On Gossip and populations

Marin Bertier, Yann Busnel, Anne-Marie Kermarrec

On Gossip and populations.

Marin Bertier
INSA Rennes

Yann Busnel
University of Rennes 1

Anne-Marie Kermarrec
INRIA Rennes – Bretagne Atlantique

Abstract

Gossip protocols are simple, robust and scalable and have been consistently applied to many (mostly wired) distributed systems. Nevertheless, most validation in this area has been empirical so far and there is a lack of a theoretical counterpart to characterize what can and cannot be computed with gossip protocols.

Population protocols, on the other hand, benefit from a sound theoretical framework but little empirical evaluation. In this paper, we establish a correlation between population and gossip-based protocols. We propose a classification of gossip-based protocols, based on the nature of the underlying peer sampling service. First, we show that the class of gossip protocols, where each node relies on an arbitrary sample, is equivalent to population protocols. Second, we show that gossip-based protocols, relying on a more powerful peer sampling service providing peers using a clearly identified set of other peers, are equivalent to community protocols, a modern variant of population protocols.

Leveraging the resemblances between population and gossip protocols enables to provide a theoretical framework for distributed systems where global behaviors emerge from a set of local interactions, both in wired and wireless settings. The practical validations of gossip-protocols provide empirical evidence of quick convergence times of such algorithms and demonstrate their practical relevance. While existing results in each area can be immediately applied, this also leaves the space to transfer any new results, practical or theoretical, from one domain to the other.

1 Introduction

In analogy with rumour spreading among human beings, gossip protocols provide a scalable, robust and reliable substrate for many peer to peer applications [11, 13, 17, 21, 22]. They have recently received an increased attention due to their scalability and quick convergence in large-scale dynamic settings. In a gossip protocol, each node in the system periodically exchanges information with another peer sampled from the network. The robustness of gossip protocols stems from their random flavour in the sampling. However, while there are some ad-hoc analyses available for specific protocols [11, 14, 21], most validation in the area has been so far achieved through extensive simulations and experimentations.

In [20], a generic practical gossip substrate has been defined. In this model, a protocol is defined by three functions: *(i)* **the peer selection**, identifying the gossip target, provided by a *peer sampling service*; *(ii)* the **data exchanged**, specifying the information exchanged between the peers during a gossip interaction and *(iii)* the **data processing** following an interaction. While this framework was initially defined to unify gossip membership systems, it has been shown to be generic enough to be applied for the whole spectrum of gossip protocols including reliable dissemination, distributed computation, and overlay construction [23]. Yet, there is a lack of clear theoretical framework enabling reasoning about the power and limitations of this model.

On the other hand, population protocols [1] provide theoretical foundations for distributed systems in which global behavior emerges from a set of simple interactions between their agents. Originally developed in the context of mobile tiny devices, typically sensors, in this model, agents are considered anonymous, and therefore, undistinguishable. Many variants of population protocols exist [2, 4, 5, 6, 10]. Among them, community protocols [16] augment the original model by assigning agents a unique identifier and letting nodes remember a limited number of other identifiers. Not only this significantly increases the computation

power of the system but also provides a way to tolerate a bounded number of byzantine failures. In the sequel, the class of population protocols and variants will be refered as *population protocols*, and the original model as *basic population protocol*.

More specifically, the population protocol model consists in a finite space of agent's states, a finite set of inputs, a finite set of outputs and a transition function. The set of possible node's interactions is represented by a graph. When two agents are sufficiently close for a sufficiently long time, they interact by exchanging their local information, and update their state according to the transition function. For instance, if agents are small devices embedded on animals, an interaction takes place each time two animals are in the same radio range. The interaction patterns, orchestrated by a scheduler, are considered as unpredictable. Yet, the scheduler is assumed to be *fair i.e.*, it ensures that any reachable global system state can be reached infinitely often. In the absence of global knowledge, agents cannot usually verify that the protocol has terminated, therefore the model considers convergence (of the distributed output) rather than termination.

Contributions: Correlating population and gossip protocols Our contributions in this paper stem from the observation that population and gossip protocols bear many resemblances. They both rely on a *scheduler* orchestrating the interactions between nodes. The scheduler, fair by assumption in population protocols, specifies the node interactions in a mobile environment while the scheduler is a *peer sampling service* in gossip protocols providing nodes with gossip targets. Both aim at achieving an emerging global behavior from a set of local interactions in a fully decentralized manner. The main contribution of this paper is to acknowledge these similarities and leverage them in both contexts.

On one hand, the gossip structure presented in [20] provides a practical generic framework in the context of wired systems. Yet, this does not provide a fine-grained classification. Works in this area have shown that the gossip protocols scale well in practice and convergence quickly. On the other hand, population protocols provide a theoretical framework for wireless systems. They clearly define the power and limitation of such protocols. Population protocols show that such systems composed of anonymous agents ensure the convergence of a clearly defined set of functions. Community protocols extend this model, by adding a bounded set of identifiers. However, until now, these models have not significant practical implication. In this paper, we present the following contributions:

– We establish a correlation between population and gossip protocols and introduce a *first classification* of gossip protocols depending on the nature of the underlying peer sampling service. More precisely, we identify two classes of gossip protocols: *anonymous* (AGP) and *non anonymous* (NGP).

– We show that anonymous gossip protocols are *equivalent* to basic population protocols;

– We show that non anonymous gossip protocols are *equivalent* to community protocols;

– By doing so, we leverage the theoretical framework of population protocols for understanding the power and limitations of gossip protocols. Likewise, we exploit the results obtained in the area of gossip protocols to draw conclusions on the practicality of population protocols. This enables us to provide both theoretical and practical considerations for such large-scale systems: the parallel between population and gossip protocols can be exploited for both existing and new results, as we propose in [9]. For instance, applying gossip experiments to population protocols enables to show the convergence behavior of these protocols. Likewise, applying some results from population protocols to the gossip-based protocols enable to show their computability or extract some interesting bounds as the one proposed in [9], where we provide a new result in the context of population protocol namely the *optimality of uniform distribution of interactions* [8] with respect to speed of convergence. Then, we use the equivalence property to extend it to gossip protocols. This enables us to conclude that the random peer sampling service [20] is optimal for the speed of convergence of gossip protocols. This is a clear illustration of how the correlation can be exploited. For space reasons, this last point is not developed in the paper. Details are available in [8].

2 Population vs gossip protocols

2.1 Background on population protocols

In this section, we briefly present the basic population protocol model and the community protocol variant, which relaxes the assumption on the anonymity of agents.

Basic population protocol The basic population protocol model, initially introduced in [1], is composed of a collection of agents, interacting pairwise in an order determined by a fair scheduler. Each agent has an input value and is represented by a finite state machine. This agent can only update its state through an interaction. Updates are defined by a transition function that describes the function f computed by the system. At each interaction, the agents compute an output value from their current state and converge eventually to the correct output value, depending to the inputs initially spread to the agents.

More formally, a population protocol is composed of:

- a complete interaction graph Λ linking a set of $n \geq 2$ agents;
- a finite input alphabet Σ;
- a finite output alphabet Y;
- a finite set of possible agent's states Q;
- an input function $\iota : \Sigma \to Q$ mapping inputs to states;
- an output function $\omega : Q \to Y$ mapping states to outputs;
- a transition relation $\delta : Q \times Q \to Q \times Q$ on pairs of states.

In the following, we call $(p, q) \mapsto (p', q')$ or (p, q, p', q') a transition if $(p, q, p', q') \in \delta$. A transition can occur between two agents' states only if these two agents have an interaction. The protocol is deterministic if δ is a function (*i.e.* at most one possible transition for each pair in Q^2).

A configuration of the system corresponds to an unordered multiset containing states of all agents. We denote $C \to C'$ the fact that a configuration C' can be obtained from C in one step (*i.e.* with only one transition for one existing interaction). An execution of the protocol is a finite or infinite sequence of population configurations $C_0, C_1, C_2, \ldots$ such that $\forall i, C_i \to C_{i+1}$.

As introduced above, the order of the interactions is unpredictable, and decided by the scheduler. The scheduler is assumed to be *fair*, *i.e* a feasible configuration cannot be endlessly ignored. In other words, if a configuration C appears an infinite number of times during an execution, and there exists a step $C \to C'$, then C' must also appear an infinite number of times in the execution. This ensures that any attainable configuration is eventually reached.

Community protocols Many variants of the last model exist. In this paper, we focus on the community protocol [16] extension, which significantly increases the computational power. This model augments the basic population protocol model by assigning unique identifiers to agents. All possible identifiers and a special symbol $\perp$ are grouped in an infinite set U. The difference between basic population protocols and community protocols is the definition of the set of states: $Q = B \times U^d$ where B is the initial definition of the population protocol's set of states collapsed to a memory of d identifiers. As in population protocols, algorithms cannot use any bound on the number of agents and moreover, U is infinite. In order to maintain the population protocol spirit in this extended model, some constraints are added: only existing agent identifiers can be stored in the d slots intended for identifiers of an agent's state and no other structural information about identifiers can be used by algorithms. We consider, for $q \in Q$ and $id \in U$, that $id \in q$ means that q stores id in one of its d indentifier slots. Thus, community protocols have to verify the two following formal constraints:

- $\forall (q_1, q_2) \mapsto (q_1', q_2') \in \delta, id \in q_1' \vee id \in q_2' \Rightarrow id \in q_1 \vee id \in q_2$
- For $q = \langle b, u_1, u_2, \ldots, u_d \rangle \in Q$, let $\hat{\pi}(q) = \langle b, \pi(u_1), \pi(u_2), \ldots, \pi(u_d) \rangle$ where π a permutation of U with $\pi(\perp) = \perp$. We assume that: $\forall (q_1, q_2) \mapsto (q_1', q_2') \in \delta : (\hat{\pi}(q_1), \hat{\pi}(q_2)) \mapsto (\hat{\pi}(q_1'), \hat{\pi}(q_2')) \in \delta$.

In short, the first assumption ensures that no transition introduce new identifiers and the second one

Algorithm 1: Generic Gossip Protocol

Active thread
Do once for each T time units at a random time
begin
  p = SelectPeer()
  Send DataExchange($state$) **to** p
  Receive *info*$_p$ **from** p
  $state$ = DataProcessing(*info*$_p$)
end

Passive thread
Do forever
begin
  Receive *info*$_q$ **from** q
  Send DataExchange($state$) **to** q
  $state$ = DataProcessing(*info*$_q$)
end

that identifiers can only be stored or compared for equality, but not manipulated in any other way. Any population protocol can be viewed as a community protocol with $d = 0$.

Finally, a population or community protocol stably computes a function $f : \Sigma^+ \to Y$ if $\forall n \in \mathbb{N}, \forall \sigma \in \Sigma^n$, every fair execution with n agents initialized with the elements of σ, eventually stabilizes to output $f(\sigma)$. That means that the output value of every agent eventually stabilizes to $f(\sigma)$.

2.2 Gossip protocols: A practical framework

Originally introduced for information dissemination, gossip protocols are simple, robust and scalable. Initially, epidemic-based algorithms have been proposed for database maintenance in [11]. Since then, they have been applied in many settings in wired and wireless systems as in [7, 13, 14, 15, 17, 25, 26, 27].

A generic framework has been proposed in [20] to provide a generic substrate for gossip peer sampling protocols, providing a common ground for membership systems. In this paper, the authors explore the resulting topologies and show that the parameters of the generic gossip protocols can be set to achieve random-like graph topologies *i.e.* providing each node with a random sample of the network. This has been achieved through extensive experimentations and has been recognized as a way to achieve random peer sampling in large-scale dynamic networks.

In this framework, each peer maintains a local view of size c of the system, representing its restricted knowledge of the systems. The peer sampling service provides a sample from that view. This framework relies on three basic functions:

SelectPeer() returns a peer from the local view. This function is used to select the gossip target;

DataExchange() returns the data to be exchanged over a gossip communication;

DataProcessing() returns the resulting state and specifies the way the data exchanged are processed.

Each peer runs an active and a passive threads (see Algorithm 1). The active thread is run periodically and launches a gossip interaction. A gossip target is selected from its local view using (*SelectPeer()*), data is exchanged (*DataExchange()*) and processed between the two interacting peers (*DataProcessing()*).

Initially proposed in the context of protocols achieving unstructured topologies, it turns out that this very substrate is generic enough to be used for many other purposes [23]. For example, message dissemination [22, 13] can be achieved by parameterizing the protocol as follows. *(i) SelectPeer()* should return a random peer; *(ii) DataExchange()* should contain the message to disseminate; and *(iii) DataProcessing()* should do nothing. Likewise, distributed computations (average, sum or quantile) [19, 21], gossip size estimation [12, 24] or overlay construction [17, 27] can be achieved using the same protocol. This substrate provides a general framework for practical implementations of gossip protocols. Yet, to the best of our knowledge, there is no theoretical counterpart (in the sense of a framework).

3 A classification of gossip protocols

3.1 On the power of the peer sampling service

In this section, we propose a novel classification of gossip protocols stemming from the observation that, although studied independently by two different communities, population and gossip protocols have a lot in common. Both class of protocols rely on the following properties:

– a fully decentralized model;

– a set of agents, having a finite storage capacity, periodically interacting in a pairwise manner. The agents are mobile and communicate in a wireless manner in population protocols; they are static and communicate through a dynamic network on a fixed infrastructure in gossip protocols;

– an unpredictable order of interactions orchestrated by a fair scheduler in population protocols modelling the agents' mobility patterns and by a peer sampling service serving the *selectPeer()* function in gossip protocols;

– a function specifying the way data is processed over an interaction: this is the transition function δ in population protocols and the *DataProcessing* function in gossip protocols;

– a state exchange over an interaction: this is the state value in Q in population protocols and the *DataExchanged* function in gossip protocols.

Considering the gossip protocols in this light, we were able to make a parallel between *(i)* the difference between the basic population and community protocols and *(ii)* the requirements for peer identifiers in gossip protocols.

More specifically, gossip protocols differ from their requirement with respect to peer anonymity. This is instantiated by the nature of the underlying peer sampling protocol. The peer sampling service, *i.e.* the "black box" providing a peer with a given sample of the network, may either return any sample for the implementation of the gossip protocol. Therefore, we introduce the following classification. Two main classes of gossip protocols can then be defined depending on the power of the underlying peer sampling service with respect to anonymity requirements.

AGP: Anonymous Gossip protocols do not require being aware of the identities of any peer for any of the three functions of the generic protocol. This is typically the case of protocols achieving some simple distributed computations such as average computation [19, 21] or system size estimations [18, 24]. Gossip dissemination protocols where each peer gossips to k nodes picked uniformly at random also fall into this category. Such protocols only rely on a peer sampling service providing them with a sample of the network, be it random or biased [7, 22].

NGP: Non-anonymous Gossip Protocols are not oblivious to the identities of peers they are communicating with or any other. Typically, gossip overlay construction protocols fall into this class. The identities of peers are required in the three functions of the substrate aforementioned. Non-anonymous gossip protocols have been used to implement overlays ranging from unstructured networks, providing random-graph like topologies [20] to structured networks [17, 22].

3.2 Between synchronous and asynchronous

To refine the classification, we take into account the two main models of communication channels. In a synchronous model, message delay, clock drift and time required to execute an algorithm step are bounded and these bounds are known. On the contrary, in an asynchronous model, there is no bound. Gossip protocols have a periodic behavior, in which at each step, every agent launches an exchange with another agent. In a synchronous system, the gossip keeps its periodic behavior, and so every agent is in the same communication round. In an asynchronous system, the clock drift and the fact that a time required by a gossip exchange cannot be bound infers that the periodic behavior is lost.

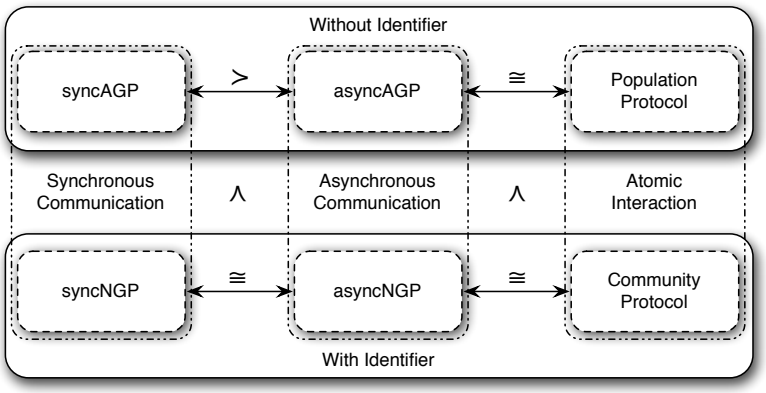

Figure 1: Relationship between gossip-based and population protocols

3.3 On the computational power of gossip protocols

Enriching our model with the communication synchronism property, we obtain a refined classification as the one presented earlier. We now classify gossip protocols in four classes:

syncAGP Synchronous Communication and Anonymous Nodes;
asyncAGP Asynchronous Communication and Anonymous Nodes;
syncNGP Synchronous Communication and Non-anonymous Nodes;
asyncNGP Asynchronous Communication and Non-anonymous Nodes.

Obviously, the power of non-anonymous gossip protocols is greater than anonymous ones as the use of node identifier enables to achieve distributed computations which are impossible in the anonymous context (*eg.* exponential computation, logical overlay construction, *etc*.). Thus, we have:

$$\text{asyncAGP} \prec \text{asyncNGP} \quad \text{and} \quad \text{syncAGP} \prec \text{syncNGP}$$

As far as anonymous gossip protocols are concerned, it is possible to leverage the periodicity of exchange in order to increase their computational power. For instance, it is possible in syncAGP to establish a global time clock, thanks to the cycle structure, but not in asyncAGP (due to the unbounded message delay). Then, it is obvious to conclude that:

$$\text{asyncAGP} \prec \text{syncAGP}$$

Finally, in the NGP context, we show in the proof of Lemma 6 that the identification of nodes enables to emulate synchronous communications, such that:

$$\text{asyncNGP} \cong \text{syncNGP}$$

This classification and the relation between all the considered models are summarized in Figure 1. This provides a refined classification of gossip protocols based on the synchronism and anonymity properties. Yet, there is no formal framework to define what can and cannot be achieved with the susmentioned protocols. Establishing the parallel between population and gossip protocols provides a first answer to that question.

4 Bridging the gap between population and gossip protocols

In a nutshell, in AGP, a peer sampling service provides each peer with another peer to communicate with, regardless of its identifier. If the peer sampling service ensures that any pairwise interaction can endlessly

take place, then a protocol of AGP resembles a basic population protocol. Inversely, a protocol from NGP requires a peer sampling service to provide each node with a set of clearly identified peers, potentially along with more information about each peer, where the identifier (whether it is an identifier or an IP address) is crucial. This means that the *SelectPeer* or the *DataProcessing* use the identifier or information attached to specific peer to achieve a given functionality. This clearly matches the community protocol model described above. We claim that these resemblances are actually equivalences and provide the proofs in this section as illustrated on Figure 1.

4.1 Equivalence between basic population and anonymous asynchronous gossip protocols

In this section, we prove that the basic population and anonymous asynchronous gossip protocols (asyncAGP) are equivalent.

Theorem 1 *A predicate is computable by a basic population protocol if and only if it can be computed by an anonymous gossip protocol via an asynchronous communication model (asyncAGP).*

Proof. In order to prove the equivalence, we consider the functions computable by basic population protocols and asyncAGP. Then, we prove in Lemmas 2 and 3 that they belong to the same equivalence class. In fact, we prove below that the class of functions computable by a basic population protocol is a subset of the ones computable by asyncAGP, and *vice-versa*. ■

On one hand, consider that PP $\prec$ asyncAGP with the following lemma.

Lemma 2 *If f is computable by a basic population protocol, then there exists a protocol from asyncAGP which can compute f.*

Proof. Let $\mathcal{P}$ the basic population protocol computing f and defined by the 7-tuples $(\Lambda, \Sigma, Y, Q, \iota, \omega, \delta)$. Consider the anonymous gossip protocol $\mathcal{G}$ described below. We have to show that $\mathcal{G}$ simulates $\mathcal{P}$.

Similarities: Each agent in Λ is hosted by a specific peer of $\mathcal{G}$. Input, output and state sets are the same in $\mathcal{G}$ than in $\mathcal{P}$. *A fortiori*, both map function ι and ω remain identical in $\mathcal{G}$.

Dealing with the transition function: The $\mathcal{G}$'s *DataProcessing* function is defined from δ: consider two peers in the system l and r, gossiping with each other at a time t. Let assume that l initiates the gossip exchange with r. Let p_l (respectively p_r) the selected information obtained by *DataExchange* from l's state (respectively r's state). Thus, as l calls the function during its active thread, *DataProcessing* returns locally the third entry of the 4-tuple $(p_l, p_r, p'_l, p'_r) \in \delta$, and the state of l becomes p'_l. On the remote peer r, a call to the function *DataProcessing* during its passive thread returns the last entry of the same 4-tuple (p_l, p_r, p'_l, p'_r).

Thus, the sequence of population configurations is valid and represents the basic population protocol $\mathcal{P}$, as it only stems from the transition function δ using pairwise interactions.

On the fairness assumption: The last assumption to verify is the fairness condition. In asyncAGP, the scheduler is fully defined by the order of gossip exchanges, itself defined by *(i)* the *selectPeer* function; *(ii)* the randomization of the gossip time and; *(iii)* the asynchronous environment which, as we mentioned before, potentially leads to gossip exchange losses. In that context, every possible finite scheduling has a non-null probability to happen. Thus, every possible transaction between two system configurations $C \rightarrow C'$ has a non-null probability to happen. Moreover, if the asynchronism acts against this condition, given a specific interaction, which is avoided for a while, the probability that it continues to be avoided tends to zero. So, if the configuration C appears an infinite number of times during an execution of $\mathcal{P}$ in the aforementioned context, then C' also appears an infinite number of times in this execution. The fairness assumption is then verified.

Then, $\mathcal{G}$ simulates the basic population protocol $\mathcal{P}$, which computes the function f. Thus, for any function computable by basic population protocols, there exists an anonymous gossip protocol, which stably computes this function. ■

On the other hand, let's show the inverse of Lemma 2, corresponding to the second implication of Theorem 1: asyncAGP $\prec$ PP.

Lemma 3 *If f is computable by a protocol from asyncAGP, then there exists a basic population protocol which can compute f.*

Proof. Let $\mathcal{G}$ an anonymous gossip protocol that computes a specific function f using the primitive *DataExchange* and *DataProcessing*. As presented above, in a gossip protocol, peers are modeled by a finite-state machine.

Mapping the domain of the transition function: The domain of *DataProcessing* is finite (and corresponds to the Cartesian product of $\mathcal{D}_S$, the set of peer states, with $\mathcal{D}_E$, the range of *DataExchange*). Moreover, as *DataProcessing* is a function, its range is also finite by definition. Based on these sets, we define $\mathcal{D}_\mathcal{G}$ a specific subset of the Cartesian product between the domain and the codomain of *DataProcessing* (*i.e.* $\mathcal{D}_\mathcal{G}$ contains each ordered pair such that the first entry is in the domain of *DataProcessing* and the second entry is the mapped element of this first entry by *DataProcessing*). More formally, $\mathcal{D}_\mathcal{G} \subseteq (\mathcal{D}_S \times \mathcal{D}_E) \times \mathcal{D}_S$. Thus, $\mathcal{D}_\mathcal{G}$ is finite and contains all the possible transitions of peer states, based on the knowledge of a remote peer sub-state.

Design of the basic population protocol for the purpose of simulation: Consider the following basic population protocol $\mathcal{P}$, represented by the 7-uplet $(\Lambda, \Sigma, Y, Q, \iota, \omega, \delta)$. Consider a complete interaction graph Λ. Let the set of agent states be identical to the set of peer states, *i.e.* $Q = \mathcal{D}_S$. Consider that Σ and Y are the same as the input and output sets of $\mathcal{G}$, if they exist. In this case, ι and ω are the same functions than the ones which respectively associate the input set of $\mathcal{G}$ to $\mathcal{D}_S$, and $\mathcal{D}_S$ to the output set of $\mathcal{G}$. Conversely, if no specific input and output sets are defined in $\mathcal{G}$, then $\Sigma = Y = \mathcal{D}_S$ and $\iota \equiv \omega$ corresponding to the identity function. Finally, the transition function δ is defined as follows.

$$\forall (s_l, s_r, s'_l) \in \mathcal{D}_\mathcal{G}, \exists (s_r, s_l, s'_r) \in \mathcal{D}_\mathcal{G} \quad \text{such that} \quad (s_l, s_r, s'_l, s'_r) \in \delta.$$

On the periodicity of the fair scheduler: Periodicity of exchange is an inherent characteristic of many gossip protocols. However, the only difference between asyncAGP and syncAGP is that asyncAGP potentially temporary jeopardize the periodicity of exchanges, in case of arbitrary long transmission delays. Thus, as presented in Lemma 2, no periodic assumption can be considered in an asynchronous environment. Therefore, a fair scheduler is sufficient to lead to a correct execution of $\mathcal{G}$, using the aforementioned $\mathcal{P}$.

Thus, there exists a basic population protocol $\mathcal{P}$, which simulates the considered asyncAGP $\mathcal{G}$ and computes the function f. Then, for any function computable by a protocol from asyncAGP, there exists a basic population protocol, which stably computes this function. ■

4.2 Equivalence between community protocols and NGP

Along the same lines, we prove here the following theorem in order to prove the equivalence between community and NGP protocols. In fact, we show in the proof of Lemma 6 that it is possible to simulate the periodicity of protocols from syncNGP using a protocol from asyncNGP (these two classes are then equivalent).

Theorem 4 *A predicate is computable by a community protocol if and only if it can be computed by a non-anonymous gossip protocol (NGP).*

Proof. As Theorem 1, the proof of Theorem 4 is directly infered from the statements of Lemmas 5 and 6, which show respectively both implications of this equivalence. ■

Consider the first implication of this theorem. Inspired from the equivalence between basic population protocols and asyncAGP, the following theorem is almost trivial.

Lemma 5 *For each f computable by a community protocol, there exists a NGP protocol which compute f.*

Proof. The only difference between a basic population protocol and a community protocol consists in the definition of the set of states (*i.e.* $Q = B \times U^d$) and the two constraints on the state's identifier part (*i.e.* the part belonging to U^d cannot be used freely). Then, due to Lemma 2 and its sketch of proof, the protocol from NGP has to be designed to simulate a given community protocol $\mathcal{C}$. Then, we can still consider the function *selectPeer* as a black box which provides a fair scheduler. In other hand, functions *DataExchange* and *DataProcessing* are defined respectively on the domain $\mathcal{D}_S = B \times U^d$ (instead of B in the anonymous gossip version) and $\mathcal{D}_S \times \mathcal{D}_E$.

This does not violate the definition of NGP as the additional information used here only depends on the presence of unique identifier on agents, which is a mandatory assumption in non anonymous gossip protocols. ■

Finally, let now show the opposite of Lemma 5, corresponding to the second part of Theorem 4.

Lemma 6 *For each f computable by a NGP protocol, there exists a community protocol which computes f.*

Proof. Let $\mathcal{G}$ the given protocol from NGP. Thus, each peer in the system is aware of its unique identifier. Consider the following community protocol $\mathcal{C}$.

Preliminary assumptions on $\mathcal{C}$: We assume that, in the community protocol used on $\mathcal{C}$, agents are uniquely identified, and a unique agent is assigned a specific identifier $id_{\mathcal{L}}$. This agent is considered as the leader by all other agents. In this specific community protocol, we assume that this leader is aware of the size of the system[1] (denoted n in the sequel). In other words, the view of a peer is modeled as the set of $d-1$ identifiers in the community protocol model (one space of the d-tuple in U^d is set aside for storing its own identifier).

Summary of agent state requirement: In addition of the gossip peer state in $\mathcal{D}_S$, each agent maintains:

– a binary variable gc_{parity} to memorize the current gossip cycle parity;

– a ternary variable gc_{progress} storing the gossip state of an agent at the corresponding cycle. gc_{progress} can only take one of the three following values: *(i)* $todo$ if the active thread has not been run yet during the current gossip cycle, *(ii)* $done$ if it has been run or *(iii)* $wait$ representing the inter-cycle state, as explained below (*i.e.* the agent waits to pass across the synchronization barrier);

– an agent's identifier variable id_{next}, which stores the identifier of the next agent to gossip with;

– the leader agent $\mathcal{L}$ maintains a counter gc_{count} used in the synchronization cycle process.

To make a long story short, the set of agent states is defined as

$$Q = \mathcal{D}_S \times \{true, false\} \times \{todo, done, wait\} \times U \times [\![1; n]\!] \times U^d$$

i.e. the Cartesian product between *(i)* the domain of the function *DataProcessing* (to represent the gossip peer state), *(ii)* the domain of gc_{parity}, *(iii)* the domain of gc_{progress}, *(iv)* the identifier set U for id_{next}, *(v)* the domain of gc_{count} and finally *(vi)* U^d for the view of the agent. At initialization, each agent sets gc_{parity} to $false$, gc_{progress} to $todo$ and id_{next} to $id_{\mathcal{L}}$. Moreover, gc_{count} is set to 0 for the leader agent $\mathcal{L}$.

Simulating a gossip cycle in $\mathcal{C}$: We now describe the behavior of an agent during a gossip cycle, according to its gc_{progress} value. In the case that $gc_{\text{progress}} = todo$, the corresponding agent has not run its active thread in a given gossip round. Then, it waits to meet its next gossip partner, corresponding to the identifier stored in id_{next}. For each interaction (id_1, id_2), the agent corresponding to id_1 verifies if $gc_{\text{progress}} = todo$. If this is the case, it checks if $id_2 = id_{\text{next}}$. Only in that case, id_1 and id_2 run respectively *DataProcessing*(q_1,*DataExchange*(q_2)) and *DataProcessing*(q_2,*DataExchange*(q_1)) (where q_1 and q_2 represents respectively the state of these agents). All the possible transitions are included in $\mathcal{D}_{\mathcal{G}}$ introduced in the proof of Theorem 3. At the end of this interaction, id_1 and id_2 have updated their own state according to the previous one (q_i) and the remote one (*DataExchange*(q_j)). Finally, id_1 sets gc_{progress} to $done$. In other words, each agent waits until it encounters the agent with the identifier stored in id_{next} to gossip with.

How to simulate the T periodicity: In order to simulate the cycle in the context of community protocol, the T time slot can be simulated through a synchronization barrier. In the rest of this proof, we present how to establish such a barrier and how to assign agents to gossip cycles. So, we introduce the behavior of agents in case that $gc_{\text{progress}} \neq todo$.

Consider an agent id. After its own active gossip, the gossip state of id becomes $done$. In this state, it only waits until it encounters the agent $id_{\mathcal{L}}$. During its next interaction with $id_{\mathcal{L}}$, id sets its gc_{progress} to $wait$ and $id_{\mathcal{L}}$ increments gc_{count} by 1. Thus, all agents eventually stabilize to the $wait$ state, and gc_{count} eventually converges to n (the number of agents in the population). At this point, all agents have reached the synchronization barrier.

After the barrier, $id_{\mathcal{L}}$ enables all agents to begin the next gossip cycle as described hereinafter. $id_{\mathcal{L}}$ switches its own gc_{parity} to its opposite value, gc_{progress} to the $todo$ value, id_{next} to the returned value of *selectPeer* and finally gc_{count} to 0. At this point, each time an agent in the $wait$ state interacts with an agent owning the opposite value of gc_{parity}, will fall into the next cycle by switching gc_{parity}, and setting gc_{progress} and id_{next} respectively to the $todo$ value and the returned value of *selectPeer*. Then, all agents eventually leave the $wait$ state of the last cycle, and are ready for their next gossip exchange.

[1] This assumption is only expected for the synchronisation barrier. It can be relaxed using the probabilistic clock phase mechanism proposed for population protocols in [2].

In conclusion, if $\mathcal{G}$ computes the function f, then the community protocol $\mathcal{C}$ simulates the behavior of $\mathcal{G}$ and also computes the function f. ■

The last step of this proof lets us show that a protocol from NGP in an asynchronous environment is able to simulate a syncNGP. It is obvious that asyncNGP $\prec$ syncNGP (for the same reason that in AGP). Thus, we can conclude that asyncNGP $\cong$ syncNGP, and consequently, that community protocols are equivalent to all protocols from NGP (asyncNGP $\cup$ syncNGP). We then have proved all the claims presented in Figure 1.

4.3 Leveraging the relations

The equivalences above are of the utmost importance in the area of gossip protocols. They clearly define what can be computed with an anonymous gossip protocol. They show that the functions of the Presburger arithmetic eventually converge using an anonymous gossip protocol. They also prove that no other function can be computed with such a protocol [1, 3]. This is a new and important result in the area of gossip protocols. On the other hand, the empirical results on the convergence times and practicality of the gossip protocols can be used to evaluate the efficiency of population protocols.

Likewise, gossip protocols relying on a peer sampling service providing peers, with a bounded set of identifiers, are equivalent to community protocols. This can be used to achieve any computation of symmetric function from $NSPACE(n \log n)$ (namely a high number of functions) and also to implement algorithms tolerating failures (and not only benign ones).

These results can be leveraged for existing results as well as results to come in both areas. Due to space constraint, we cannot develop these observations in this paper, but in [9], we illustrate this claim by considering a gossip protocol and considering it from the population protocol standpoint and the other way around.

5 Conclusion and future works

The main contribution of this paper is to establish a correlation between population and gossip protocols. This parallel between two worlds, explored so far independently, offers several extremely interesting outcomes. First it enables to provide a first classification of gossip protocols, a theoretical framework for such protocols, allowing to specify in a formal way what can and cannot be computed by a gossip protocol depending on the nature of the underlying peer sampling service. If a gossip protocol relies on a peer sampling service oblivious to identifiers, it is equivalent to a population protocol. If the peer sampling service is identifier-aware, a gossip protocol is equivalent to a community protocol. For example, it is now clear that the multiplication, which does not belong to the Presburger arithmetic, cannot be achieved by an anonymous gossip protocol.

Conversely, this equivalence enables to leverage the properties obtained empirically on gossip protocols with respect to scalability, practicality and speed of convergence and apply them to population protocols. Apart from exploiting the already known results, this opens the door to leverage any new result in one of these two areas as our case studies demonstrate [9].

Part of the future work is to explore further the classes of population protocol to refine our classification of gossip protocols. Quantifying formally the convergence times of such protocols also remains an open issue.

Acknowledgment We would like to warmly thank Davide Frey, Carole Delporte-Gallet and Rachid Guerraoui for theirs comments and suggestions.

References

[1] D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing (Special Issue: PODC'04)*, 18(4):235–253, March 2006.

[2] D. Angluin, J. Aspnes, and D. Eisenstat. Fast computation by population protocols with a leader. In *Distributed Computing (Special Issue: DISC'07)*, 21(2):183–199, June 2008.

[3] D. Angluin, J. Aspnes, and D. Eisenstat. Stably computable predicates are semilinear. In *25th annual ACM Symposium on Principles of Distributed Computing (PODC '06)*, pages 292–299, August 2006.

[4] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. In *Distributed Computing (Special Issue: PODC'06)*, 20(4):279–304, November 2007.

[5] D. Angluin, J. Aspnes, M.J. Fischer, and H. Jiang. Self-stabilizing population protocols. In *9th International Conference Principles of Distributed Systems (OPODIS'05)*, volume LNCS 3974:103–117, December 2005.

[6] J. Aspnes and E. Ruppert. An introduction to population protocols. *Bulletin of the European Association for Theoretical Computer Science, Distributed Computing Column*, 93:98–117, October 2007.

[7] Y. Busnel, M. Bertier, E. Fleury, and A.-M. Kermarrec. GCP: Gossip-based code propagation for large-scale mobile WSN. In *The First International Conference on Autonomic Computing and Communication Systems (Autonomics'07)*, October 2007.

[8] Y. Busnel, M. Bertier, and A.-M. Kermarrec. On the Impact of the Mobility on Convergence Speed of Population Protocols. Research Report RR-6580, INRIA, Rennes, France, July 2008.

[9] Y. Busnel, M. Bertier, and A.-M. Kermarrec. Bridging the Gap between Population and Gossip-based Protocols. Research Report RR-6720, INRIA, Rennes, France, November 2008.

[10] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and E. Ruppert. When birds die: Making population protocols fault-tolerant. In *Second IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'06)*, pages 51–66, June 2006.

[11] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *6th ACM Symposium on Principles of Distributed Computing (PODC 1987)*, 1987.

[12] K. Dionysios, D. Psaltoulis, I. Gupta, K. Birman, and A. Demers. Active and passive techniques for group size estimation in large-scale and dynamic distributed systems. In *Elsevier Journal of Systems and Software*, vol.80:1639–1658, October 2007.

[13] P. T. Eugster, S. Handurukande, R. Guerraoui, A.-M. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. *ACM Transactions on Computer Systems*, 21(4):341–374, 2003.

[14] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié. Epidemic information dissemination in distributed systems. *IEEE Computer*, 37(5):60—67, May 2004.

[15] D. Gavidia, S. Voulgaris, and M. van Steen. Epidemic-style monitoring in large-scale wireless sensor networks. Technical Report IR-CS-012, Vrije Universiteit Amsterdam, 2005.

[16] R. Guerraoui and E. Ruppert. Even small birds are unique: Population protocols with identifiers. Technical Report Technical Report CSE-2007-04, Dept of Computer Science and Engineering, York University, September 2007.

[17] M. Jelasity and O. Babaoglu. T-Man: Fast gossip-based construction of large-scale overlay topologies. Technical Report UBLCS-2004-7, University of Bologna, Department of Computer Science, Bologna, Italy, May 2004.

[18] M. Jelasity and A.-M. Kermarrec. Ordered slicing of very large-scale overlay networks. In *6th IEEE International Conference on Peer-to-Peer Computing (P2P '06)*, pages 117–124, September 2006.

[19] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, 23(3):219–252, 2005.

[20] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM Transactions on Computer Systems*, 25(3):8, 2007.

[21] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *44th Annual IEEE Symposium on Foundations of Computer Science (FOCS '03)*, pages 482–491, octobre 2003.

[22] A.-M. Kermarrec, L. Massoulié, and A. J. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Transactions on Parallel and Distributed Systems*, volume 14(3), March 2003.

[23] A.-M. Kermarrec and M. van Steen, ed. *ACM Operating Systems Review on Gossip Potocols*, volume 41(5), October 2007.

[24] L. Massoulié, E. Le Merrer, A.-M. Kermarrec, and A. Ganesh. Peer counting and sampling in overlay networks: random walk methods. In *25th ACM Symposium on Principles of Distributed Computing (PODC '06)*, pages 123–132, July 2006.

[25] R. van Renesse. Power-aware epidemics. In *International Workshop on Reliable Peer-to-Peer Systems*, 2002.

[26] E. Simonton, B. Kyu Choi, and S. Seidel. Using gossip for dynamic ressource discovery. In *35th International Conference on Parallel Processing (ICPP'06)*, pages 319–328, August 2006.

[27] S. Voulgaris, D. Gavidia, and M. van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Journal of Network System Management*, 13(2), 2005.