all have light T. Note that the number of these robots is smaller than the robots that were in T at t_1.

From the above observations, we have that all robots will have light S at some time after t_0. Once all robot's lights are S, by construction, all robots turn their lights to F, and then eventually to T. Thus the next Mega-Cycle safely starts (with all lights being T). We also observe that, by construction, all robots can change light to M exactly once in a Mega-Cycle; this implies that every robot execute $\mathcal{P}$ exactly once. ■

From Lemma III.1, it follows that protocol SIM produces a semi-synchronous execution of any SSYNC input protocol $\mathcal{P}$. This, in turn, provides the proof of Theorem III.1.

B. ASYNC$^{O(1)}$ *is more powerful than* SSYNC

In the previous section we have shown that ASYNC$^{O(1)} \geq SSYNC$; that is, asynchronous robots, if endowed with $O(1)$ visible lights, are at least as powerful as if they were semi-synchronous.

In this section we show that there are problems that robots cannot solve without visible bits, even if they are semi-synchronous, but can be solved with $O(1)$ visible bits even if the robots are asynchronous; in particular, we show that for any team $R \in \mathcal{R}_o$ of oblivious robots $\mathcal{A}^{O(1)}(R) \setminus \mathcal{S}(R) \neq \emptyset$.

Consider the extensively investigated *gathering* or *rendezvous* problem GATHERING{K} of having k oblivious robots terminally gather in the same location, not previously known in advance. It is well known that the gathering of two oblivious robots cannot be guaranteed:

Lemma III.2 ([24])**.** $\forall R \in \mathcal{R}_o$, GATHERING$\{2\} \notin \mathcal{S}(R)$.

We now prove that two oblivious robots can gather, even if they are asynchronous when enabled with $O(1)$ visible lights; more precisely:

State *Look*

Let y be the other robot, and x be me;
Pos[x] :=my current position;
Pos[y] := position of the other robot;
Light[x] := Value of my light;
Light[y] := Value of the light of the other robot.

State *Compute*

If Gather **Then** STOP.
p := Pos[x].
Case Light[x]:
- OFF
  If Light[y] = OFF **Then**
    p := Half point between me and the other robot;
    Light[x] := RED.
  Else If Light[y] = BLUE **Then**
    p := Position of the other robot;
    Light[x] := RED.
- RED
  Light[x] := BLUE.
- BLUE
  If Light[y] $\in$ {BLUE, GREEN} **Then**
    Light[x] := GREEN.
- GREEN
  If Light[y] $\in$ {GREEN, OFF} **Then**
    Light[x] := OFF.

State *Move*

Move(p).

Figure 5. TWOGATHERLIGHT, a protocol for gathering two robots in ASYNC^4.

Theorem III.2. $\forall R \in \mathcal{R}$, GATHERING$\{2\} \in \mathcal{A}^4(R)$.

We prove the theorem constructively. Consider the protocol TWOGATHERLIGHT shown in Figure 5; let x and y be the two robots. The protocol uses four colors: OFF, RED, GREEN, and BLUE; initially the light of both x and y are set to OFF. The idea behind the protocol is as follows. If, after the beginning of the execution, both robots observe OFF as the color of the other robots' light, then they both try to reach the point halfway between the two robots. On the other hand if one robot begins execution earlier than the other it will move towards the midpoint, turning its light RED before moving. If the second robot now performs a *Look* operation, it will see the RED light and know that the other robot is potentially moving. In this case the second robot waits for the first robot to change colors from RED to BLUE. When the robot sees the BLUE light on the other robot it will try to move directly towards it. A robot with BLUE light waits until the second robot has also turned its light to BLUE. When both robots have BLUE lights, they turn their lights to GREEN to signal the end of one round of the algorithm (i.e. the

robots synchronize with each-other at the end of each round). Now the robots turn their lights to OFF to start the next round. As before we will use the term Mega-Cycle to refer to the time period during which the robot has its light exactly once in each color starting from OFF to RED, BLUE, GREEN and just before turning to OFF again.

Based on the rules of the algorithm TWOGATHERLIGHT, the following properties can be shown:

Lemma III.3. *In the execution of Algorithm* TWOGATHERLIGHT*:*

(i) *Unless the robots are already gathered, each Mega-Cycle completes in finite time. (i.e. there are no deadlocks)*

(ii) *If the distance between the robots, $dist(x, y) > 2\delta$, then after each complete Mega-Cycle this distance decreases by at least 2δ and the robots never cross each-other.*

(iii) *If $dist(x, y) \leq 2\delta$, then the robots gather during the next Mega-Cycle.*

Proof:

(i) An activated robot x stays in its current state if Light[x] is OFF and Light[y] is RED or GREEN, or if Light[x] is BLUE and Light[y] is RED or OFF. Note that none of this two conditions can hold for more than one activation cycle for each robot. Thus, there could be no deadlock.

(ii) It is easy to see that during a complete Mega-Cycle, each robot is guaranteed to move. Let robot x be the first robot to start moving during this Mega-Cycle. Let d be the distance between the robots at this time. The robot x may move only if Light[y] is OFF or BLUE during the *Look* operation. If Light[y] is BLUE, then robot y has already moved during this Mega-Cycle contradicting the assumption. Thus, Light[y] is OFF. Thus, robot x moves towards robot y by at least distance δ and at most a distance $d/2$ during this Mega-Cycle. The distance moved by robot y depends on what robot y sees during the *Look* operation performed when its light is OFF. If Light[x] was OFF at that time the robot y moves at most by $d/2$ towards robot x (so they do not cross). Otherwise if Light[x] was BLUE, then robot x has already finished moving and robot y moves a distance of d' which is at most the current distance between the robots. Thus, the robots do not cross. In both cases, the distance between the robots after the Mega-Cycle is at most $d - 2\delta$.

(iii) Let $d \leq 2\delta$ be the distance between the two robots and let p be the midpoint between the two locations. Without loss of generality, let x be the first robot to perform *Look* operation in the next Mega-Cycle. Robot x would decide to the move a distance $d/2 \leq \delta$ and thus it would eventually arrive at the location p. If robot y sees robots x when Light[x] is OFF, then robot x has not moved yet and the distance between the robots is still d. Thus the robot y will also decide to move a distance $d/2$ and will eventually reach p. The only other case is when robot y sees robot x when Light[x] is BLUE. In this case, robot x has already arrived at p. During the *Compute* operation, robot y will decide to move directly to the other robot and it will also reach location p. Hence in all cases, the robots will gather at p during this Mega-Cycle.

■

We have shown that algorithm TWOGATHERLIGHT correctly solves the problem of gathering two robots when provided with a light of 4 colors. This completes the proof of Theorem III.2.

By Theorem III.1, Lemma III.2, and Theorem III.2 it follows that $\text{ASYNC}^{O(1)} \geq \text{SSYNC}$ and $\exists R \in \mathcal{R}$, $\mathcal{A}^{O(1)}(R) \setminus \mathcal{S}(R) \neq \emptyset$; that is,

Theorem III.3. $\text{ASYNC}^{O(1)} > \text{SSYNC}$.

C. $\text{ASYNC}^{O(1)}$ is as powerful as $\text{SSYNC}^{O(1)}$

To complete this section we now prove that, when enhanced with a constant number of visible bits, semi-synchronous robots are not more powerful than asynchronous ones with the same capability. More precisely we show the following:

Theorem III.4. $\forall R \in \mathcal{R}$, $\mathcal{S}^{O(1)}(R) \equiv \mathcal{A}^{O(1)}(R)$.

Proof: First, let us show that $\mathcal{S}^{O(1)}(R) \subseteq \mathcal{A}^{O(1)}(R)$; in particular, we will show that $\mathcal{S}^{k}(R) \subseteq \mathcal{A}^{6k}(R)$, $\forall k > 1$. Let $\mathcal{P}$ be a protocol designed for the SSYNC^k model. We show how to extend the simulation algorithm SIM to execute $\mathcal{P}$ in ASYNC^{6k}. Suppose we equip the robots with a second light bulb of k colors. The second light bulb would initially be set to the same color as the robots executing $\mathcal{P}$ in SSYNC^k. During the *Compute* step of the simulation whenever the robots compute a new destination, they also comupte the new color of the light bulb according to $\mathcal{P}$ and set the color of the second light bulb accordingly. All other steps of the algorithm are same as in SIM. From the correctness of SIM protocol it follows that the above algorithm would correctly simulate any protocol for semi-synchronous robots with k lights. Notice that we can replace the two light bulbs in the above simulation with a single light bulb having $6k$ colors. The other inclusion, hence the theorem, follows from the obvious relationship $\text{SSYNC}^{O(1)} \geq \text{ASYNC}^{O(1)}$. ■

Thus, we have shown that: $\text{ASYNC}^{O(1)} \equiv \text{SSYNC}^{O(1)}$. In other words, when enhanced with visible lights, *the difference between asynchrony and semi-synchrony disappears*. This result must be contrasted with the strict dominance between the models without lights.

D. A Note on Self-stabilization

Note that protocol TWOGATHERLIGHT is *self-stabilizing* in the sense that it works even when initially the lights have arbitrary colors. Protocol SIM, which is not self-stabilizing as described, can be easily modified to have this property. In fact, it is easy to characterize the "illegal" configurations; we can then add to SIM the rule that, if the result of a robot *Look* is an illegal configuration, then the robot turns its light to `S`, and waits until all lights become `S`. From that moment on, the protocol behaves correctly.

IV. ASYNCHRONY WITH VISIBLE LIGHTS VERSUS FULL SYNCHRONY

In this section we address the relationship between *full synchrony* and ASYNC when the latter is enhanced with both visible bits and persistent internal memory. We show that asynchronous robots if empowered with both a constant number of lights and the ability to remember a single snapshot from the past, become at least as powerful as synchronous robots.

A. $\text{ASYNC}^{O(1)}_{O(1)}$ is at least as powerful as FSYNC

We now present a protocol for ASYNC robots, which uses 3 colors, one past snapshot, and simulates FSYNC. In other words, we show that any problem solvable in FSYNC is solvable also in ASYNC^3_1.

Protocol SYNCSIM, whose rules are shown in Figure 6, uses three colors: `OFF`, `GREEN`, and `RED`; initially, all lights are `OFF`. Similarly to Protocol SIM, protocol SYNCSIM enforces a sequence of Mega-Cycles $mc_0, mc_1, \ldots$: the difference here is that *all* robots execute $\mathcal{P}$ in each Mega-Cycle based on the same snapshot (we are simulating FSYNC). Each mega-cycle mc_i starts with all robots being `OFF`; within finite time, all `OFF` robots become `GREEN`; when a robot becomes `GREEN` in a Mega-Cycle, it stores in a local array `Perm[]` the configuration it just observed: this is necessary to ensure that all robots will compute on the same configuration in this Mega-Cycle. After all robots become `GREEN`, the destination point is computed, using as configuration the one locally stored in `Perm[]` and the robot starts to perform the *Move* operation, turning its light to `RED`. After a robot has completed the *Move*, it changes its light to `OFF`. When the lights of all robots are `OFF`, the current Mega-Cycle ends and the next one begins.

Theorem IV.1. $\forall R \in \mathcal{R}$, $\mathcal{F}(R) \subseteq \mathcal{A}^3_1(R)$.

Sketch: The proof is by construction. We show that Protocol SYNCSIM correctly simulates a fully synchronous execution of any protocol $\mathcal{P}$.

Initially all robots are `OFF`. By construction, an `OFF` robot becomes `GREEN`, storing the current snapshot, but does not execute $\mathcal{P}$. As a consequence, all robots that become `GREEN` for the first time have stored the same snapshot. By construction, a `GREEN` robot does not execute $\mathcal{P}$ as long as it sees some `OFF` robot; on the other end, it does execute $\mathcal{P}$ on the stored snapshot if all the other robots are either `GREEN` or `RED`, and in this case it itself becomes `RED`. This means that all robots that become `GREEN` for the first time eventually execute $\mathcal{P}$ on the same snapshot and then become `RED`.

By construction, a `RED` robot always turns its light to `OFF` in the next activation cycle (i.e. after performing the *Move*). So, eventually all robots will be `OFF` again. At this point, each robot has executed one *Look-Compute-Move* cycle according to the protocol $\mathcal{P}$. We are now in the same conditions as before and the argument applies for the next cycle of activities. ■

The above result proves that: $\text{ASYNC}^3_1 \geq \text{FSYNC}$.

B. ASYNC^3_1 is more powerful than FSYNC

We proved that ASYNC^3_1 is at least as powerful as FSYNC. We now show that there are problems that can be solved in ASYNC^3_1 but are not solvable in FSYNC. Consider the BLINKING problem defined as follows:

Definition IV.1 (BLINKING)**.** The BLINKING problem requires $n > 2$ robots to perform subtasks T1 and T2 repeatedly in alternation. In T1, the robots must form a circle, i.e. each robot lies on a distinct point on the same circle C of radius $r_C > 0$; While in T2, the robots must gather at a single point.

Observe that, once n robots are gathered at a single point, it is not possible to separate these robots using any deterministic algorithm, even in the synchronous model. Thus,

Lemma IV.1. $\forall R \in \mathcal{R}_o$, BLINKING $\notin \mathcal{F}(R)$.

We will now show that

Lemma IV.2. $\forall R \in \mathcal{R}$; BLINKING $\in \mathcal{A}^3_1(R)$.

Proof: We will prove the above result by providing an algorithm for solving BLINKING. Starting from any initial configuration with robots in distinct locations, the robots simply move to the smallest enclosing circle (SEC) without creating multiplicity. Note that during this operation, the SEC remains invariant. Once a robot arrives at the SEC, it turn

State *Look*

Take the snapshot of the positions of the robots, that returns for all robots $r \in R$:

- $Pos[r]$, the position on the plane of robot r (according to my coordinate system);
- $Light[r]$, the color of the light of robot r.

(**Note**: I am robot x)

State *Compute*

$p :=$ `Pos[x]`.
Case `Light[x]`

- `OFF`
  If $\forall r \neq x$, `Light[r]` $=$ `OFF` $\vee$ `Light[r]` $=$ `GREEN`,
  Then
  Store the current snapshot into the *non-volatile* array `Perm[]`.
  `Light[x]` $:=$ `GREEN`.
- `GREEN`
  If $\forall r \neq x$, `Light[r]` $=$ `GREEN` $\vee$ `Light[r]` $=$ `RED`
  Then
  Execute $\mathcal{P}$ using the snapshot in `Perm[]`.
  $p :=$ computed destination.
  `Light[x]` $:=$ `RED`.
- `RED`
  `Light[x]` $:=$ `OFF`.

State *Move*

`Move(`p`)`.

Figure 6. Protocol SYNCSIM, that simulates FSYNC protocols in ASYNC_1^3.

its light `RED`. When a robot sees that all robots have their lights `RED`, the robot stores a snapshot and then turns its light to `GREEN` and waits for the other robots to turn their lights to `GREEN`. Note that each robot will store the same snapshot but oriented according to its local coordinate system. When a robot sees all other robots have `GREEN` light, the robot executes any standard gathering algorithm without changing the light. Once all robots have gathered at a point, each robot turns its light to `OFF`. When a robot sees all other lights are turned `OFF`, it moves back towards its previous location on the circle C which is the SEC of the robot's locations in the snapshot. Note that in the snapshot each robot knows its own location and can compute the vector from this point to the center of the circle C. The robot then simply moves along the inverse of this vector. The robot may not reach the circle, but once each robot has executed at least one cycle, all the robots will be in distinct locations and the robots can re-execute the same algorithm.

Using the above algorithm the robots in the ASYNC_1^3 model can solve the BLINKING problem. ■

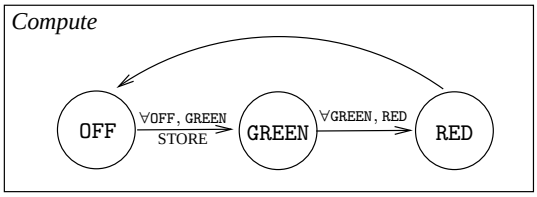

Figure 7. The transition diagram of the SYNCSIM protocol.

The above results show that: $\exists R \in \mathcal{R}$, $\mathcal{A}_1^3(R) \setminus \mathcal{F}(R) \neq \emptyset$. This combined with the results of the previous section imply the following:

Theorem IV.2. $\text{ASYNC}_1^3 > \text{FSYNC}$.

Thus, we have shown that the capability of using external lights and remembering a single snapshot allows ASYNC robots to become more powerful than FSYNC robots. In constrast, without the use of external lights, remembering any number of snapshots does not allow ASYNC robots to achieve the power of even the SSYNC model [22].

V. CONCLUSIONS

The results of this paper show the power of using lights, i.e. visible external memory, for distributed computations with autonomous robots. In fact, we have shown that using only a few bits of visible memory asynchronous robots can perform tasks which cannot be performed even with unbounded amount of internal memory. Moreover a team of robots empowered with lights (without or with snapshot-memory) is more powerful than an otherwise similar team of semi-synchronous robots (or, fully- synchronous robots respectively). In other words, asynchrony can be overcome with the power of lights.

REFERENCES

[1] N. Agmon and D. Peleg, "Fault-tolerant gathering algorithms for autonomous mobile robots," *SIAM Journal on Computing*, vol. 36, pp. 56–82, 2006.

[2] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita, "A distributed memoryless point convergence algorithm for mobile robots with limited visibility," *IEEE Transaction on Robotics and Automation*, vol. 15, no. 5, pp. 818–828, 1999.

[3] L. Barrière, P. Flocchini, E. Mesa-Barrameda, and N. Santoro, "Uniform scattering of autonomous mobile robots in a grid," *International Journal on Foundation of Computer Science*, vol. 22, no. 3, pp. 679–697, 2011.

[4] Z. Bouzid, S. Dolev, M. Potop-Butucaru, and S. Tixeuil, "Robocast: Asynchronous communication in robot networks," in *114th International Conference on Principles of Distributed Systems (OPODIS)*, ser. LNCS 6490, 2010, pp. 16–31.

[5] D. Canepa and M. G. Potop-Butucaru, "Stabilizing flocking via leader election in robot networks," in *9h International Conference on Stabilization, Safety, and Security of Distributed Systems (SSS)*, ser. LNCS 4838, 2007, pp. 52–66.

[6] M. Cieliebak, "Gathering non-oblivious mobile robots," in *6th Latin American Conference on Theoretical Informatics (LATIN)*, ser. LNCS 2976, 2004, pp. 577–588.

[7] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro, "Solving the robots gathering problem," in *30th International Colloquium on Automata, Languages and Programming (ICALP)*, ser. LNCS 2719, 2003, pp. 1181–1196.

[8] S. Das, P. Flocchini, N. Santoro, and M. Yamashita, "On the computational power of oblivious robots: forming a series of geometric patterns," in *29th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, 2010, pp. 267–276.

[9] X. Défago, M. Gradinariu, S. Messika, and P. Raipin-Parvédy, "Fault-tolerant and self-stabilizing mobile robots gathering," in *20th International Symposium on Distributed Computing (DISC)*, ser. LNCS 4167, September 2006, pp. 46–60.

[10] X. Défago and S. Souissi, "Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity," *Theoretical Computer Science*, vol. 396, no. 1-3, pp. 97–112, 2008.

[11] Y. Dieudonné, O. Labbani-Igbida, and F. Petit, "Circle formation of weak mobile robots," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 3, no. 4, 2008.

[12] Y. Dieudonné, F. Petit, and V. Villain, "Leader election problem versus pattern formation problem," in *International Symposium on Distributed Computing (DISC)*, ser. LNCS 6343, 2010, pp. 267–281.

[13] Y. Dieudonné and F. Petit, "Scatter of robots," *Parallel Processing Letters*, vol. 19, no. 1, pp. 175–184, 2009.

[14] A. Efrima and D. Peleg, "Distributed models and algorithms for mobile robot systems," in *33rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, ser. LNCS 4362, 2007, pp. 70–87.

[15] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer, "Hard Tasks for Weak Robots," in *10th Annual International Symposium on Algorithms and Computation (ISAAC)*, ser. LNCS 1741, 1999, pp. 93–102.

[16] ——, "Gathering of robots with limited visibility," *Theoretical Computer Science*, vol. 337, no. 1-3, pp. 147–168, 2005.

[17] ——, "Arbitrary pattern formation by asynchronous oblivious robots," *Theoretical Computer Science*, vol. 407, pp. 412–447, 2008.

[18] V. Gervasi and G. Prencipe, "Coordination without communication: the case of the flocking problem," *Discrete Applied Mathematics*, vol. 144, no. 3, pp. 324–344, 2004.

[19] T. Izumi, Z. Bouzid, S. Tixeuil, and K. Wada, "Brief announcement: The BG-simulation for byzantine mobile robots," in *25th International Symposium on Distributed Computing (DISC)*, 2011, pp. 330–331.

[20] Y. Katayama, Y. Tomida, H. Imazu, N. Inuzuka, and K. Wada, "Dynamic compass models and gathering algorithms for autonomous mobile robots," in *14th Colloquium on Structural Information and Communication Complexity (SIROCCO)*, ser. LNCS 4474, 2007.

[21] D. Peleg, "Distributed coordination algorithms for mobile robot swarms: New directions and challenges," in *7th International Workshop on Distributed Computing (IWDC)*, ser. LNCS 3741, 2005, pp. 1–12.

[22] G. Prencipe, "The effect of synchronicity on the behavior of autonomous mobile robots," *Theory Comput. Syst.*, vol. 38, no. 5, pp. 539–558, 2005.

[23] S. Souissi, X. Défago, and M. Yamashita, "Using eventually consistent compasses to gather memory-less mobile robots with limited visibility," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 4, no. 1, pp. 1–27, 2009.

[24] I. Suzuki and M. Yamashita, "Distributed anonymous mobile robots: formation of geometric patterns," *Siam Journal on Computing*, vol. 28, no. 4, pp. 1347–1363, 1999.

[25] M. Yamashita and I. Suzuki, "Characterizing geometric patterns formable by oblivious anonymous mobile robots," *Theoretical Computer Science*, vol. 411, no. 26-28, 2010.