Coupling-Based Internal Clock Synchronization for Large Scale Dynamic Distributed Systems

Roberto Baldoni, Angelo Corsaro, Leonardo Querzoni, Sirio Scipioni, and Sara Tucci Piergiovanni

***Abstract*—This paper studies the problem of realizing a common software clock among a large set of nodes without an external time reference (i.e., internal clock synchronization), any centralized control and where nodes can join and leave the distributed system at their will. The paper proposes an internal clock synchronization algorithm which combines the gossip-based paradigm with a nature-inspired approach, coming from the *coupled oscillators* phenomenon, to cope with scale and churn. The algorithm works on the top of an overlay network and uses a uniform peer sampling service to fullfill each node's local view. Therefore, differently from clock synchronization protocols for small scale and static distributed systems, here each node synchronizes regularly with only the neighbors in its local view and not with the whole system. An evaluations of the convergence speed and of the synchronization error of the coupled-based internal clock synchronization algorithm has been carried out, showing how convergence time and the synchronization error depends on the coupling factor and on the local view size. Moreover the variation of the synchronization error with respect to churn and the impact of a sudden variation of the number of nodes have been analyzed to show the stability of the algorithm. In all these contexts, the algorithm shows nice performance and very good self-organizing properties. Finally, we showed how the assumption on the existence of a uniform peer-sampling service is instrumental for the good behavior of the algorithm.**

***Index Terms*—Peer-to-Peer, Internal Clock Synchronization, Peer Sampling, Overlay Networks.**

I. Introduction

CLOCK synchronization is a fundamental building block for many distributed applications. As such, the topic has been widely studied for many years, and several algorithms exist which address different scales, ranging from local area networks (LAN), to wide area networks (WAN). For instance, the Network Time Protocol (NTP) [30], [31], has emerged as a standard de facto for external clock synchronization in both LAN and WAN settings. The work presented in this paper is motivated by an emergent class of large scale infrastructures, applications and services (e.g. managing large scale datacenters [38], [39], Cloud Computing [17], peer-to-peer enterprise infrastructures [5]), operating in very challenging settings, for which the problem of synchronizing clocks is far from being solved. These applications are required to (1) operate without any assumption on deployed functionalities, pre-existing infrastructure, or centralized control, while (2) being able to tolerate churn, due to crashes or to node joining or leaving the system, and (3) scaling from few hundreds to tens of thousands of nodes. For instance, publish/subscribe messaging implemented with the data distribution service [32], requires synchronized clocks to guarantee real-time message ordering. Also, in several relevant scenarios, due to security issues, or limited assumptions on the infrastructure, there cannot be assumed that members of the system, either have access to an NTP server, or are equipped with an NTP daemon.

A promising approach to tackle this kind of problems is to embrace a fully decentralized paradigm in which peers implement all the required functionalities, by running so called $gossip-based$ algorithms [14].

In this paper, in order to attain clock synchronization, we propose a novel algorithm that combines the gossip-based paradigm with a nature-inspired approach stemming from the *coupled oscillators* phenomenon. This phenomenon shows enormous systems of oscillators spontaneously locking to a common phase, despite the inevitable differences in the natural frequencies of the individual oscillators. Examples from biology include network pacemaker cells in the heart, congregations of synchronously flashing fireflies and crickets that chirp in unison. A description of the phenomenon was pioneered by Winfree [42]. He mathematically modeled a population of interacting oscillators and discovered that assuming nearly identical individual frequencies and a certain strength of the coupling (which is a measure of the sensitivity each oscillator has to interactions with others), a dramatic transition to a globally entrained state, in which oscillators freeze into synchrony, occurs. A valuable contribution has been subsequently introduced by Kuramoto [23] who simplified the Winfree model by considering the coupling strength constant for all oscillators and depending only on their phase difference. Both Winfree's and Kuramoto's work was done assuming that each oscillator is coupled directly and equally to all others, which means assuming a fully connected oscillators network. However a considerable amount of work has been done also on so called "non-standard topologies". Satoh in [34] performed numerical experiments comparing the capabilities of networks of oscillators arranged in two-dimensional lattices and random graphs. Results showed that the system becomes globally synchronous much more effectively in the random case. In fact, Matthews et al. in [29] note that the coupling strength required to globally synchronize oscillators in a random network is the same as the one required in the fully interconnected case.

In our algorithm we adapt the Kuramoto model to let a very large number of computer nodes deployed over an overlay network to synchronize their software clocks without an ex-

A short and preliminary version of this paper appeared in Proceedings of OTM Conferences, pp. 701-716, 2007.

R. Baldoni, L. Querzoni, S. Scipioni and S. Tucci Piergiovanni are with the Department of Computer and Systems Sciences, Sapienza University of Rome, Rome.

A. Corsaro is with PrismTech, Marcoussis, France.

ternal time reference. More specifically, each process in a node managing the software clock will periodically synchronize, this clock with the software clocks of a set of neighbors chosen uniformly at random from the entire population of nodes. The first issue we tackle is how to artificially reproduce the physical phenomenon in a computer network in which every software clock can be influenced by other clocks only by exchanging messages. In our approach, each node explicitly asks software clock values from neighboring processes in order to calculate their difference in phase. Then, following a Kuramoto-like model, these differences in phase are combined and multiplied by a so-called *coupling factor*, expressing the coupling strength, in order to adjust the local clock.

In our algorithm we adopt an *adaptive* coupling factor, with the objective of reducing the impact of system dynamics (such as node churn) while still keeping the time to converge small. The idea behind the adaptive coupling factor is simple: the local coupling factor reflects the age of a node (expressed by the number of adjustments already performed) in order to let young nodes quickly absorb values from other nodes, while limiting the sensitivity of old node to system perturbations. The rationale behind this mechanism comes from the observation that an old node is more aligned to the values of other clocks than a new one. The adoption of adaptive coupling leads the system to maintain its stability, a property strongly needed in face of system dynamics. Finally, our algorithm adopts a mean-based convergence function to compute the adjustement of the local clock value. Contrarily to median-based functions, mean-based functions are well suited to environments where network delays are unbounded (such as internet based overlay networks), as shown in the evaluation section, because they average all the clock values received from its neighbors during a synchronization.

The rest of the paper is organized as follows: Section II presents the system model describing the internal clock synchronization problem and the details of the node architecture. Section III presents the clock coupling model, the details of the algorithm and describes how we derive the discreet convergence function employed within our algorithm from the original Kuramoto's continuous time convergence function. An experimental evaluation based on simulations is presented in Section IV. First the convergence time is evaluated in a setting with no churn and no clock reading errors, then we evaluate the impact of clock reading errors on the overall synchronization error by considering various scenarios with asymmetric channel delays, message losses and network dynamics. Finally, a comparison between the convergence function introduced in [27] and the one proposed in this paper is shown. Both statistical and simulation-based analysis show the adaptability of our algorithm to a wide range of different settings described by our very general system model. Section V discusses related works, while Section VI concludes the paper.

II. System Model and Node Architecture

Let us consider a distributed system composed of a set of nodes that can vary over time, we denote as $N(t)$ the number of nodes belonging to the distributed systems at time t. Each node may join and leave the system at will. Moreover each node can be correct or crash. In the latter case, when a node crashes it will not recover. In each node several processes might run concurrently.

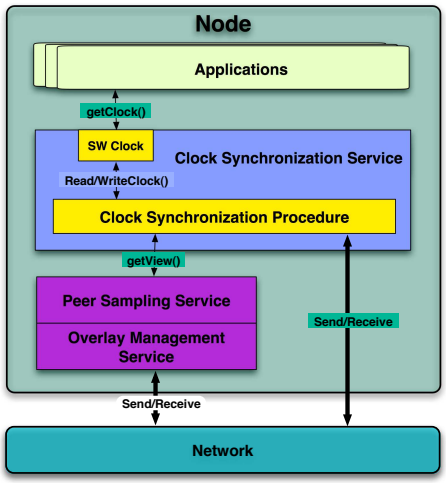


Fig. 1. Node Architecture

Each pair of processes can exchange messages and message delays are unbounded. A message is delivered reliably to destination if both the sender of the message and the receiver process are hosted on nodes that belong to the system at time of the sending and remain both in the system for a time greater than the message delay. Moreover messages can be dropped by the communication channels so also a communication between processes can suffer message omissions.

A. Hardware and Software Clocks

Every node n_i is equipped with a hardware clock consisting of an oscillator and a counting register that is incremented at every tick of the oscillator. Depending on the quality of the oscillator, and the operating environment, its frequency may drift. Manufacturers typically provide a characterization for ρ – the maximum absolute value for oscillator drift. Ignoring, for the time being, the resolution due to limited pulsing frequency of the oscillator, the hardware clock implemented by the oscillator can be described by the following equation:

$$C_H(t) = ft + C_0$$

where: $(1-\rho) \leq f \leq (1+\rho)$.

Moreover each node endows a software clock. This software clock is managed by a process that executes the sum of the current value of n_i's hardware clock and a periodically determined *adjustment* $A(t)$. Consequently each software clock C_i is also characterized by a frequency $f_i \in [1-\rho, 1+\rho]$ and by the following equation:

$$C_i(t) = f_i t + C_0 + A(t)$$

Initially the software clocks of nodes are not synchronized, meaning that they might show different time readings following an unknown distribution. Also any node joining the distributed system at a certain time shows an arbitrary time reading with respect to other nodes already in the system.

B. Internal Clock Synchronization

Internal Clock Synchronization aims to build a "common" software clock among a set of cooperating nodes. In this paper, the "common" clock assumes a value that tries to minimize the maximum difference between any two local software clocks. To do that each node can modify the local software clock by using the adjustment function $A(t)$.

In the internal clock synchronization realized in this paper, the "common" clock represents the mean of the values of the software local clock, namely the *Synchronization Point* (i.e., $SP(t) = \mu(t) = E[C_1(t), \ldots, C_n(t), \ldots]$), of our system, and its aim is to minimize the standard deviation along the time among these local software clocks. Formally,

$$\forall t \ \sigma(t) = \sqrt{\frac{1}{N(t)} \sum_{i=1}^{N(t)} (C_i(t) - \mu)^2} = SE(t) \qquad (1)$$

where $SE(t)$ represents the Synchronization Error at time t i.e., the standard deviation, computed at time t, of software clock values of nodes belonging to the system at that time. Therefore the smaller $SE(t)$ the more accurate is the synchronization among the nodes.

C. Node Architecture

Node architecture is depicted in Figure 1. We consider each node endows a *Clock Synchronization Service* whose aim is to provide local applications with a software clock synchronized with other nodes belonging to the distributed system. To do that, the *Clock Synchronization Service* working on distinct nodes interacts through an existing network infrastructure, that is usually represented by a WAN, and leverages a peer sampling service [20] provided by an overlay management protocol.

The Overlay Management Protocol is a logical network built on top of a physical one (usually the Internet), by connecting a set of nodes through some links. A distributed algorithm running on nodes, known as the Overlay Maintenance Protocol (OMP), takes care of managing these logical links. Each node usually maintains a limited set of links (called view) to other nodes in the system. The construction and maintenance of the views must be such that the graph obtained by interpreting nodes as vertices and links as arcs is connected and keeps some topology. In this manner an overlay management service can realize either deterministic graphs (e.g. a ring) [33], [36] or random graphs [13], [41]. Usually the first are called structured overlay networks and the latter unstructured ones.

The *Peer Sampling Service* is implemented over the overlay network and it returns, through a $getView()$ function, to a process a view $V_i(t)$ of nodes in the overlay at time t. If a node crashes, the Peer Sampling Service will not include eventually the crashed node in any view. In particular we assume the presence of an *Uniform* Peer Sampling Service that provides views containing a *uniform random* sample of nodes currently in the distributed system. It has been shown that theoretically uniform peer sampling can be achieved over both structured overlay networks [22] and unstructured ones [28]. As an example, uniform random samples of nodes over an unstructured overlay are provided through either a random periodic exchange of partial content of the view [20], or random walks [28] (a random view is filled passing through the unstructured network following random walks). Due to the fact that practically a pure uniform peer sampling is difficult to implement on top of a computer network, we remove this assumption in some simulation tests contained in Section IV and assume that peer sampling follows a power law distribution.

Clock Synchronization Service maintains a software clock. It is composed by a set of processes, running at each node, that executes a *Clock Synchronization Algorithm*. Each process exchanges information with processes hosted by nodes contained in the current view returned by the peer sampling service. The collected information is used to minimize differences between the software clocks of nodes by periodically computing the adjustment value $A(t)$.

III. The General Coupling Based Synchronization Algorithm

In this section we present the mathematical basics underlying the coupling clock synchronization along with the clock synchronization algorithm.

A. Time Continuous Clock Coupling

Coupled oscillator phenomenon, pioneered by Winfree [42] and also described by Kuramoto [23], was initially studied in order to analyze behavior of coupled pendulum clocks, and it was subsequently extended to describe a population of interacting oscillators like hardware clocks. Recently this paradigm founds a novel utilization in the analysis of enormous systems of oscillators: network pacemaker cells in the heart, congregations of synchronously flashing fireflies, etc... Assuming a certain strength of the coupling(i.e. of the sensitivity each oscillator has to interactions with others), these enormous systems of oscillators are able to lock to a common phase, despite the differences in the frequencies of the individual oscillators. In a network of coupled oscillator clocks, thanks to a continuous coupling of these clocks over time, they will lock to a so-called stable point: each clock will show the same value, without changing the value once reached.

Even though our coupling resembles the model proposed by [23], [37], it is worth noting that Kuramoto modeled a non-linear oscillator coupling which is not directly applicable to our problem. In fact, the non-linear oscillator used by Kuramoto to model the emergence of fireflies flashing synchrony, models intentionally a phenomenon which is characterized by several stable points (which arise due to the sinusoidal coupling), *i.e.*, the system does not converge to a unique point, but it can partition in subsystems each with a different stable point. On the other hand, for synchronizing clocks in a distributed system it is highly desirable that a single point of synchronization exists. This leads to consider a *linear* coupling equation of the form:

$$\dot{C}_i(t) = f_i + \frac{\phi_i}{|V_i(t)|} \sum_{j=1}^{|V_i(t)|} (C_j(t) - C_i(t)), \quad i = 1..N(t) \ (2)$$

The intuition behind Equation 2 is that a software clock has to speed up if its neighboring clocks are going faster, while it has to slowdown when they are going slower. The coupling frequency ϕ_i provides a measure of how much the current clock rate should be influenced by others. It can be shown analytically that Equation 2 has a single stable fixed point, and thus converges, in the case in which all the clocks are connected to each other. Even with clocks not directly connected to each other, the coupling effect still arises. Provided that the underlying graph is connected, each clock will continue to influence others. In the more general case of non-fully connected graph, Equation 2 can be generalized as follows:

$$\dot{C}_i(t) = f_i + \frac{\phi_i}{|V_i(t)|} \sum_{j \in V_i(t)} [(C_j(t) - C_i(t))], \quad i = 1..N(t) \quad (3)$$

B. Time Discrete Coupling with Imperfect Estimates

The coupling model described in Equation 3 is not directly applicable to distributed systems as it is based on differential equations, and thus continuous time. In fact the physical phenomenon models entities that continually sense other entities, while in a distributed system each node is separated by others through a communication channel showing unpredictable delays. Sensing other entities means requesting explicitly their clock values through a request-reply message pattern. Delays on messages bring to imperfect estimates of clock values to be added in the equation. Before introducing imperfect estimates, let us consider the discrete counterpart of Equation 2 :

$$\begin{aligned} C_i((\ell+1)\Delta T) &= C_i(\ell\Delta T) + f_i\Delta T + \\ &+ \frac{K_i}{N_i} \sum_{j \in V_i} [(C_j(\ell\Delta T) - C_i(\ell\Delta T))] \\ & i = 1..N(t) \\ & \ell = 1 \ldots \end{aligned} \quad (4)$$

Where $K_i = \phi_i \Delta T$ and ΔT is the time interval between two successive interactions.

Let us now add the imperfect estimates of clock offsets due to communication channels. When applying Equation 4 in real distributed systems, the clock difference $(C_j(\ell\Delta T) - C_i(\ell\Delta T))$ will be estimated with an error ϵ which depends on the mechanism used to perform the estimation. In this paper we assume that the difference between neighboring clocks are estimated as NTP does [30], [31] (see Figure 2) by mean of a request-reply message pattern. As in the protocol specification, let t_1 be the p_i's timestamp on the request message, t_2 the p_j's timestamp upon arrival, t_3 the p_j's timestamp on departure of the reply message and t_4 the p_i's timestamp upon arrival, the request message delay is $\delta_1 = t_2 - t_1$ and the reply message delay is $\delta_2 = t_4 - t_3$.

Under this assumption, the real offset between C_i and C_j is such that the error is $(\delta_1 - \delta_2)/2$. Note that, if the two delays are equal (channel symmetry) the error is zero. Moreover, it has been shown that the maximum error is bounded by

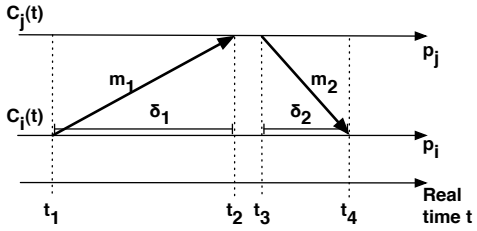


Fig. 2. NTP offset estimation

$\pm(\delta_1 + \delta_2)/2 \approx \pm RTT/2$, where RTT is the round trip time between C_i and C_j. Thus, we can now rewrite Equation 4 by considering the error which affects the $(C_j(n) - C_i(n))$ estimation[1]:

$$\begin{aligned} C_i((\ell+1)\Delta T) &= C_i(\ell\Delta T) + f_i\Delta T + \\ &+ \frac{K_i}{|V_i|} \sum_{j \in V_i} [(C_j(\ell\Delta T) - C_i(\ell\Delta T)) + \\ &+ \frac{K_i}{|V_i|} \sum_{j \in V_i} [(\frac{\delta_{i,j}(\ell\Delta T) - \delta_{j,i}(\ell\Delta T)}{2})] \\ & i = 1..N(t) \\ & \ell = 1 \ldots \end{aligned} \quad (5)$$

C. Algorithm description

A pseudocode description of clock synchronization algorithm implementing the equation 5 is given in Figure 3. The algorithm runs at each synchronization process p_i in order to synchronize its software clock C_i with other software clocks. The algorithm works on the graph defined by process views and computes C_i periodically, every ΔT time units. As a result, the algorithm at any process p_i proceeds in synchronization rounds, performing at every round the following steps:

1) select $|V_i|$ neighbors to synchronize with through the function $getView()$ ($Clock\_Sync()$ line 2). As we described in previous section, $getView()$ is provided by the Peer Sampling Service.
2) estimate the difference with every neighboring clock and with itself by mean of $getOffset()$ function ($Clock\_Sync()$ line 4). We assume $getOffset()$ estimates clock differences as NTP does and as we described in previous section .
3) sum the differences and multiply by $\frac{K_i}{|V_i|}$ ($Clock\_Correction()$ line 1).
4) update the value of C_i, with the new adjustment $A(t)$ computed by $Clock\_Correction()$, and the value of K_i, in the case it is time dependent ($Clock\_Sync()$ lines 6-8).
5) increase the value of the Age_i, reflecting the number of successive synchronizations already performed.

In particular the adaptive K_i is given by the following equation

[1]Let us note that if we consider the worst case bound on estimate error, slow channels (large RTT) may introduce more noise than fast channels (small RTT), however, it is important to keep in mind that the source of error is not the RTT per se, but the asymmetry, *i.e.*, the difference between δ_1 and δ_2.

$Double\ K_i$;
function void $init()\{$
1: $Age_i = 0$;
2: $schedule(Clock\_Sync(), C_i, \Delta T)$;
3: $\}$
function void $Clock\_Sync()\{$
1: $Offset_i = \emptyset$;
2: $V_i = getView()$;
3: **for all** $j \in V_i$ **do**
4: $\quad Offset_i = Offset_i \cup (j, getOffset(j))$;
5: **end for**
6: $C_i = C_i + \Delta T + Clock\_Correction(Offset_i)$;
7: $K_i = Compute\_K()$;
8: $Age_i + +$;
9: $\}$
//Compute A(t)
function Time $Clock\_Correction(O)\{$
1: **return** $\frac{K_i}{|V_i|} * (\sum_x O(j,x), \forall j \in V_i)$
2: $\}$

Fig. 3. Internal Clock Synchronization Algorithm running at process p_i

$$K_i(t) = max(e^{-d_i * max(Age_i - s_i, 0)}, K_{i,min}) \qquad (6)$$

where $K_{i,min}$ is the minimum value that can be assumed by the adaptive coupling factor; d influences the exponential decay time (i.e. the time employed by the adaptive coupling factor to reach the value $K_{i,min}$); finally s_i defines the Age_i value after which the exponential decay starts. Therefore young nodes have a value of K_i close to 1 absorbing thus values from other nodes while old nodes have a value of K_i equal to $K_{i,min}$ limiting thus the update of the local clock due to other remote clocks and network delays.

IV. Experimental evalutation

This section aims at evaluating through a simulation study the behaviour of the proposed coupling algorithm in large scale scenarios. Scenarios are defined with the aim of isolating some specific aspects (e.g. churn, coupling factor effect, view size, etc.). Tests were run on Peersim, a simulation software specifically designed to reproduce large-scale scenarios[2].

In this section we will evaluate the algorithm using the following metrics:

a) Synchronization Error: We will consider here the synchronization error as previously defined in Section II.

b) Convergence Time: The convergence time metric is defined as the number of synchronization rounds (SR) needed by the system to converge to a desired synchronization error. More specifically, in our tests we will measure the number of synchronization rounds taken to reach a synchronization error equal to $10\mu sec$. It should be noticed that we consider SR as the only convergence time metrics as, once the duration ΔT of a synchronization round has been fixed, the time to reach a predefined clock dispersion only depends on the number of synchronizations.

c) Stability: The stability metrics applies only in dynamic scenarios, i.e. scenarios with node churn. The stability metric measures how much clocks already synchronized are sensitive to the injection of new nodes. A perfectly stable algorithm should not allow these clocks to significantly change the convergency value already reached.

A. Simulation settings

The proposed algorithm is evaluated against the metrics and the scenarios described below.

1) Simulation Scenarios:

a) System with no churn and symmetric channels: This scenario corresponds to a system in which (1) the network is static (no nodes are added/removed during the simulation), (2) the network delay is unbounded, (3) communication channels are symmetric $\delta_1 = \delta_2$ and thus no estimation error occurs and (4) processes execute the algorithm periodically every ΔT time units, but not in a lock step mode. The round-trip time (RTT) is modeled by means of a Gaussian distribution whose mean and standard deviation have been derived, as described in [4] by fitting several round-trip data set measured over the Internet. To be more precise, in this scenario we consider two different kind of channels, slow and fast. Slow channels are characterized by an average round trip delay of $180msec$, while fast channels are characterized by an average round trip delay of $30msec$. When generating a communication graph links between nodes are randomly chosen to be of one kind or another. ΔT is the third quartile of the slow channels RTT Gaussian distribution. This model is worth considering as it closely matches the model underlying Equation 4.

b) System with no churn, asymmetric channels and message losses: This simulation scenario adds to the previous one communication channel asymmetry. Channel asymmetry defines how the round-trip time is distributed between the two ways of a channel (*i.e.* given a channel connecting A to B, which is the ratio between the transfer delay of a message from A to B and the delay back from B to A). The asymmetry is modeled by means of a Gaussian distribution with mean 0.5 (*i.e.*, symmetric channel). The parameters of this distribution are used in order to explore the sensitivity of the algorithm to channel asymmetry, and thus to estimate clock difference errors. Finally in this scenario we explore the behavior of the algorithm under message losses introducing different percentile, in the set $1\%, 5\%, \ldots, 20\%$, of message dropped by communication channels.

c) System with churn and Symmetric Channels: The third scenario considered in our tests takes into account network dynamics, and thus considers the evolution of a network under the continual addition/removal of nodes. In order to characterize only the dependency of the proposed algorithm under dynamics, we ignore the estimation errors on clock differences, thus assuming symmetric channels.

d) System with no churn, non-uniform peer sampling and symmetric channels: In the last scenario we remove the

[2]The simulator code for the coupling mechanism is available for download at the following address: http://www.dis.uniroma1.it/~midlab/clocksync_sim.zip

assumption of the presence of an Uniform Peer Sampling Service in each node, replacing it with a Non Uniform Peer Sampling Service that provides views containing biased random samples of nodes currently in the distributed system. We assume the Non Uniform Peer Sampling Service follows a power-law distribution and in order to explore the sensitivity of the algorithm to this non uniform peer sampling we also assume symmetric channels.

2) Simulation Parameters: Tests have been conducted varying the number of clocks N, the size of the local views $|V_i|$, and the coupling factor K. We assume in the follows, as in previous section, that every node has the same view size and $|V_i| = v, \forall i = 1, \dots N$. In order to test our approach under different system scales, ranging from very small to very large, we will be considering values of N in the set of $\{8, 16, \dots, 64K\}$. To show the dependency with respect to a constant coupling factor K we will consider values in the set $\{0.1, 0.2, \dots, 1\}$. Tests aimed at evaluating the adaptive coupling factor will consider at each node the following values for the parameters characterizing the Equation 6: $K_{min} = 0.1$, $d = 0.2$ and $s = 5$. The dependency with respect to the size of local views is showed considering different values in the set $\{5, 10, \dots, 100\}$. Specific tests aimed at evaluating the dependency of the synchronization error on channel asymmetry have been conducted varying the amount of asymmetry, either using a fixed value in the set $\{0.1, \dots, 0.5\}$, or varying the variance σ_A^2 of the Gaussian distribution used to model it within the set $\{10^{-3}, \dots, 10^{-11}, 0\}$. This value describes how message delay is distributed between request and reply messages employed to estimate a remote clock (See Section III-B). More specifically for an asymmetry value a the request message suffers a delay $\delta_1 = a * RTT$, while the reply introduces a delay $\delta_2 = (1-a) * RTT$. Tests for the dynamic symmetric scenario have been conducted varying either the size of the stable core, *i.e.* the amount of nodes that remain in the system from the beginning to the end of the test, or the amount of replaced nodes for a single time unit. Moreover another important parameter is the shape of pareto distribution that describes the distribution of nodes in processes' views in tests aimed at evaluating the impact of a Non Uniform Peer Sampling Service on the synchronization algorithm. We varying the shape within the set $\{0.1, 0.2, \dots, 2\}$ and consequently we consider different levels of bias in peer sampling.

In tests for the static scenario with symmetric channels, different initial distributions are used to show that the synchronization error decay does not depend on the type of the initial clock values distribution but only on the variance of clocks, as expected from the application of a random choice of neighbors. For this reason, in all successive tests we assume that the initial value assumed by a clock, referred as X_0, is a uniform random number in the interval $[0, 60]$ sec.

B. System with No Churn and Symmetric Channels

In this setting, we show how the convergence time, of the proposed algorithm, depends on the system size N, the view size $|V_i|$, and on the coupling factor K – the case of K adaptive is also considered – while it does not depend on the initial distribution of clock values.

Note also that as the communication channels are symmetric, and thus the neighboring clock estimate is perfect, the synchronization error will tend to zero as a negative exponential (this comes from Equation 2) for a number of synchronization rounds that goes to infinity. This is true for any coupling factor K.

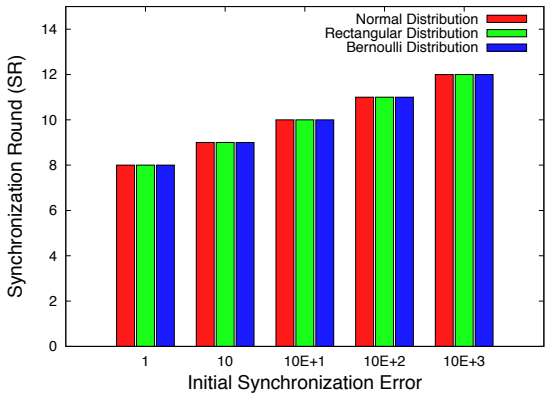


Fig. 4. Convergence time dependency on Initial Distribution.

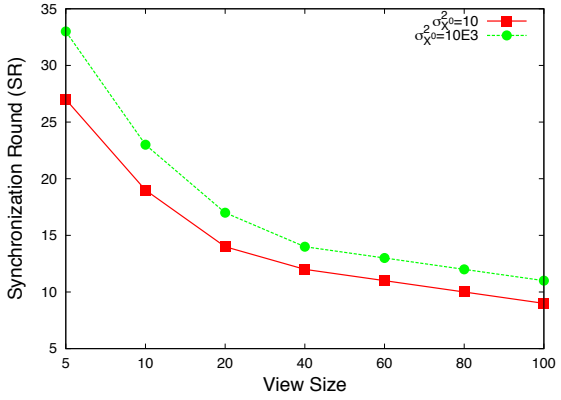


Fig. 5. Convergence dependency on Initial Distribution Variance and View Size.

a) Convergengence of the Mean-based Algorithm: In order to validate these results we analyze several different initial distributions with a fixed size of local views $v = 100$ and $K = 1$ in a system composed by 64K nodes. In Figure 4 we can see the convergence time for different amplitudes of initial SE and for three different distributions: the uniform random, the normal and the bernoulli described by the same mean and variance. How we can see in Figure 4 there is no difference among these scenarios, considering the same initial SE; only increasing the initial SE we can see a larger SR. These simulation results experimentally validate the independence from the shape of initial distribution that comes from the random choice of neighbors at each round. These results have been confirmed also analytically by the study we did in [35] where we have shown that our mean based algorithm converges with a speed (number of synchronization rounds SR) only depending on the initial variance (initial SE), being thus independent of the distribution shape.

In addition, Figure 5 shows how the number of synchronization rounds SR depends on the view size v and the variance of initial distribution. In this test $K = 1$ and $N = 64K$. The test shows as growing the view size we experienced a reduction on the number of synchronization rounds required to reach a SE smaller than $10\mu sec$.

b) System Size, Coupling Factor and Convergence Time: Figure 6, shows how the number synchronization rounds SR depends on the size of the system N, and on the coupling factor K. As it can be seen from the plot, given a value of N, there is a negative exponential dependency of K and SR. This dependency, can roughly be approximated to an inverse dependency between K and SR, as (see Figure 6) doubling K almost halves SR. On the other hand, if we fix the value of K we can see how SR grows slightly with N when $K \geq 0.5$, while it remains constant, or slightly diminishes with N when $K > 0.5$.

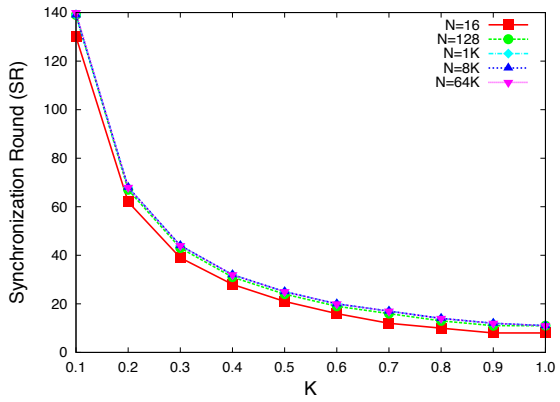


Fig. 6. Convergence time dependency on N and K (Static K).

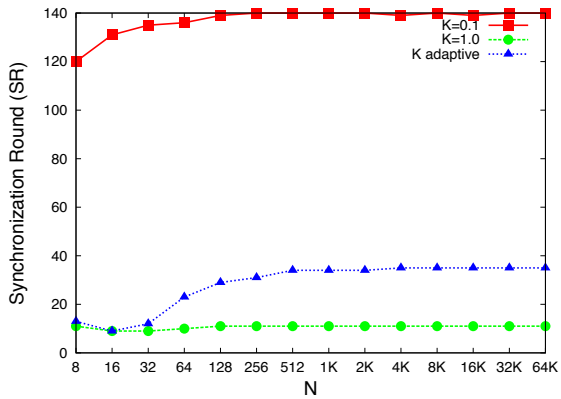


Fig. 7. Convergence time dependency on N and K (Adaptive K).

c) Adaptive Coupling Factor and Convergence Time: Figure 7 compares the effect of an adaptive K on the convergence time with respect a fixed K. To this end, it shows the dependence of SR from the network size for $K = 1$, $K = 0.1$ and K adaptive. From this plot it is easy to see that K *adaptive* provides a performance improvement with respect to the convergence time that is close to what obtained with a fixed $K = 1$.

d) Local View Size, Coupling Factor and Convergence Time: Figure 8 shows how the number of synchronization rounds SR depends on the size of the local view v and on the coupling factor K in a network composed by 64K nodes. From this plot we can see that absolute values of SR depend from K, as we have shown on Figure 6 and 7, but these values only have a small dependency on the size of local views. In fact, only with very small local views we can see an increment of SR required to reach the defined precision and growing the view size, we experienced only a small reduction of SR that does not justify a larger overhead.

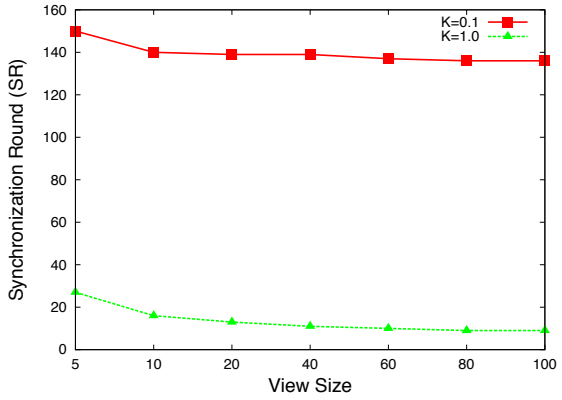


Fig. 8. Convergence time dependency on K and view size.

e) Observations on Algorithm Scalability: Most distributed algorithms tend to degrade their performance as the scale of the systems grows, when this happens, the scale of a system can practically exclude certain algorithmic solutions. Thus, it is extremely important to characterize what happens to a distributed algorithm as the number of nodes involved in the computation grows. To this end Figure 6 and Figure 7 show how the synchronization round SR changes over a very wide set of network sizes. Contrarily to many existing clock synchronization solutions, the proposed algorithm scales extremely well: in fact its performance remain constant with a wide range of network size N with either static and adaptive coupling factor. This is a very important property which makes this solution appealing for applications that need to scale to a very large number of nodes.

C. System with No Churn and Asymmetric Channels Results

In this setting, we investigate how the asymmetry impacts on the synchronization error within which software clocks synchronize. For this scenario we won't show results for the convergence time as these are analogous to what is described in the previous Section.

f) Channel Asymmetry, Coupling Factor and Synchronization Error: Figure 9 reports results obtained using a fixed value of asymmetry for all communication channels. The system size N is fixed to $64K$ for this plot. As the plot shows there is (1) a linear dependency between the channel asymmetry and the synchronization error, and (2) the value of K, as predicted by the Equation 5, behaves as a scaling factor on the synchronization error. The results obtained with the use

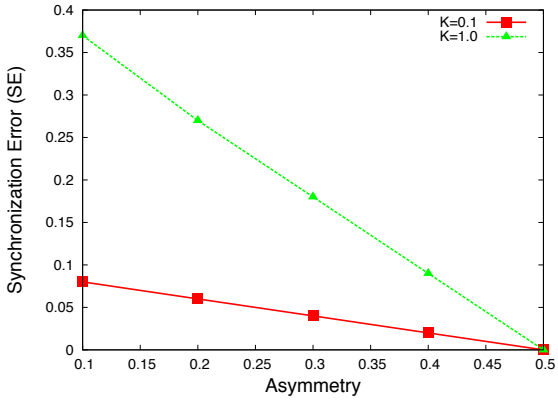


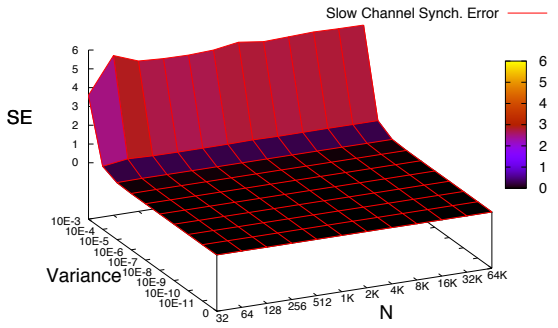
Fig. 9. Synchronization error varying channel asymmetry.

of K adaptive, are not shown as completely overlap with those found for $K = 0.1$. This should come at no surprise as the K after a transitory assumes definitively the value 0.1–the only relevant difference is that, as shown in Figure 7, the use of K adaptive leads to shorter convergence times. Let us finally remark that in these plots we consider clock drift rate is fixed and never changes.

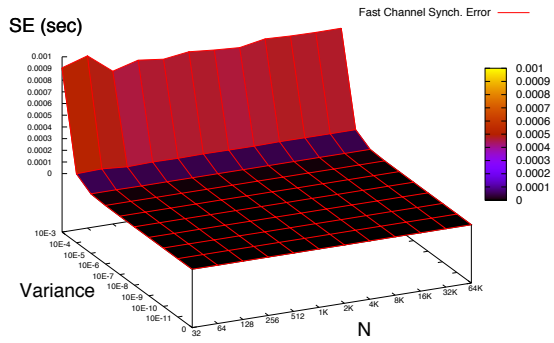
g) Channel Asymmetry, Channel Delays and Synchronization Error: The specific scenario used for the previous plot is far from being realistic, as it assumes a fixed value for asymmetry. Thus, to better model realistic channel asymmetry, we used a Gaussian distribution with mean 0.5 and studied how systems with variable sizes behave with respect to synchronization error, varying the variance σ_A^2 of the distribution. Results for slow channels are showed in Figures IV-C0g, where the synchronization error is reported. As the graph shows, the more channels are "symmetric", *i.e.*, the more the asymmetry variance is low, the lower is the synchronization error with a clear exponential dependency. It is interesting to point out that the error difference between slow and fast channels quickly becomes negligible as soon as we consider fairly symmetric channels as we showed in [3]. This plot therefore confirms that the impact of RTT on synchronization error is not straight, but it strongly depends on channel asymmetry.

h) Local Views, Coupling Factor and Synchronization Error: In Figure 11 we can see the dependency of synchronization error on the coupling factor K and the view size v. We consider $K = 0.1$ and $K = 1$. In both cases we can see the same behaviour, i.e. increasing the view size the synchronization error decreases. In other words, the system is more resilient to perturbations induced by the asymmetry of channels. This result directly follows from Equation 5 where errors, with opposite signs, induced by transmission channels are summed and averaged. Moreover, we can see from Figure 11 that larger views have a stronger effect, in terms of reduction of synchronization error, in systems with coupling factor $K = 1$ where the synchronization error is five times smaller with a view of size 100 than with a view of size 5.

i) Local Views, Message Losses and Synchronization Error: Figure 12 describes the behavior of our system in



(a)



(b)

Fig. 10. Synchronization error for slow and fast channels with a Gaussian asymmetry distribution.

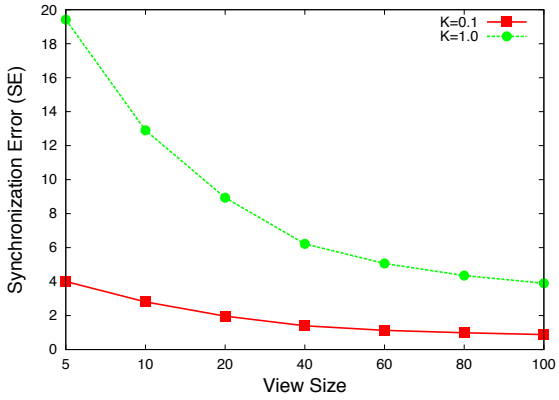


Fig. 11. Synchronization error dependence on K and view size.

presence of different percentages of message loss. The plots show that also in presence of large percentages of message loss (up to 20% of the messages are not delivered) the synchronization error remains almost unchanged. Interestingly, a message loss during a synchronization round is equivalent to a "virtual" reduction of the view size for that round, e.g. the synchronization error in a system with view size 100 and 20% of message losses is roughly equivalent to the one experienced in a system with view size 80 and 0% message losses. Due to lack of space we do not report the case of crashes of processes.

However as message losses, crashes do not impact significantly the synchronization error.

In addition, Figure 13 shows the behaviour of our system for different view sizes and different asymmetry variances σ_A^2 (considering slow channels).

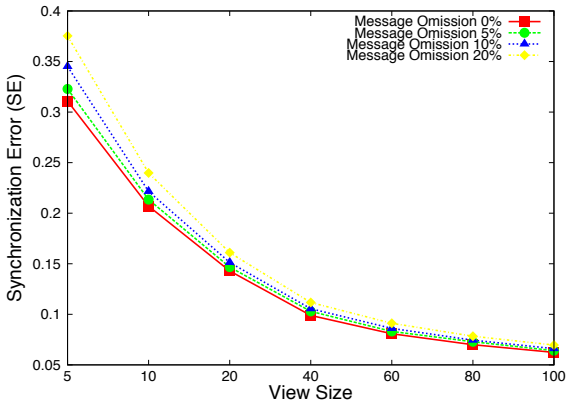


Fig. 12. Synchronization error varying the percentage of message losses and view size.

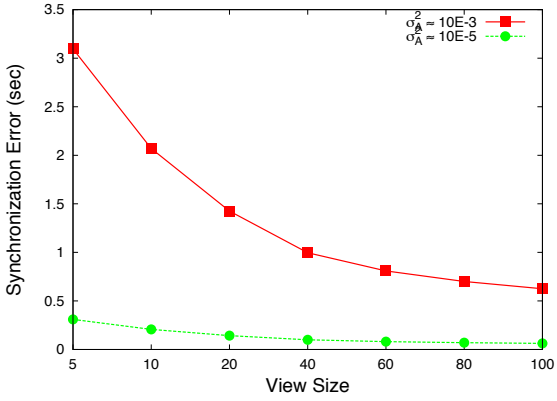


Fig. 13. Synchronization error varying remote clock reading variance and view size.

D. System with Churn and Symmetric Channels

In this setting, we investigate how node churn impacts on the synchronization error within which clocks synchronize, as well as on the stability of clock values.

j) Churn, Coupling Factor and Core Synchronization Error: First we evaluated the resilience of our solution with respect to a continuous addition/removal of nodes. In this test we built a system with $64K$ nodes and considered a fixed *core* made up of nodes that remain in the system for the whole simulation. [3] Churn is modeled by substituting 1% of the system population at each time unit. Then we evaluated how standard deviation of clocks residing in these nodes varies when the remaining part of the system keeps changing. The evaluation was done for two extreme values of the coupling factor K (*i.e.* $K = 0.1$ and $K = 1$), and also using an adaptive

[3]Several studies on peer lifetime in P2P applications confirms this model [18].

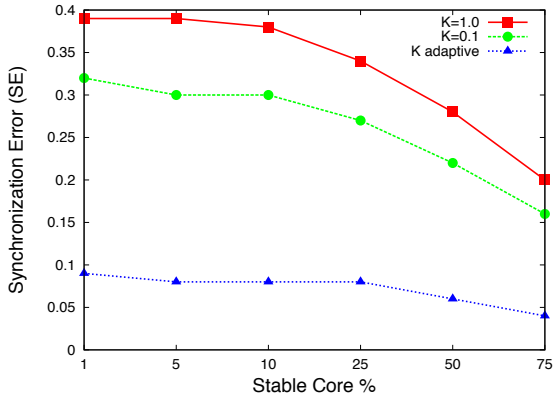


Fig. 14. Synchronization error dependency on core network size and K value.

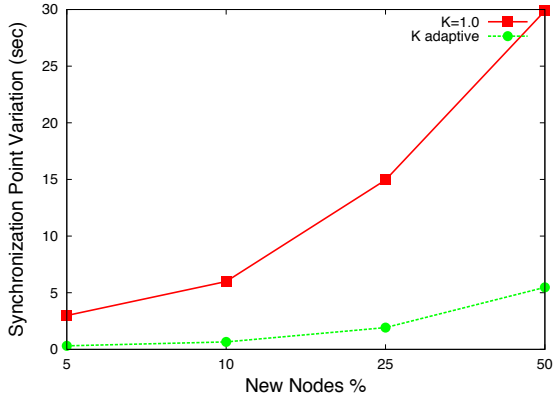


Fig. 15. Stability versus the number of new injected nodes and K value.

K value. Curves reported in Figure 14 show that the core size has a relevant impact on synchronization error as long as we consider a fixed K value. Intuitively, the larger is the core, the less nodes pertaining to it are prone to change their clock due to reads done on newly joined nodes. In this case, by adopting a small value for the coupling factor, nodes belonging to the fixed core are more resilient to node dynamics. More interesting is the behavior of the system when we adopt the adaptive K strategy. In this case, new nodes enter the system using a large K value and therefore rapidly absorb timing information from nodes in the core, while these are slightly perturbed.

k) Churn, Coupling Factor and Stability: Figure 15 shows the stability of the system to a perturbation caused by the injection of a huge number of new nodes. In order to better show this behavior we introduced during the simulation in a network made up of $64K$ nodes, all with clocks synchronized on a specific value, a set of new nodes (expressed in the graph as a percentage of the original network size). Newly injected nodes start with a clock value that is distant 60 seconds from the synchronization point of nodes in the original system. The plot shows how the synchronization point varies from the original one (the distance between the synchronization points is reported in seconds on the y axis) when the new network converges again. If we assume $K = 1$ the system is prone to huge synchronization point variation, that, intuitively,

is larger if a larger number of nodes is injected. However, the introduction of K adaptive mechanism drastically reduces this undesired behavior, limiting the variation of synchronization point, even when the amount of nodes injected is close to 50% of the original system. These results justify the introduction of an age-based adaptive mechanism for the coupling factor value tuning as an effective solution to improve the stability of systems in face of node churn.

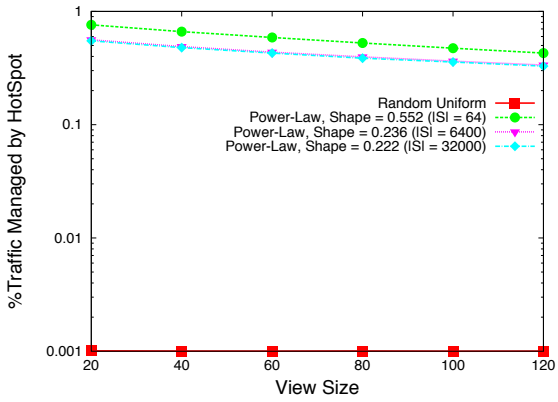


Fig. 16. Percentile of traffic managed by a hot spot of 64 nodes in a system of 64000 nodes.

E. System with No Churn, Non Uniform Peer Sampling and Symmetric Channels

In this setting we simulated that the Peer Sampling Service chooses nodes, in the set of currently active ones, following a power-law distribution. The importance of this analysis is strictly related to difficulty, showed by several algorithms, of realizing a uniform peer sampling service in a dynamic environment where it can be likely that a peer sampling service selects some nodes with higher probability than other ones [2], [22]. In this scenario we analyze different shapes of the power-law distribution corresponding to different sizes of the sets of nodes chosen with high probability. Formally, we noted with S, the set of nodes chosen by the Peer Sampling Service with a probability $P > 0.9$. Let us remark that if the peer-sampling follows a power-law distribution, it introduces actually a *core* of nodes, able to synchronize very fast among themselves, and acting like central time servers with respect to other nodes. Therefore, from a synchronization error viewpoint, this creation of a core of nodes in the system would create some benefits in terms of convergence speed. Unfortunately, this advantage is only virtual as communication channels bringing to a node belonging to the core can rapidly get congested. This node will be indeed present in many of the local views of other nodes exchanging thus a huge number of messages per round. To point out this behavior, Figure 16 shows results obtained analyzing the percentile of the whole traffic managed by a small subset, namely the *hot spot*, of nodes currently in the distributed system for different shapes of the power-law distribution. These values are compared with the traffic managed by nodes belonging to the hot-spot under the presence of uniform peer sampling service. In our test the hot spot is composed by the most frequent 64 nodes (out of 64000 composing the entire system) as they are returned by the Peer Sampling Service. As depicted in Figure 16 if the peer-sampling follows a power-law distribution each node in the hotspot manages three orders of magnitude more messages than the ones managed by nodes in the presence of uniform peer-sampling (almost independently of the shape of the power-law). Actually nodes composing the hot-spot manage up to 80% of the whole traffic generated by the clock synchronization algorithm. This critical overhead will bring nodes in the core getting congested and then unusable for synchronization purposes, destroying thus the synchronization.

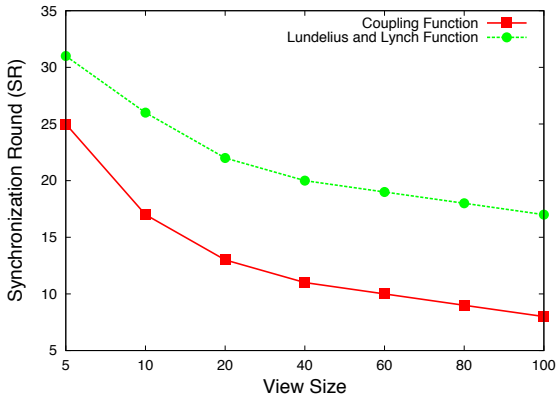


Fig. 17. Convergence time dependency on N and Convergence Function

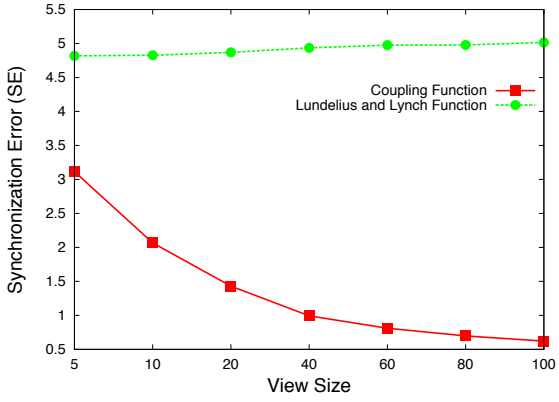


Fig. 18. Synchronization Error dependency on N and Convergence Function

F. System with no churn and different convergence functions

Many different convergence functions have been proposed in the literature for algorithms tailored to LAN-based distributed systems (e.g. [7], [27]). These functions can be, in principle, embedded also in our algorithm (introduced in Section III) as they are usually independent from the setting where they were originally proposed. The aim of this section is thus to propose a comparison between our convergence function and the one proposed by Lundelius and Lynch (LL) [27] in order to test their convergence speed and the overall synchronization quality in the system model proposed in

Section II[4]. Differently from our convergence function, the function proposed in [27] employs a midpoint-based convergence function to reach a synchronization point. In this way the new adjustment is obtained computing the midpoint of the interval of offsets estimated in a synchronization round. Figure 17 reports the synchronization round for both functions in the same ideal setting as the one proposed in Section IV-A1a. The plots show how our convergence function is able to converge faster than the LL function in absence of network delays and how its efficiency increases with the view size.

Introducing network delays changes the behavior of the two convergence functions in two different ways: our function, as previously discussed (see Section IV-C0g), is able to improve its resilience to network delays as the view size grows converging to an asymptotical synchronization error value, while the LL function provides slightly increasing synchronization errors as the view size grows. This behaviour is due to the median-based approach used in the LL function versus the average approach used in our function. The median approach was adopted in [27] as their function was designed to work in a system model where network delays are bounded and, thus, remote clock reading errors are bounded as well. This assumption is no longer true in our system model where channel delays are modelled with a gaussian distribution and are therefore unbounded. The median approach adopted in the LL function uses at each synchronization round only the extreme values to calculate the clock correction; due to the channel delay model, the extreme values report clock readings with possibly unbounded errors and thus cause larger errors in the final clock correction value. Increasing the view size only makes bad things go worse: the more are the remote readings performed in a single synchronization round, the larger is the probability to perform reads with huge errors.

V. Related Work

A work that follows an approach similar to ours is presented in [21]. In this paper, gossiping is used for computing aggregate values over network components in a fully decentralized fashion. Differently from our work, Babaoglu et al. do not face the clock synchronization problem but they only analyze the properties of the class of aggregate functions that their solution can compute(e.g. counting, sums and products). Moreover they do not show the behavior of their solution in the presence of churn and in the presence of non-uniform peer-sampling services.

We can divide clock synchronization algorithms in two main classes: deterministic and probabilistic. Deterministic clock synchronization algorithms [8], [9], [12], [15], [24]–[27] guarantee strict properties on the accuracy of the synchronization but assumes that a known bound on message transfer delays exists. Lamport in [24] defines a distributed algorithm for synchronizing a system of logical clocks which can be used to totally order events, specializes this algorithm to synchronize physical clocks, and derives a bound on how far out of synchrony the clocks can go. Following works of Lamport and Melliar-Smith [25], [26] analyze the problem of clock synchronization in presence of faults, defining Byzantine clock synchronization. Some deterministic solutions, such as those proposed in [7], [10], [11], [26], prove that, when up to F reference time servers can suffer arbitrary failures, at least 2F+1 reference time servers are necessary for achieving clock synchronization. In this case, these solutions can be fault-tolerant also for Byzantine faults. Currently, we do not analyze byzantine-tolerant behavior of our solution. The deterministic approach, normally tuned to cope with the worst case scenario, assures a bounded accuracy in LAN environments but loses its significance in WAN environments where messages can suffer high and unpredictable variations in transmission delays. Several works of Dolev et al. [10]–[12], [16] propose and analyze several decentralized synchronization protocols applicable for WAN but that require a clique-based interconnecting topology, which is hardly scalable with a large number of nodes.

Clock synchronization algorithms based on a probabilistic approach were proposed in [1], [6]. The basic idea is to follow a master-slave pattern and synchronize clocks in the presence of unbounded communication delays by using a probabilistic remote clock reading procedure. Each node makes several attempts to read a remote clock and, after each attempt, calculates the maximum error. By retrying often enough, a node can read the other clock to any required precision with a probability as close to 1 as desired. This implies that the overhead imposed by the synchronization algorithm and the probability of loss of synchronization increases when the synchronization error is reduced. The master-slave approach and the execution of several attempts are basic building blocks of the most popular clock synchronization protocol for WAN settings: NTP [30], [31]. NTP works in a static and manually-configured hierarchical topology. A work proposing solutions close to NTP is CesiumSpray [40] that is based on a hierarchy composed by a WAN of LANs where in each LAN at least a node has a GPS receiver. These solutions require static configuration and the presence of some nodes directly connected with a external time reference in order to obtain external time synchronization. Finally, a probabilistic solution based on a gossip-based protocol to achieve external clock synchronization is proposed in [19]. Each node uses a peer sampling service to select another node in the network and to exchange timing information with. The quality of timing information is evaluated using a dispersion metric like the one provided by NTP.

VI. Concluding remarks

Clock synchronization for distributed systems is a fundamental problem that has been widely treated in the literature. However, today's large scale distributed applications spanning from cloud computing, managing of large scale datacenters to millions of networked embedded systems, pose new issues that are hardly addressed by existing clock synchronization solutions (hardly relying, for example, on fixed numbers of processes). These systems require the development of new approaches able to reach satisfying level of synchronization while providing the desired level of scalability.

[4]Note that, a comparison between our algorithm and the one proposed in [27] would be meaningless due to the striking differences between the two system models.

In this paper we proposed a novel algorithm for clock synchronization in large scale dynamic systems in absence of external clock sources. Our algorithm stems from the work on coupled oscillators developed by Kuramoto [23], adequately adapted to our purposes. Through experimental study based on simulations we showed that our solution is able to converge and synchronize clocks in systems ranging from very small to very large sizes, achieving small synchronization errors that strictly depend on the quality of links used for communication (with respect to delay and symmetry). Our solution, thanks to the employment of an adaptable coupling factor, is also shown to be resilient to node churn. We analyzed the impact of having a non-uniform peer sampling service on the synchronization error of our solution. We showed that this is a critical issue because as soon as the peer-sampling follows a power-law distributions, there will be the formation of a core of nodes that could rapidly becomes congested being then unusable to the synchronization activities. Finally the paper has also shown that in system models where network delays are unbounded, a mean-based convergence function reaches a lower synchronization error than median-based convergence functions exploiting the number of averaged clock values.

ACKNOWLEDGMENT

The authors want to thank the anonymous reviewers for comments and suggestions that greatly improved the quality of the paper. The authors are also indebted with Stefano Leonardi and Roberto Beraldi for insignful remarks on an early draft of the work. This work has been partially supported by the FP7-European Project "Comifin" and by a grant CINI-FINMECCANICA.

REFERENCES

[1] K. Arvind. Probabilistic Clock Synchronization in Distributed Systems, IEEE Transaction on Parallel and Distributed Systems, vol. 5(5), 1994.

[2] A. Awan, R. A. Ferreira, S. Jagannathan and A. Grama. Distributed Uniform Sampling in Unstructured Peer-to-Peer Networks. In Proceedings of the 39th Annual Hawaii International Conference on System Sciences, pp. 223-233, 2006.

[3] R. Baldoni, A. Corsaro, L. Querzoni, S. Scipioni, S. Tucci Piergiovanni. An Adaptive Coupling-Based Algorithm for Internal Clock Synchronization of Large Scale Dynamic Systems, In Proceedings of OTM Conferences, pp. 701–716, 2007

[4] R. Baldoni, C. Marchetti, A. Virgillito. Impact of WAN Channel Behavior on End-to-end Latency of Replication Protocols, In Proceedings of European Dependable Computing Conference, 2006.

[5] R. Baldoni, R. Jimenez-Peris, M. Patino-Martinez, L. Querzoni, A. Virgillito. Dynamic Quorums for DHT-based Enterprise Infrastructures, Journal of Parallel and Distributed Computing, 68(9), pp. 1235-1249, 2008.

[6] F. Cristian. A probabilistic approach to distributed clock synchronization. Distributed Computing, 3:146-158, 1989.

[7] F. Cristian and C. Fetzer. Integrating Internal and External Clock Synchronization, Journal of Real Time Systems, Vol. 12(2), 1997

[8] F. Cristian, H. Aghili and R. Strong. Clock synchronization in the presence of omission and performance faults, and processor joins, In Proceedings of 16th International Symposium on Fault-Tolerant Computing Systems, pp. 218-223,1986.

[9] F. Cristian and C. Fetzer. Lower bounds for convergence function based clock synchronization. In Proceedings of the Fourteenth Annual ACM Symposium on Principles of distributed computing, pp.137-143, 1995

[10] A. Daliot, D. Dolev and H. Parnas. Linear Time Byzantine Self-Stabilizing Clock Synchronization, Technical Report TR2003-89, Schools of Engineering and Computer Science, The Hebrew University of Jerusalem, Dec. 2003.

[11] A. Daliot, D. Dolev, H. Parnas. Self-Stabilizing Pulse Synchronization Inspired by Biological Pacemaker Networks, In Proceedings of the Sixth Symposium on Self-Stabilizing Systems, pp. 32-48, 2003

[12] S. Dolev. Possible and Impossible Self-Stabilizing Digital Clock Synchronization in General Graph, Journal of Real-Time Systems, no. 12(1), pp. 95-107, 1997.

[13] P. Eugster, S. Handurukande. R. Guerraoui, A. Kermarrec and P. Kouznetsov. Lightweight Probabilistic Broadcast. In ACM Transactions on Computer Systems, vol. 21(4), pp. 341-374, 2003.

[14] R. Friedman, A. Kermarrek, H. Miranda, L. Rodriguez. Gossip-Based Dissemination. In Middleware for Eccentric and Mobile Applications, Springer, 2009.

[15] J. Halpern, B. Simons and R. Strong. Fault-tolerant clock synchronization, In Proceedings of the 3rd Annual ACM Symposium on Principles of Distributed Computing, pp. 89-102, 1984.

[16] T. Herman and S. Ghosh. Stabilizing Phase-Clock. Information Processing Letters, 5(6):585-598, 1994

[17] C. Hewitt. ORGs for Scalable, Robust, Privacy-Friendly Client Cloud Computing. IEEE Internet Computing, 12(5), 96-99,2008

[18] K. Ho, J. Wu, J Sum. On the Session Lifetime Distribution of Gnutella, International Journal of Parallel, Emergent and Distributed Systems, Vol. 23(1), pp. 1-15, 2008.

[19] K. Iwanicki, M. van Steen and S. Voulgaris. Gossip-based Synchronization for Large Scale Decentralized Systems, In Proceedings of the Second IEEE International Workshop on Self-Managed Networks, Systems and Services, pp. 28-42, 2006.

[20] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, M. van Steen. The peer sampling service: experimental evaluation of unstructured gossip-based implementations, In Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware, pp. 79-98,2004.

[21] M. Jelasity, A. Montresor and O. Babaoglu. Gossip-based aggregation in large dynamic networks, In ACM Transactions on Computer Systems, Vol. 23(3) pp. 219-252, 2005.

[22] V. King, S. Lewis, J. Saia, M. Young. Choosing a Random Peer in Chord, Algorithmica, Volume 49(2), pp. 147-169, 2007.

[23] Y. Kuramoto. Chemical oscillations, waves and turbulence. Chap. 5. Springer-Verlag, 1984.

[24] L. Lamport. Time, clocks and ordering of events in a distributed system. Commun ACM, vol 21, no. 7, pp. 558-565, 1978.

[25] L. Lamport and P. M. Melliar-Smith. Byzantine clock synchronization. In Proceedings of the 3rd Annual ACM Symposium on Principles of Distributed Computing, pp. 68-74, 1984

[26] L. Lamport and P. M. Melliar-Smith. Synchronizing clocks in the presence of faults, Journal of the ACM, 32(1):52-78, 1985.

[27] J. Lundelius-Welch and N. Lynch. A new fault-tolerant algorithm for clock synchronization. In Proceedings of the 3rd Annual ACM Symposium on Principles of Distributed Computing, pp. 75-88, 1984.

[28] L. Massouli, E. Le Merrer, A.-M. Kermarrec, A. Ganesh. Peer Counting and Sampling in Overlay Networks: Random Walk Methods, In Proceedings of the twenty-fifth annual ACM symposium on Principles of Distributed Computing, pp. 123-132, 2006.

[29] P.C. Matthews, R. E. Mirollo and S. H. Strogatz. Dynamics of a large system of coupled nonlinear oscillators. Physica D 52, Vol. 52(2-3), p. 293-331, 1991.

[30] D. L. Mills. Network Time Protocol (Version 1) specification and implementation. Network Working Group Report RFC-1059. University of Delaware, 1988.

[31] D. L. Mills. Network Time Protocol Version 4 Reference and Implementation Guide. Electrical and Computer Engineering Technical Report 06-06-1, University of Delaware, 2006.

[32] Object Management Group. Data distribution service for real-time systems specification v1.2, ptc/2006-04-09.

[33] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pp. 329-350, 2001.

[34] K. Satoh. Computer Experiment on the Cooperative Behavior of a Network of Interacting Nonlinear Oscillators. Journal of the Physical Society of Japan, Vol. 58(6), pp. 2010-2021, 1989.

[35] S. Scipioni, L. Querzoni, S. Tucci Piergiovanni, R. Baldoni. A Theoretical Evaluation of Peer-to-Peer Internal Clock Synchronization. Autonomics Conference, 2008.

[36] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Protocol. In IEEE/ACM Transactions on Networking, Vol. 11(1), pp. 17-32, 2003.

[37] S.H. Strogatz and R.E. Mirollo. Phase-locking and critical phenomena in lattices of coupled nonlinear oscillators with random intrinsic frequencies, Physica D, vol. 31, pp. 143-168, 1988.
[38] C. Tang, R. N. Chang, E. So, A distributed service management infrastructure for enterprise data centers based on peer-to-peer technology. Proceedings of the IEEE International Conference on Services Computing, pp. 52-59, 2006.
[39] C. Tang, M. Steinder, M. Spreitzer, G. Pacifici, A scalable application placement controller for enterprise data centers. Proceedings of the 16th international conference on World Wide Web, pp 331-340, 2007.
[40] P. Verissimo, L. Rodrigues and A. Casimiro. CesiumSpray: a Precise and Accurate Global Time Service for Large-scale Systems, Journal of Real-Time Systems, Vol. 12(3), pp. 243-294, 1997.
[41] S. Voulgaris, D. Gavidia, M. van Steen. CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays. In Journal of Network and System Management, vol. 13(2), pp. 197-217, 2005.
[42] A.T. Winfree. Biological rhythms and the behaviour of populations of coupled oscillators. Journal of Theoretical Biology, vol. 28, pp. 327-374, 1967.

Roberto Baldoni is a Professor in Computer Science at the University of Rome "La Sapienza". He is doing research since twenty year on dependable distributed computing and systems where he published more than two hundred papers in peer reviewed scientific forums. He founded MIDLAB a laboratory on middleware technology and he is the coordinator person of the FT7 project SM4ALL and the Co-Technical Director of the EUproject CoMiFin. In the last years he has been the PC Chair of IEEE SRDS07, the General Chair of ACM DEBS 2008 and the PC chair of the track on "Reliable and Dependable systems" at IEEE ICDCS 2009. Finally he is the director of the Joint-Lab on Security Research at Sapienza Innovazione, the technology Incubator or Univ. La Sapienza.

Angelo Corsaro is a Professor in Computer Science at the University of Rome "La Sapienza". He is doing research since twenty year on dependable distributed computing and systems where he published more than two hundred papers in peer reviewed scientific forums. He founded MIDLAB a laboratory on middleware technology and he is the coordinator person of the FT7 project SM4ALL and the Co-Technical Director of the EUproject CoMiFin. In the last years he has been the PC Chair of IEEE SRDS07, the General Chair of ACM DEBS 2008 and the PC chair of the track on "Reliable and Dependable systems" at IEEE ICDCS 2009. Finally he is the director of the Joint-Lab on Security Research at Sapienza Innovazione, the technology Incubator or Univ. La Sapienza.

Leonardo Querzoni is an assistant professor at University of Rome "La Sapienza". He obtained a PhD in in computer engineering from the same institution with a work on efficient data dissemination though the publish/subscribe communication paradigm. In the recent past he published several peer-reviewed papers in various computer-science fields including large scale and dynamic distributed systems, publish/subscribe data diffusion, wireless and sensor networks. He has been invited to chair the student forum for the 7th European Dependable Computing Conference; he serves in the technical program committees of many conferences in the field of dependability, event-based communications and autonomic systems (DSN, EDCC, DEBS, Autonomics), and he regularly serves as a reviewer for international conferences and journals of his research areas. Being part of the MIDLAB research group, he is currently involved in several international EU-funded projects dealing with the protection of critical financial infrastructures (CoMiFin project) and with dependability and performance in smart-space enabled environments (SM4All, eDIANA and SOFIA projects). His teaching duties includes yearly courses on distributed systems and computer programming.

Sirio Scipioni is a PhD student in Computer Science at the University of Rome "La Sapienza". He is doing research on various computer-science fields including large scale and dynamic distributed systems, publish/subscribe systems and clock synchronization algorithms. In these research fields, he published several papers in peer reviewed scientific forums. As a part of the MIDLAB research group, he is currently involved in an EU-funded project dealing with dependability and performance in smart-space enabled environments (SOFIA project) and he worked to developing dependable distributed systems (ReSIST network of excellence) and to definition of new methodology for eGovernment (eG4M). His teaching duties includes a yearly course on computer programming.

Sara Tucci Piergiovanni is a Professor in Computer Science at the University of Rome "La Sapienza". He is doing research since twenty year on dependable distributed computing and systems where he published more than two hundred papers in peer reviewed scientific forums. He founded MIDLAB a laboratory on middleware technology and he is the coordinator person of the FT7 project SM4ALL and the Co-Technical Director of the EUproject CoMiFin. In the last years he has been the PC Chair of IEEE SRDS07, the General Chair of ACM DEBS 2008 and the PC chair of the track on "Reliable and Dependable systems" at IEEE ICDCS 2009. Finally he is the director of the Joint-Lab on Security Research at Sapienza Innovazione, the technology Incubator or Univ. La Sapienza.