Computer Science and Information Systems 17(1):141–160 https://doi.org/10.2298/CSIS181115017M

Correctness of the Chord Protocol

Bojan Marinković[1], Zoran Ognjanović[1], Paola Glavan[2], Anton Kos[3], and Anton Umek[3]

[1] Mathematical Institute of the Serbian Academy of Sciences and Arts
Beograd, Serbia
[bojanm,zorano]@mi.sanu.ac.rs
[2] Faculty of Mechanical Engineering and Naval Architecture
University of Zagreb, Zagreb, Croatia
pglavan@fsb.hr
[3] Faculty of Electrical Engineering
University of Ljubljana, Ljubljana, Slovenia
[anton.kos, anton.umek]@fe.uni-lj.si

Abstract. Internet of Things (IoT) can be seen as a cooperation of various devices with limited performances that participate in the same system. IoT devices compose a distributed architecture system. The core of every IoT system is its discovery and control services. To realize such services, some authors used the developed solutions from the different domains. One such solution is the Chord protocol, one of the first, the simplest and the most popular distributed protocols. Unfortunately, the application of the Chord protocol was realized using the correctness of the Chord protocol for granted, or by the very hard assumptions. In this paper we prove the correctness of the Chord protocol using the logic of time and knowledge with the respect to the set of possible executions, called regular runs. We provide the deterministic description of the correctness of the Chord protocol and consider Chord actions that maintain ring topology while the nodes can freely join or leave.

Keywords: IoT, DHT, Chord, correctness, temporal logic, epistemic logic

1. Introduction

Internet of Things (IoT) paradigm can be defined as: "The pervasive presence around us of a variety of things or objects which, through unique addressing schemes, are able to interact with each other and cooperate with their neighbors to reach common goals" [1].

In this framework the smart objects, which are connected by a network structure, are able to communicate and exchange information and to enable new forms of interaction among things and people [2]. The core of every IoT system consists of its discovery and control services. It is common that various homogeneous and heterogeneous devices participate in the same IoT system. Usually, these devices are highly distributed, therefore it can be seen as they participate in a distributed Peer-to-Peer (P2P) system.

In a case of homogeneous decentralized distributed P2P system, nodes (peers) run the same application, and share the same properties in terms of computation and storage capacities and network connectivity [3]. Without any centralized control processes are dynamically distributed to nodes that can join or leave the system at any time. Thus, P2P systems are highly scalable, as they have no inherent bottlenecks. Also, such systems are

resilient to failures, attacks, etc., since there is no single node or a group of nodes that implement a critical functionality, which would render the system unusable, if disrupted. P2P systems are used for file sharing, redundant storage, and real-time media streaming.

While the underlying network actually connects devices, P2P systems are frequently implemented in a form of overlay networks, structures which organize system resources in an independent logical topology [4]. Overlay networks can be realized in the form of Distributed Hash Tables (DHTs). A DHT provides a lookup service similar to a hash table. Pairs of the form $\langle key, value \rangle$ are stored in a DHT, while nodes participating in the network can efficiently retrieve the value associated with a given key. The functionality of maintaining the mapping from keys to values is implemented by nodes in a distributed manner and changes in the set of participating nodes cause only minimal amount of disruption.

The Chord protocol [5,6,7] is one of the first, the simplest and the most popular implementations of DHTs. Nodes in a network executing the Chord protocol are organized in a ring. Because of the simplicity and popularity of the Chord protocol, it was used for the realization of the discovery and/or control service of IoT systems described in [2,8,9,10,11]. Although Chord is one of the simplest protocols, since it is distributed, it is subject to faults and bugs, so the protocol verification is extremely important.

In this paper we use a temporal epistemic logic to prove the correctness of the Chord protocol with respect to the Chord actions that maintain ring topology. We consider the case when the nodes are allowed to leave or join the network with respect to the set of possible executions, called regular runs. It means that a network executing a regular run will be always brought from an arbitrary state to a state in which links between nodes form a ring.

Up to our knowledge there were of only a few papers related to formal verification of DHTs, and particularly Chord [12,13,14,15,16]. They are considered them in Section 2 and compared with our approach.

The rest of the paper is organized in the following way: in Section 2 we consider other approaches for proving the correctness of the Chord protocol and clearly present the contributions of this paper; Section 3 presents a short description of the Chord protocol; in Section 4 we present a logical framework which is used to prove the correctness of the maintenance of the ring topology of the Chord protocol with the respect to the regular runs; the proof is given in Section 5; we conclude with Section 6. In Appendix A we provide detailed proofs of the main lemmas and theorems from the paper.

2. Related Work and Contributions

2.1. Related Work

The Chord protocol is introduced in [5,6,7]. The detailed examination of the protocol's performance and robustness is given under the assumption that nodes and keys are randomly chosen. The provided approach is probabilistic, e.g., statements are of the form "With high probability, the number of nodes that must be contacted to find a successor in a N-node network is $O(\log N)$".

The only deterministic given result is [5,6,7, Theorem IV.3] which partially corresponds to our Lemma 6. Theorem IV.3 proves that inconsistent states produced by executing several concurrent joins of the new nodes are transient, i.e., that after the last node

joins the network will form a cycle. A more general sequences of concurrent joining and leaving of nodes is presented in [15]. The paper gives a lower bound of the rate at which nodes need to maintain the system such that it works correctly with high probability. In this paper we consider the case when the nodes are allowed to leave the network with respect to the set of possible executions, called regular runs.

While it is hard to directly compare these two approaches (deterministic and probabilistic), they are complementary and may be used together to improve our understanding of the Chord protocol. One can argue that the probabilistic approach is useful to study robustness of protocols. On the other hand, it will be also useful to describe sequences of actions that lead to (un)stable states of Chord networks, in order to be able to analyze properties of systems incorporating Chord and assuming its correctness, as it is the case with the discovery and/or control service of an IoT system.

The paper [14] uses the theory of stochastic processes to give estimations of the probability that a Chord network is in a particular state. The correctness of the Chord's stabilization algorithm is proved in the framework of π-calculus by showing the equivalence of the corresponding specification and implementation in [12,13], but assuming that nodes cannot leave a network. The correctness of the pure join model is also proved using the Alloy formal language in [16], and some counterexamples to correctness of Chord ring-maintenance in the general case are presented.

As we mentioned in the Introduction, using DHT/Chord in IoT domain is not a novelty [2,8,9,10,11]. In all papers regular discovery (and/or control) service, to improve performances, is replaced by DHT protocol. In [8] authors proposed distributed control plane. They consider the problem how to deliver control messages to the devices that are in sleeping mode most of the time. The proposed DHT algorithm, to realize the *control-plane*, is Chord. The system consists from two type of nodes - peers and clients. Peers are those nodes which participate in the Chord network, and by the assumption they are **stable**. Clients are "normal" nodes that use peers as proxies to connect to the Chord network. Paper [2] introduces scalable, self-configuring and automated service and resource discovery mechanism based on structured DHT architecture to provide support of more complex search queries. Article [9] presents the overview and comparison of the discovery service mechanisms in IoT domain, both traditional and distributed approaches. In [10] authors give the description of a novel discovery service for IoT. In this approach the information repositories are organized in a DHT network to enable multidimensional search procedure. Authors of [11] presented discovery service, that improves scalability, load balance and reliability, for objects carrying RFID tags based on double Chord ring . In all these articles, the **correctness** of the Chord protocol was accepted **for granted**, or by an assumption of stable DHT network.

In [17] a joint frame for reasoning about knowledge and linear time is presented, and the proof of weak completeness for a logic which combines expressions about knowledge with linear time is provided. Related strongly complete logics that concern linear and branching time are presented in [19,20,21]. We use this framework, as a starting point for our logic of time and knowledge. We describe the Chord protocol using the formal language of that temporal epistemic logic and prove properties of the Chord protocol.

2.2. Contributions

The main contributions of this paper are:

- a description of the Chord protocol using a temporal epistemic logic;
- a proof of the correctness of the maintenance of the ring topology of the Chord protocol with the respect to the set of possible executions called regular runs.

This work was motivated by the importance of the discovery and control service of an IoT system and the obvious fact that errors in concurrent systems are difficult to reproduce and find merely by program testing. This proof could be, also, the foundation for the formal proof created using a formal proof assistant (like, Coq or Isabelle/HOL).

3. Chord Protocol

The Chord protocol was introduced in [5,6,7] where its specification in C++-like pseudocode was given. Here we provide a short description of the Chord protocol that is needed to understand the rest of the paper.

Nodes that run the Chord protocol construct a network that is ring-shaped. The main operations supported by the Chord protocol are:

- adding a node to network,
- removing a node from a network, and
- mapping the given key onto a node using consistent hashing.

The *primitive actions* on which the procedure of maintaining the ring topology in the Chord protocol is based are:

- a node starts a network,
- a new node joins a network,
- a node leaves a network,
- a node updates its successor, and
- a periodic check of the predecessor.

The consistent hashing is used because it provides load-balancing, i.e., every node receives roughly the same number of keys, and when nodes join or leave the network only a few keys are required to be moved [18]. Since Chord networks are overlay systems, each node in a network, which consists of N-nodes, needs "routing" information about only $O(\log N)$ other nodes, and resolves all lookups via $O(\log N)$ messages to other nodes.

Nodes are assigned random identifiers and objects are stored across these nodes using consistent hashing. The identifier for a node (a key), $hash(node)$ ($hash(key)$) is obtained by hashing the node's IP address, or the value of the key, respectively. The length of identifiers, e.g., m bits, guarantees that the probability that two objects of the same type are assigned the same identifiers is negligible. Identifiers are ordered in an identifier circle modulo 2^m (see Figure 1), while all arithmetic is preformed modulo 2^m. A key is assigned to a node if their $hash$-values are equal. Otherwise, if there is no node such that $hash(node) = hash(key)$, the key is assigned to the first node in the ring such that $hash(node) > hash(key)$.

Every node stores its current successor and predecessor nodes in the identifier circle. To improve performance of the lookup procedure, every node organizes routing information in the *Finger Table* with up to m entries. The i^{th} entry in the table which belongs to the node n contains the identifier of the first node s such that $s = successor(n + 2^{i-1})$,

where $1 \leqslant i \leqslant m$. In other words, the node s succeeds the node n by at least 2^{i-1} in the identifier circle. Figure 1 presents Finger tables of nodes n_2, n_{37}, n_{50} and n_{56}.

A node can be aware of only a few other nodes in the system. For example, node n_2 from Figure 1 knows about the existence of only 4 other nodes. Other nodes can have different node identifiers in almost every entry in its Finger table, like the node n_{50} from Figure 1.

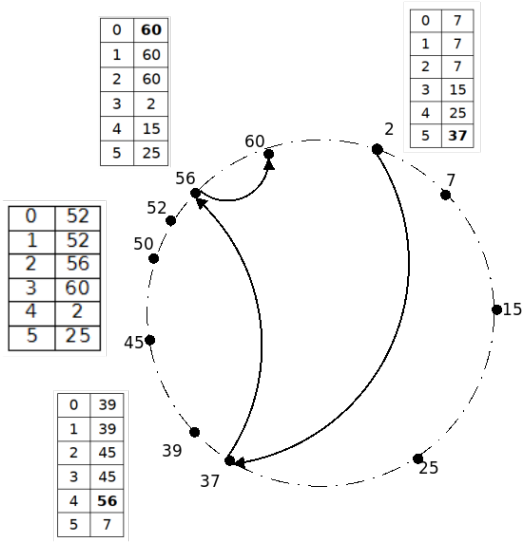


Fig. 1. Chord lookup procedure. Node n_2 is looking for the node responsible for the key with the identifier 57.

During the lookup procedure, a node forwards a query to the largest element of the Finger table smaller than the key used in the query, in respect to the used arithmetic modulo 2^m. In the example illustrated by Figure 1, if n_2 is looking for the responsible node for the key with identifier 57, it will forward this query to node n_{37}, the closest predeceasing node from its finger table to the identifier 57, the closes node from its finger table. After that, this query will be forwarded to n_{56} (again as the closest predeceasing node from its finger table to the identifier 57), until it finally ends at n_{60}, as the successor of the node n_{56}. The answer if n_{60} contains the key and respected value with identifier 57 will be returned to node that started query, in this case n_2.

When a node n joins an existing network, certain keys previously assigned to n's successor now become assigned to n. When a node n leaves the network regularly, it notifies its predecessor and successor and reassigns all of its keys to the successor.

The stabilization procedure implemented by Chord runs periodically in the background at each node and must guarantee that each node's finger table, predecessor and successor pointers are up to date.

Figure 2 illustrates the process of stabilization after joining of a new node n_5 between nodes n_2 and n_7. Figure 2a shows a pair of stable nodes n_2 and n_7 that are a part of the

Chord ring. When a new node n_5 joins the system, it sets its successor to n_7 as presented in Figure 2b. Node n_7 will then set its predecessor to n_5 as seen in Figure 2c, and node n_2 will later set its successor to n_5 as shown in Figure 2d. Finally, node n_5 will set its predecessor to n_2 as seen in Figure 2e.

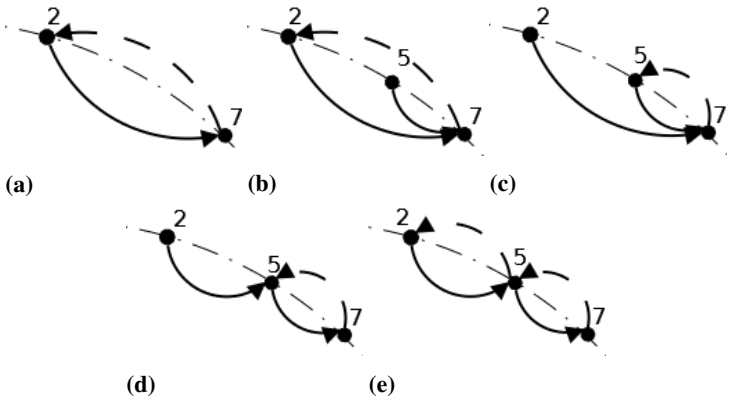


Fig. 2. Stabilization process during the joining of a new node: (a) a stable pair of nodes n_2 and n_7; (b) node n_5 joins and sets its successor to n_7; (c) node n_7 sets its predecessor to n_5; (d) node n_2 sets its successor to n_5; (e) node n_5 sets its predecessor to n_2.

Figure 3 illustrates the process of stabilization after leaving of the node n_5. Figure 3a shows two pairs of stable nodes n_2 and n_5, and n_5 and n_7 that are a part of the Chord ring. When the node n_5 leaves the system, nodes n_2 and n_7 point to nothing, as presented in Figure 3b. Then, node n_2 will set its successor to n_7 as shown in Figure 3c. Finally, node n_7 will set its predecessor to n_2 as seen in Figure 3d.

4. Logic of Time and Knowledge

As we mentioned in the previous Section, a system which runs the Chord protocol is a dynamic multi-agent system, where every node has it own partial view of the surrounding environment. To be able to reason about such systems, we need to introduce a framework for formal description of changes of the knowledge of a node during the time, and which allows nodes to share their knowledge. In this section we present the corresponding temporal-epistemic logic.

4.1. Syntax

Let $\mathbb{N}$ be the set of non-negative integers. We denote $\mathbf{Nodes} = \{n_0, \ldots n_{m-1}\}$, where $m \in \mathbb{N}$, and then let $\mathbf{N_1} = \mathbf{Nodes} \cup \{u\}$ be the set of propositional variables. The elements from the set $\mathbf{Nodes}$ will represent nodes of the Chord network, while u is the special letter used in the situation when some value is undefined.

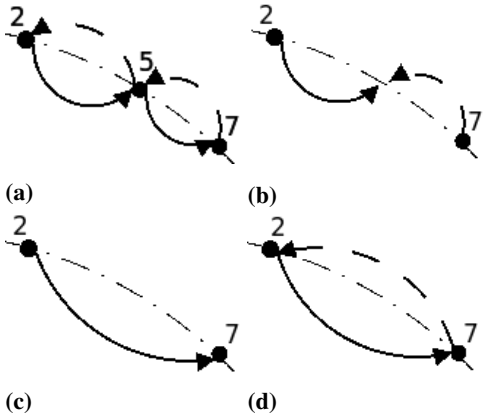
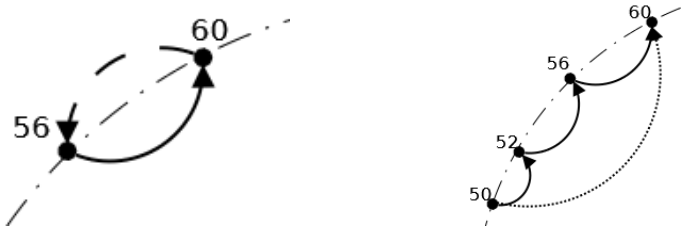


Fig. 3. Stabilization process after leaving of a Chord ring node: (a) two stable pairs of nodes n_2 and n_5, and n_5 and n_7; (b) node n_5 leaves the Chord ring; (c) node n_2 sets its successor to n_7; (d) node n_7 sets its predecessor to n_2;.

The set For of all formulas is the smallest superset of $\mathbf{N_1}$ which is closed under the following formation rules:

- $\langle \phi, \psi \rangle \mapsto \phi * \psi$ where $* \in \{\succ, \prec\}$ and $\phi, \psi \in \mathbf{N_1}$,
- $\langle \phi, \psi, \varphi \rangle \mapsto \phi \mathtt{M} \langle \psi, \varphi \rangle$ where $\phi, \psi, \varphi \in \mathbf{Nodes}$,
- $\psi \mapsto *\psi$ where $* \in \{\neg, \bigcirc, \bullet, \mathtt{K}_i\}$,
- $\langle \phi, \psi \rangle \mapsto \phi * \psi$ where $* \in \{\wedge, \mathtt{G}, \mathtt{H}\}$.

The operators $\succ$ and $\prec$ represent the relations successor and predecessor of a node, respectively. The tip of the "arrow" is pointing to the node with "greater" identifier, with respect to the ordering determined by the ring shaped Chord network. An abbreviation $n_i \succ^2 n_k$ will be used for $n_i, n_k \in N$ iff there is an $n_j \in N$ such that $n_i \succ n_j$ and $n_j \succ n_k$, and $n_k \prec^2 n_i$ for $n_i, n_k \in N$ iff there is an $n_j \in N$ such that $n_k \prec n_j$ and $n_j \prec n_i$. Similarly, we can define $n_j \succ^i n_k$, as well as $n_j \prec^i n_k$ for $n_j, n_k \in N$ and $0 < i < m$. Figure 4 illustrates the relations $\succ$, $\prec$ (Figure 4a) and $\succ^i$ (Figure 4b).



(a). $n_{56} \succ n_{60}$ and $n_{60} \prec n_{56}$ **(b).** $n_{50} \succ^3 n_{60}$

Fig. 4. Examples of $\succ$, $\prec$ and $\succ^i$

The operators $\neg$ and $\wedge$ are logical negation and conjunction. The operators $\bigcirc$, $\bullet$, $\mathtt{G}$ and $\mathtt{H}$ are standard temporal operators *next*, *previous*, *always in the future* and *always in the past*. The operator $\mathtt{K}_i$ represents the knowledge of the node i.

The remaining logical $\vee$, $\rightarrow$, $\leftrightarrow$, and temporal $\mathtt{F}$ (*eventually in the future*), $\mathtt{P}$ (*eventually in the past*) connectives, $\mathtt{G}$, $\mathtt{H}$ are defined in the usual way:

- $\phi \vee \psi =_{def} \neg(\neg\phi \wedge \neg\psi)$,
- $\phi \rightarrow \psi =_{def} \neg\phi \vee \psi$,
- $\phi \leftrightarrow \psi =_{def} (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$,
- $\mathtt{F}\psi =_{def} \neg\mathtt{G}\neg\psi$,
- $\mathtt{P}\psi =_{def} \neg\mathtt{H}\neg\psi$,
- $\bigcirc^0\psi =_{def} \psi; \bigcirc^{n+1}\psi = \bigcirc \bigcirc^n \psi, n \geqslant 0$,
- $\bullet^0\psi =_{def} \psi; \bullet^{n+1}\psi = \bullet\bullet^n\psi, n \geqslant 0$.

4.2. Semantics

In this paper we will consider time flow which is isomorphic to the set $\mathbb{N}$. We take into account both future and past.

We will defined models as Kripke's structures, where the central notion is that of run. A run is an ordered list (a sequence) of consecutive states of the system and corresponds to a possible execution which starts in an initial state.

Definition 1. *A model $\mathcal{M}$ is any tuple $\langle R, \pi, \mathcal{A}, \mathcal{K}\rangle$ such that*

- *R is the set of runs, where:*
  - *every run $r = \langle (x_0^t, \ldots, x_{m-1}^t) | t = 0, 1, 2 \ldots \rangle$, is a countably infinite sequence, where $x_i^t \in \{\top, \bot\}$, and*
  - *a state (or a possible world) of a run r is $\langle r, t\rangle = (x_0^t, \ldots, x_{m-1}^t)$,*
- *$\pi : R \times \mathbb{N} \times \mathbf{N_1} \rightarrow \{\top, \bot\}$ is the set of valuations:*
  - *$\pi(r, t, n_l) = x_l^t$, associates truth values to propositional letters of the possible world $\langle r, t\rangle$,*
- *$\mathcal{A}$ associates sets of active nodes to possible worlds, and*
- *$\mathcal{K} = \{\mathcal{K}_a : a \in \mathbf{A}\}$ is the set of transitive and symmetric accessibility relations for nodes, such that:*
  - *if $a \notin \mathcal{A}(\langle r, t\rangle)$, then $\langle r, t\rangle \mathcal{K}_a \langle r', t'\rangle$ is false for all $r' \in R$ and all $t' \in \mathbb{N}$.*

Actually, a state, or a possible world $\langle r, t\rangle$, in a run r represents the state of the system in the corresponding time instant t, and we will alternatively use these notions in the rest of the paper.

Figure 5 illustrates a Kripke model which contains the runs r^1, r^2, r^3, r^4, where r^1 is the sequence of $\langle r^1, 0\rangle$, $\langle r^1, 1\rangle$, $\langle r^1, 2\rangle$, etc. and similarly for other runs. In this model, for example $\langle r^2, 1\rangle \mathcal{K}_1 \langle r^2, 2\rangle$, etc.

An $n_i \in \mathbf{Nodes}$ is $true$ in the time instant t in the run r ($x_i^t = \top$) if the Chord network node i is active in the corresponding realization of the network. For $n_i, n_j, n_k \in \mathbf{Nodes}$ we define the relation $\mathtt{M}$ which represents the fact that n_i is the member of the ring interval $(n_j, n_k]$ as: $n_i \mathtt{M} \langle n_j, n_k\rangle$ is $true$ iff

- $j = k$, or
- $j < k$ and $j < i \leqslant k$, or
- $k < j$ and $\neg(k < i \leqslant j)$.

Note that the relation $\mathtt{M}$ is defined for all members of the set $\mathbf{Nodes}$, regardless the fact if the members are active or not.

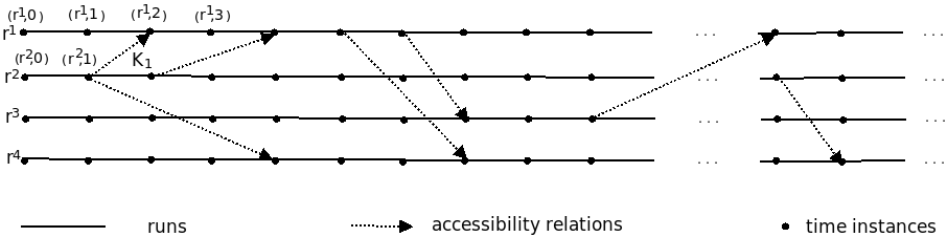


Fig. 5. Kripke model

4.3. Satisfiability relation

A formula is satisfiable if there is a possible world in a model which makes that formula true.

Definition 2. *Let $\mathcal{M} = \langle R, \pi, \mathcal{A}, \mathcal{K}\rangle$ be any model. The satisfiability relation $\models$ (formula α is satisfied in a time instant t of a run r) is defined recursively as follows:*

1. $\langle r,t\rangle \models n$ *iff* $\pi(r,t,n) = true$, $n \in \mathbf{N_1}$
2. $\langle r,t\rangle \models \alpha \wedge \beta$ *iff* $\langle r,t\rangle \models \alpha$ *and* $\langle r,t\rangle \models \beta$
3. $\langle r,t\rangle \models \neg\alpha$ *iff not* $\langle r,t\rangle \models \alpha$ *(* $\langle r,t\rangle \not\models \alpha$*)*
4. $\langle r,t\rangle \models \bigcirc\alpha$ *iff* $\langle r,t+1\rangle \models \alpha$
5. $\langle r,t+1\rangle \models \bullet\alpha$ *iff* $\langle r,t\rangle \models \alpha$
6. $\langle r,0\rangle \models \bullet\alpha$
7. $\langle r,t\rangle \models \mathtt{G}\alpha$ *iff for all* $i \geqslant 0$ *holds* $\langle r,t+i\rangle \models \alpha$
8. $\langle r,t\rangle \models \mathtt{H}\alpha$ *iff for all* $0 \leqslant i \leqslant t$ *holds* $\langle r,t-i\rangle \models \alpha$
9. $\langle r,t\rangle \models \mathtt{K}_i\alpha$ *iff* $\langle r',t'\rangle \models \alpha$ *for all* $\langle r',t'\rangle \in \mathcal{K}_i(\langle r,t\rangle)$
10. $\langle r,t\rangle \models n_i \succ n_j$ *iff*
    (a) $i = j$ *and* $\langle r,t\rangle \models n_i \wedge \mathtt{K}_i(\bigwedge_{n_j \in \mathbf{Nodes}\setminus\{n_i\}} \neg n_j)$
    (b) $i < j \leqslant m$ *and* $\langle r,t\rangle \models n_i \wedge n_j \wedge \mathtt{K}_i(\bigwedge_{k=i+1}^{j-1} \neg n_k) \wedge \mathtt{K}_i n_j$
    (c) $j < i < m$ *and* $\langle r,t\rangle \models n_i \wedge n_j \wedge \mathtt{K}_i(\bigwedge_{k=i+1}^{m} \neg n_k) \wedge \mathtt{K}_i(\bigwedge_{k=1}^{j-1} \neg n_k) \wedge \mathtt{K}_i n_j$
    (d) $j < i$ *and* $i = m$ *and* $\langle r,t\rangle \models n_i \wedge n_j \wedge \mathtt{K}_i(\bigwedge_{k=1}^{j-1} \neg n_k) \wedge \mathtt{K}_i n_j$
11. $\langle r,t\rangle \models n_j \prec n_i$ *iff*
    (a) $i = j$, $t \neq 0$ *and* $\langle r,t\rangle \models n_i \wedge \mathtt{K}_i(\bigwedge_{n_k \in \mathbf{Nodes}\setminus\{n_i\}} \neg n_k)$
    (b) $i < j \leqslant m$ *and* $\langle r,t\rangle \models n_i \wedge n_j \wedge \mathtt{K}_i(\bigwedge_{k=i+1}^{j-1} \neg n_k) \wedge \mathtt{K}_i n_j$
    (c) $j < i < m$ *and* $\langle r,t\rangle \models n_i \wedge n_j \wedge \mathtt{K}_i(\bigwedge_{k=i+1}^{m} \neg n_k) \wedge \mathtt{K}_i(\bigwedge_{k=1}^{j-1} \neg n_k) \wedge \mathtt{K}_i n_j$
    (d) $j < i$ *and* $i = m$ *and* $\langle r,t\rangle \models n_i \wedge n_j \wedge \mathtt{K}_i(\bigwedge_{k=1}^{j-1} \neg n_k) \wedge \mathtt{K}_i n_j$
    (e) $n_i = u$ *and* $\langle r,t\rangle \models \neg n_j \vee (n_j \wedge (\bullet(\neg \mathtt{K}_k(n_k \succ n_j))))$

Now, we can explain the intuitive meaning of the operators. For example, the formula $\bigcirc\alpha$ is true in a state $\langle r,t\rangle$ of a run r iff α is true in the next state $\langle r,t+1\rangle$ of the same run. Next, $\mathtt{K}_i\alpha$ is true in a state $\langle r,t\rangle$ iff α is true in all states $\langle r',t'\rangle$ accessible from $\langle r,t\rangle$ by the relation $\mathcal{K}_i(\langle r,t\rangle)$. On the other hand, the formula $n_i \succ n_j$ is true if there is only one active node in the Chord network (case (a)), or there is not other active nodes between n_i and n_j and node n_i is aware of activity of n_j (cases (b)-(d)). The cases (b)-(d) take in

account the ring structure of the Chord network, i.e. $[n_i, n_j]$ is one ring interval. Similarly for $n_j \prec n_i$.

By Definition 2 it is trivial to prove that the following lemma holds:

Lemma 1. **TP:** $\langle r,t\rangle \models (\bigcirc\alpha \wedge \bigcirc\beta) \leftrightarrow \bigcirc(\alpha \wedge \beta)$.

We use this property in the proofs provided in Appendix A.

5. Proof of Correctness

In this Section we analyze correctness of the Chord protocol. More precisely, we prove that any execution of a regular run maintains the ring topology of the corresponding network. It means that the network will be brought, after a finite number of steps, from a state in which links between nodes do not form a ring to a stable state. First, we need to introduce the following definitions:

Definition 3 (Stable pair). *The pair of active nodes $\langle n_k, n_l\rangle$ is stable in a time instant t of a run r (i.e., in the state $\langle r,t\rangle$), denoted with $n_k \Subset n_l$, iff*

$$\langle r,t\rangle \models (n_l \succ^{m_1} n_k) \wedge (\bigwedge_{j=1}^{m_1} \mathtt{K}_{i_j}(n_{i_j} \succ n_{i_{j+1}})) \wedge (n_l \prec^{m_1} n_k) \wedge (\bigwedge_{j=1}^{m_1} \mathtt{K}_{i_{j+1}}(n_{i_{j+1}} \prec n_{i_j})),$$

where $n_k, n_l, n_{i_j} \in \mathcal{A}(\langle r,t\rangle)$ for $1 \leqslant j \leqslant m_1$.

Intuitively, a pair $\langle n_k, n_l\rangle$ is stable if:

- the nodes n_k and n_l are active, and
- every active node n_j such that $n_j\mathtt{M}\langle n_k, n_l\rangle$ knows its successor and predecessor.

It means that there is no node n_i between n_k and n_l which tries to join or leave the network in $\langle r,t\rangle$.

Definition 4 (Stable network). *A Chord network is stable (we denote it with $\circledcirc$) at $\langle r,t\rangle$ iff $n_k \Subset n_k$ for all $n_k \in \mathcal{A}(\langle r,t\rangle)$.*

Intuitively, the whole network is stable if all successor and predecessor pointers are sorted.

Definition 5. *A node n_i is a member of a stable pair $\langle n_k, n_l\rangle$ iff $n_i\mathtt{M}\langle n_k, n_l\rangle$ and $n_k \Subset n_l$.*

Definition 6 (Regular runs). *A run is regular if in every state of the run a node can leave the network only if it is a member of a stable pair of nodes.*

Definition 6 is essential in producing a deterministic proof of the correctness of the Chord protocol, since if it is allowed that nodes can leave the network when they are members of unstable pairs, there are counter examples in which those nodes can be left isolated or the network can be divided into more then one separate subnetworks, as it is shown in [16].

We assume the following form of the fairness condition: there is a constant $f \in \mathbb{N}$ which denotes a uniform limit from above for duration of all primitive actions of the Chord protocol (listed in Section 3).

The basic properties of $\succ$ and $\prec$ relations can be described with:

AS1: $\langle r,t\rangle \models n_i \succ n_j \rightarrow \bigwedge_{n_k \in \mathbf{N_1}\setminus\{n_j\}} \neg(n_i \succ n_k), n_i, n_j \in \mathbf{Nodes}$
AS2: $\langle r,t\rangle \models n_i \prec n_j \rightarrow \bigwedge_{n_k \in \mathbf{N_1}\setminus\{n_j\}} \neg(n_i \prec n_k), n_i, n_j \in \mathbf{Nodes}$
AS3: $\langle r,t\rangle \models n_i \prec n_j \rightarrow \bigwedge_{n_k \in \mathbf{N_1}\setminus\{n_i\}} \neg(n_k \prec n_j), n_i, n_j \in \mathbf{Nodes}$
AS4: $\langle r,t\rangle \models n_i \prec n_j \rightarrow n_j \succ n_i, n_i, n_j \in \mathbf{Nodes}$
AS5: $\langle r,t\rangle \models n_i \succ n_j \rightarrow \mathtt{K}_i(n_i \succ n_j), n_i, n_j \in \mathbf{Nodes}$
AS6: $\langle r,t\rangle \models ((n_i \succ n_j) \wedge n_k \mathtt{M}\langle n_i, n_j\rangle \wedge \bigcirc(\neg \mathtt{K}_i n_k)) \rightarrow \bigcirc(n_i \succ n_j)$,
$n_i, n_j, n_k \in \mathbf{Nodes}$
AS7: $\langle r,t\rangle \models ((n_i \prec n_j) \wedge n_k \mathtt{M}\langle n_i, n_j\rangle \wedge \bigcirc(\neg \mathtt{K}_i n_k)) \rightarrow \bigcirc(n_i \prec n_j)$,
$n_i, n_j, n_k \in \mathbf{Nodes}$

[AS1] says that a node can have only one successor. [AS2] says that a node can be predecessor of only one node. [AS3] says that a node can have only one predecessor. [AS4] says that if a node is predecessor of some other node, that other node has to be its successor. [AS5] says that if a node n_i has the successor n_j, then n_i knows that n_j is its successor. [AS6] says that the current successor (n_j) of a node (n_i) will be successor of the node in the next time instant ($\bigcirc(n_i \succ n_j)$), assuming that n_i does not know ($\bigcirc(\neg \mathtt{K}_i n_k)$)) that a new node n_k which belongs to the ring interval $[n_i, n_j)$ (i.e., $n_k \mathtt{M}\langle n_i, n_j\rangle$) joins the network. Similarly, [AS7] says when the current predecessor will be the predecessor in the next time instant, if there are no new nodes.

The primitive actions of the Chord network can be describe in the following way:

ρ_S: $\langle r,t\rangle \models \mathtt{H}(\bigwedge_{n_j \in \mathbf{Nodes}} \neg n_j) \wedge n_i \wedge (\bigwedge_{n_j \in \mathbf{Nodes}\setminus\{n_i\}} \neg n_j) \wedge \mathtt{K}_i(n_i \succ n_i) \wedge \mathtt{K}_i(n_i \prec u)$ for one $n_i \in \mathbf{Nodes}$,
$\rho_{J,i}$: $\langle r,t\rangle \models \bullet(\neg n_i) \wedge n_i \wedge \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_i(n_i \prec u), n_j \in \mathcal{A}(\langle r,t\rangle)$,
$n_i \in \mathbf{Nodes}, i \neq j$,
$\rho_{L,i}$: $\langle r,t\rangle \models \bullet(n_i \wedge n_j \Subset n_k) \wedge n_i \mathtt{M}\langle n_j, n_k\rangle \wedge \neg n_i, n_i \in \mathbf{Nodes}\, n_k, n_j \in \mathcal{A}(\langle r,t\rangle)$
$\rho_{S1,i,j}$: $\langle r,t\rangle \models (\mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_j(n_j \prec u)) \vee (\mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_k) \wedge n_i \mathtt{M}\langle n_k, n_j\rangle) \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_j(n_j \prec n_i), n_i, n_k, n_j \in \mathcal{A}(\langle r,t\rangle)$,
$\rho_{S2,i,j}$: $\langle r,t\rangle \models \mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_k) \wedge n_k \mathtt{M}\langle n_i, n_j\rangle \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_i(n_i \succ n_k)$,
$n_i, n_k, n_j \in \mathcal{A}(\langle r,t\rangle)$,
$\rho_{S3,i}$: $\langle r,t\rangle \models \mathtt{K}_i(n_i \succ n_j) \wedge \neg n_j \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_i(n_i \succ n_k) \wedge \bigvee_{l \in \mathcal{A}(\langle r,t\rangle)} \neg n_l \mathtt{M}\langle n_i, n_k\rangle$,
$n_i, n_k \in \mathcal{A}(\langle r,t\rangle), n_j \in \mathbf{Nodes}$
$\rho_{S4,i}$: $\langle r,t\rangle \models \mathtt{K}_i(n_j \prec n_i) \wedge \neg n_j \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_i(u \prec n_i), n_i \in \mathcal{A}(\langle r,t\rangle)$,
$n_j \in \mathbf{Nodes}$

[ρ_S] describes the start of the new Chord network, i.e., there was no active node in the past ($\mathtt{H}(\bigwedge_{n_j \in \mathbf{Nodes}} \neg n_j)$), in the current time instant only n_i becomes active ($n_i \wedge (\bigwedge_{n_j \in \mathbf{Nodes}\setminus\{n_i\}} \neg n_j)$), and n_i knows that it is its own successor ($\mathtt{K}_i(n_i \succ n_i)$), while its predecessor is still undefined ($\mathtt{K}_i(n_i \prec u)$). Similarly, for the other formulas: [$\rho_{J,i}$] represents the situation when a new node n_i joins the existing Chord network, while [$\rho_{L,i}$] represents the situation when a new node n_i leaves the existing Chord network in a regular run. [$\rho_{S1,i,j}$] - [$\rho_{S4,i}$] characterize stabilization processes (that make predecessor and successor pointers up to date).

To be able to describe periodicity of the stabilization process, we introduce the following formulas:

ACF1: $\langle r,t\rangle \models n_i \wedge \rho_S \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S1,i,j}, n_i \in \mathcal{A}(\langle r,t\rangle)$,

ACF2: $\langle r,t\rangle \models n_i \wedge \rho_S \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S2,i,j}, n_i \in \mathcal{A}(\langle r,t\rangle)$,
ACF3: $\langle r,t\rangle \models n_i \wedge \rho_{J,i} \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S1,i,j}, n_i \in \mathcal{A}(\langle r,t\rangle)$,
ACF4: $\langle r,t\rangle \models n_i \wedge \rho_{J,i} \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S2,i,j}, n_i \in \mathcal{A}(\langle r,t\rangle)$,
ACF5: $\langle r,t\rangle \models n_i \wedge \rho_{S1,i,k} \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S1,i,j}, n_i \in \mathcal{A}(\langle r,t\rangle)$, $k \in \{0, m-1\}$,
ACF6: $\langle r,t\rangle \models n_i \wedge \rho_{S2,i,k} \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S2,i,j}, n_i \in \mathcal{A}(\langle r,t\rangle)$, $k \in \{0, m-1\}$,
ACF7: $\langle r,t\rangle \models n_i \wedge \rho_S \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S3,i}, n_i \in \mathcal{A}(\langle r,t\rangle)$,
ACF8: $\langle r,t\rangle \models n_i \wedge \rho_S \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S4,i}, n_i \in \mathcal{A}(\langle r,t\rangle)$,
ACF9: $\langle r,t\rangle \models n_i \wedge \rho_{J,i} \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S3,i}, n_i \in \mathcal{A}(\langle r,t\rangle)$,
ACF10: $\langle r,t\rangle \models n_i \wedge \rho_{J,i} \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S4,i}, n_i \in \mathcal{A}(\langle r,t\rangle)$,
ACF11: $\langle r,t\rangle \models n_i \wedge \rho_{S3,i} \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S3,i}, n_i \in \mathcal{A}(\langle r,t\rangle)$,
ACF12: $\langle r,t\rangle \models n_i \wedge \rho_{S4,i} \rightarrow \bigvee_{l=0}^{f} \bigcirc^l \bigvee_{j=0}^{m-1} \rho_{S4,i}, n_i \in \mathcal{A}(\langle r,t\rangle)$.

The correctness of the Chord protocol can be proved by compounding simpler cases. The statements 2-6 guarantee that the successor and predecessor pointers for each node will be eventually sorted after one or more nodes join existing or start a new network. Theorem 6 expresses the correctness of the model of executions without failures of nodes and corresponds to Theorem IV.3 from [5]. The statements 7-9 consider possible leaving of a node. Lemma 10 says that a stable pair of nodes in a Chord network eventually becomes stable after adding/removing of a node between them with the respect to the regular runs, while Theorem 4 shows the same, but for a stable network.

Lemma 2. *Let a node start a new Chord network. Then, there is a finite period of time after which the network will be stable, if no other nodes are trying to join in the meanwhile.*

Lemma 3. *Let a new node join a stable Chord network which consists of only one node. Then, there is a finite period of time after which the network will be stable again, if no other nodes are trying to join in the meanwhile.*

Proofs of Lemmas 2 and 3 are similar to the proof of Lemma 4 (that can be found in Appendix A).

Lemma 4. *Let a node join a Chord network, between two nodes which constitute a stable pair, such that the second node is the successor of the first node. Then, there is a finite period of time after which the starting pair will be stable again, if no other nodes are trying to join the network between the nodes that constitute the particular stable pair in the meanwhile.*

Lemma 5. *Let a node join a Chord network, between two nodes which constitute a stable pair. Then, there is a finite period of time after which the starting pair will be stable again, if no other nodes are trying to join in the meanwhile.*

Proof. Since one new node is joining the network between a stable pair, we can choose two nodes which are each others successor and predecessor and the new node is joining between them, so we can apply Lemma 4.

Lemma 6. *Let a Chord network contain a stable pair. If a sequence of nodes join between the nodes that constitute this stable pair, then there is a finite period of time after which the starting pair will be stable again.*

Proof. If we assume that all nodes that want to join the network have different successors, by Lemma 5 the statement holds.

If this is not the case, we can assume that $n_i \Cap n_k$ and that the set of nodes $n_{j_1}, n_{j_2}, \ldots$, such that $i \leqslant \ldots \leqslant j_2 \leqslant j_1 \leqslant k$, are joining this stable pair. Then, we can apply Lemma 5 on the tuples $\langle n_i, n_{j_1}, n_k\rangle$, $\langle n_i, n_{j_2}, n_{j_1}\rangle$, ... This process will produce the stable pair $n_i \Cap n_k$, again.

Lemma 7. *Let a Chord network contain a stable pair and let a node between them leave the network. Then, there is a finite period of time after which the starting pair will be stable again, if no other nodes are trying to join in the meanwhile.*

Proofs of Lemmas 8 and 9 are similar to the proof of Lemma 7 (that can be found in Appendix A).

Lemma 8. *Let a Chord network contain a stable pair $n_i \Cap n_k$. Let a node which is between those nodes leave the network followed by several nodes which want to join between $\langle n_i, n_k\rangle$.Then, there is a finite period of time after which $\langle n_i, n_k\rangle$ will be stable again.*

Lemma 9. *Let a Chord network contain a stable pair. Let a node, which is in between those nodes, leave the network. Then, there is a finite period of time after which the starting pair will be stable again.*

Lemma 10. *Let a finite initial segment of a run of a Chord network produce the state $\langle r, t\rangle$, and $\langle n, n'\rangle \in \mathcal{A}(\langle r, t\rangle)$ be nodes that do not leave the network. Then, there is a finite period of time after which $\langle n, n'\rangle$ will be stable, i.e., $n \Cap n'$.*

Proof. First note that, if the pair is stable at $\langle r, t\rangle$, then the statement trivially holds. Otherwise, since we consider only regular runs, only joining of the new nodes between $\langle n, n'\rangle$ is possible. So, this is similar to Lemma 9.

Theorem 1. *Let a finite initial segment of a run produce the state $\langle r, t\rangle$ of a Chord network. Then, there is a finite period of time after which the network will be stable again:*

$$\langle r, t\rangle \models \neg\circledcirc \rightarrow \mathtt{F}\circledcirc$$

Proof. We assume that at least one node does not leave the network. Note that an unstable state can be reached under following conditions:

- if a node starts the Chord network, which is considered in Lemma 2,
- if a node joins the Chord network, which is considered in Lemma 3, 4, 5 and 6,
- if a node leaves the Chord network, which is considered in Lemma 7, and
- if some nodes are leaving and some nodes are joining the Chord network, which is considered in Lemma 8, 9 and 10.

So, as an unstable state can be reached only in some of the cases that are already considered, this theorem is the corollary of the lemmas 2 – 10.

6. Conclusion

Discovery and control services are the core part of every IoT system. To improve the performances of them, we found several examples where the regular approach was replaced by DHT, i.e. the Chord protocol. All these examples do not consider correctness of the Chord protocol and take it for granted, or consider it by very strong assumptions. To be used in a proper manner, it is necessary to describe in a deterministic scenario for all the situation when the Chord protocol manifest incorrect behavior. For that purpose, we define the notion of regular runs and prove the correctness of the maintenance of the ring topology of the Chord protocol with the respect of them, using the framework of time and knowledge.

One of the possible directions for further work is to apply the similar technique to describe other DHT protocols and other cloud processes. Another challenge could be to verify the given proof in one of the formal proof assistants (e.g., Coq, Isabelle/HOL). It might also produce a certified program implementation from the proof of correctness.

Acknowledgments. The work presented here was supported by Serbian Ministry of Education, Science and Technology Development (the projects ON174026 and III44006), through Matematički institut SANU, and Croatian Ministry of Science and Education. This work was supported in part by the Slovenian Research Agency within the research program Algorithms and Optimization Methods in Telecommunications.

References

1. L. Atzori, A. Iera, G. Morabito. *The Internet of things: A survey*. In *Computer Networks*, 54.15, 2787–2805, 2010.
2. S. Cirani, L. Davoli, G. Ferrari, R. Léone, P. Medagliani, M. Picone, L. Veltri. *A Scalable and Self-Configuring Architecture for Service Discovery in the Internet of Things*. In *IEEE Internet of Things Journal*, Vol. 1, No. 5, 508–521, 2014.
3. R. Rodrigues, P. Druschel. *Peer-to-Peer Systems* In *Communications of the ACM*, Vol. 53 Issue 10, pages 72–82, October 2010.
4. I. Taylor. *From P2P to Web Services and Grids*. Springer-Verlag, 2005.
5. I. Stoica, R. Morris, D. Karger, M. Kaashoek, H. Balakrishnan. *Chord: A Scalable Peer-to-Peer Lookup service for Internet Applications*. In *ACM SIGCOMM*, pages 149–160, 2001.
6. I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, H. Balakrishnan. *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications*. MIT Technical report, TR-819, 2001.
7. I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, H. Balakrishnan. *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications*. In *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, 17 – 32, 2003.
8. J. J. Bolonio, M. Urueña, G. Camarillo. *A Distributed Control Plane for the Internet of Things Based on a Distributed Hash Table*. In *Mobile Networks and Management*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 125, 108–121, 2013.
9. S. Evdokimov, B. Fabian, S. Kunz, N. Schoenemann. *Comparison of Discovery Service Architectures for the Internet of Things*. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*, 2010.
10. F. Paganelli, D. Parlanti. *A DHT-Based Discovery Service for the Internet of Things*. In *Journal of Computer Networks and Communications*, doi:10.1155/2012/107041, 2012.

11. D. Xu, Z. Wu, Z. Wu, Q. Zhang, L. Qin, J. Zhou. *Internet of Things: Hotspot-based Discovery Service Architecture with Security Mechanism.* In *International Journal of Network Security*, Vol. 17, No. 2, 208–216, 2015.
12. R. Bakhshi, D. Gurov. *Verification of Peer-to-peer Algorithms: A Case Study*. Technical report, ICT, 2006.
13. R. Bakhshi, D. Gurov. *Verification of Peer-to-peer Algorithms: A Case Study*. In *Electronic Notes in Theoretical Computer Science* (ENTCS), Volume 181, 35–47, 2007.
14. S. Krishnamurthy, S. El-Ansary, E. Aurell, S. Haridi. *A Statistical Theory of Chord Under Churn*. In *4th International Workshop on Peer-To-Peer Systems*, pages 93–103, 2005.
15. D. Liben-Nowell, H. Balakrishnan, D. R. Karger. *Analysis of the Evolution of Peer-to-Peer Systems*. In *Proc.* 21^{st} *ACM Symp. Principles of Distributed Computing (PODC)*, pages 233–242, 2002.
16. P. Zave. *Using Lightweight Modeling to Understand Chord.* In ACM SIGCOMM Computer Communication Review, Vol. 42, Issue 2, pages 50–57, April 2012.
17. R. Fagin, J. Y. Halpern, Y. Moses, M. Y. Vardi. *Reasoning About Knowledge*. The MIT Press, Cambridge, Massachusetts, 1995.
18. D. R. Karger, E. Lehman, F. T. Leighton, R. Panigrahy, M. S. Levine, D. Lewin. *Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web*. In *Proceedings of STOC'97*, pages 654–663, 1997.
19. B. Marinkovic, Z. Ognjanovic, D. Doder, A. Perovic. *A Propositional Linear Time logic with Time Flow Isomorphic to* ω^2. In *Journal of Applied Logic*, 12(2), 208 – 229, 2014.
20. Z. Ognjanovic. *Discrete Linear-time Probabilistic Logics: Completeness, Decidability and Complexity*. In *Journal of Logic Computation*, Vol. 16, No. 2, 257–285, 2006.
21. Z. Ognjanovic, D. Doder, Z. Markovic. *A Branching Time Logic with Two Types of Probability Operators*. In *Fifth International Conference on Scalable Uncertainty Management SUM-2011*, Springer LNCS 6929, 219–232, 2011.

A. Proofs

Recall that f denotes the maximum of durations of all primitive actions of the Chord protocol.

Lemma 4. *Let a node join a Chord network, between two nodes which constitute a stable pair, such that the second node is the successor of the first node. Then, there is a finite period of time after which the starting pair will be stable again, if no other nodes are trying to join the network between the nodes that constitute the particular stable pair in the meanwhile.*

Proof. We will show that 5 rounds of the stabilization process (i.e., $5f$ steps of an execution of the Chord protocol) will be enough to guarantee that the considered pair of nodes becomes stable.

Let us assume that $n_i, n_j \in \mathcal{A}(\langle r, t\rangle)$ and $n_i \Cap n_j$, i.e. $(n_i \succ n_j) \wedge (n_j \prec n_i)$ and that n_k tries to join that stable pair. Let us denote

$$\alpha = \bullet(n_i \Cap n_j) \wedge \rho_{J,k} \bigwedge_{n_l \in I} \bigwedge_{t=0}^{5f} \bigcirc^t \neg n_l, I = \{n_l | n_l \mathtt{M} \langle n_i, n_j\rangle, n_k \neq n_l, n_j \neq n_l\}.$$

The formula α says that the pair $\langle n_i, n_j\rangle$ is stable in the previous time instant ($\bullet(n_i \Cap n_j)$), n_k is joining ($\rho_{J,k}$) and during the next $5f$ time instants no other node

($\bigcirc^t \neg n_l$) is trying to join the network between the nodes that constitute initial stable pair ($n_l \mathtt{M} \langle n_i, n_j \rangle$).

By MP we denote the standard inference rule *modus ponens*:

$$\text{from } \alpha \text{ and } \alpha \to \beta \text{ conclude } \beta.$$

We have,

$\langle r,t \rangle \models \alpha$ (0)

$\langle r,t \rangle \models \mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_i) \wedge n_k \mathtt{M} \langle n_i, n_j \rangle \wedge n_k$ (by AS6) (1)

$\langle r,t \rangle \models \mathtt{K}_i(n_i \succ n_j)$ (by definition of $\wedge$ and 1) (2a)

$\langle r,t \rangle \models \mathtt{K}_j(n_j \prec n_i)$ (by definition of $\wedge$ and 1) (2b)

$\langle r,t \rangle \models n_k \mathtt{M} \langle n_i, n_j \rangle$ (by definition of $\wedge$ and 1) (2c)

$\langle r,t \rangle \models n_k$ (by definition of $\wedge$ and 1) (2d)

$\langle r,t \rangle \models \rho_{J,k}$ (by definition of α) (2e)

$\langle r,t \rangle \models n_k \wedge \rho_{J,k}$ (by 2d, 2e) (2f)

$\langle r,t \rangle \models \rho_{J,k} \to \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_k(n_k \succ n_j)$ (by definition of $\rho_{J,k}$) (3)

$\langle r,t \rangle \models \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_k(n_k \succ n_j)$ (by MP, 2e, 3) (4)

$\langle r,t \rangle \models \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_k(n_k \succ n_j) \to \bigcirc^f \mathtt{K}_k(n_k \succ n_j),$ (by definition of AS6,4) (5)

$\langle r,t \rangle \models \bigcirc^f \mathtt{K}_k(n_k \succ n_j)$ (by MP, 4, 5) (6)

$\langle r,t \rangle \models n_k \wedge \rho_{J,k} \to \bigvee_{l=0}^{f} \bigcirc^l \rho_{S1,k,j}$ [ACF3] (7)

$\langle r,t \rangle \models \bigvee_{l=0}^{f} \bigcirc^l \rho_{S1,k,j}$ (by MP, 2f,7) (8a)

$\langle r,t \rangle \models \bigcirc^f \rho_{S1,k,j}$ (by AS6) (8b)

$\langle r,t \rangle \models \bigcirc^f ((\mathtt{K}_k(n_k \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_i) \wedge n_k \mathtt{M} \langle n_i, n_j \rangle) \to \bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_j(n_j \prec n_k))$ (by 8b) (9a)

$\langle r,t \rangle \models \bigcirc^f ((\mathtt{K}_k(n_k \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_i) \wedge n_k \mathtt{M} \langle n_i, n_j \rangle)) \to \bigcirc^f (\bigvee_{l=0}^{f} \bigcirc^l \mathtt{K}_j(n_j \prec n_k))$ (by AT2, 9a) (9b)

$\langle r,t \rangle \models \bigcirc^f \mathtt{K}_k(n_k \succ n_j)$ (by AS6, 6) (10a)

$\langle r,t \rangle \models \bigcirc^f \mathtt{K}_k(n_j \prec n_i)$ (by AS6, 2b) (10b)

$\langle r,t \rangle \models \bigcirc^f (n_k \mathtt{M} \langle n_i, n_j \rangle)$ (by AS6, 2c) (10c)

$\langle r,t\rangle \models \bigcirc^{f}(\mathtt{K}_k(n_k \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_i) \wedge n_k\mathtt{M}\langle n_i, n_j\rangle)$ (by TP, 10a, 10b, 10c) (11)

$\langle r,t\rangle \models \bigcirc^{f}(\bigvee_{l=0}^{f} \bigcirc^{l}\mathtt{K}_j(n_j \prec n_k))$ (by MP, 9,11) (12)

$\langle r,t\rangle \models \bigcirc^{2f}\mathtt{K}_j(n_j \prec n_k)$ (by definition of $\bigcirc$, AS6, 12) (13)

$\langle r,t\rangle \models \bigvee_{l=f}^{2f} \bigcirc^{l} \bigvee_{j=0}^{m-1} \rho_{S2,i,j}$ (by $n_i \in \mathcal{A}(\langle r,t\rangle)$ and ACF2 or ACF4) (14)

$\langle r,t\rangle \models \bigvee_{l=f}^{2f} \bigcirc^{l}\rho_{S2,i,k}$ (by definition of $\vee$,14) (15a)

$\langle r,t\rangle \models \bigcirc^{2f}\rho_{S2,i,k}$ (by definition AS6,15a) (15b)

$\langle r,t\rangle \models \bigcirc^{2f}(\mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_k) \wedge n_k\mathtt{M}\langle n_i, n_j\rangle) \rightarrow \bigvee_{l=0}^{f} \bigcirc^{l}\mathtt{K}_i(n_i \succ n_k))$ (by 15b) (16a)

$\langle r,t\rangle \models \bigcirc^{2f}(\mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_k) \wedge n_k\mathtt{M}\langle n_i, n_j\rangle)) \rightarrow \bigcirc^{2f}(\bigvee_{l=0}^{f} \bigcirc^{l}\mathtt{K}_i(n_i \succ n_k))$ (by AT2, 16a) (16b)

$\langle r,t\rangle \models \bigcirc^{2f}\mathtt{K}_i(n_i \succ n_j)$ (by AS6, 2a) (17a)

$\langle r,t\rangle \models \bigcirc^{2f}\mathtt{K}_j(n_j \prec n_k)$ (by AS6, 13) (17b)

$\langle r,t\rangle \models \bigcirc^{2f}(n_k\mathtt{M}\langle n_i, n_j\rangle)$ (by AS6, 2c) (17c)

$\langle r,t\rangle \models \bigcirc^{2f}(\mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_k) \wedge n_k\mathtt{M}\langle n_i, n_j\rangle)$ (by TP, 17a, 17b, 17c) (18)

$\langle r,t\rangle \models \bigcirc^{2f}(\bigvee_{l=0}^{f} \bigcirc^{l}\mathtt{K}_i(n_i \succ n_k))$ (by MP, 16b,18) (19)

$\langle r,t\rangle \models \bigcirc^{3f}\mathtt{K}_i(n_i \succ n_k),$ (by definition of $\bigcirc$, AS6,19) (20)

$\langle r,t\rangle \models \bigvee_{l=3f}^{4f} \bigcirc^{l} \bigvee_{j=0}^{m-1} \rho_{S1,i,j}$ (by $n_i \in \mathcal{A}(\langle r,t\rangle)$ and ACF1 or ACF3) (21)

$\langle r,t\rangle \models \bigvee_{l=3f}^{4f} \bigcirc^{l}\rho_{S1,i,k}$ (by definition of $\vee$,21) (22a)

$\langle r,t\rangle \models \bigcirc^{4f}\rho_{S1,i,k}$ (by definition AS6, 22a) (22b)

$\langle r,t\rangle \models \bigcirc^{4f}(\mathtt{K}_i(n_i \succ n_k) \wedge \mathtt{K}_k(n_k \prec u) \wedge n_k\mathtt{M}\langle n_i, n_j\rangle) \rightarrow \bigvee_{l=0}^{f} \bigcirc^{l}\mathtt{K}_k(n_k \prec n_i))$ (by 22b) (23a)

$\langle r,t\rangle \models \bigcirc^{4f}(\mathtt{K}_i(n_i \succ n_k) \wedge \mathtt{K}_k(n_k \prec u) \wedge n_k\mathtt{M}\langle n_i, n_j\rangle)) \rightarrow \bigcirc^{4f}(\bigvee_{l=0}^{f} \bigcirc^{l}\mathtt{K}_k(n_k \prec n_i))$ (by definition of AT2, 23a) (23b)

$\langle r,t\rangle \models \bigcirc^{4f}\mathtt{K}_i(n_i \succ n_k)$ (by AS6, 29) (24a)

$\langle r,t\rangle \models \bigcirc^{4f}\mathtt{K}_k(n_k \prec u)$ (by AS6, 2e) (24b)

$\langle r,t\rangle \models \bigcirc^{4f}(n_k\mathtt{M}\langle n_i, n_j\rangle)$ (by AS6, 2c) (24c)

$\langle r,t\rangle \models \bigcirc^{4f}(\mathtt{K}_i(n_i \succ n_k) \wedge \mathtt{K}_k(n_k \prec u) \wedge n_k\mathtt{M}\langle n_i, n_j\rangle)$ (by TP, 24a, 24b, 24c) (25)

$\langle r,t\rangle \models \bigcirc^{4f}(\bigvee_{l=0}^{f} \bigcirc^{l}\mathtt{K}_k(n_k \prec n_i))$ (by MP, 23b,25) (26)

$\langle r,t\rangle \models \bigcirc^{5f}\mathtt{K}_k(n_k \prec n_i)$ (by definition of $\bigcirc$, AS6,26) (27)

$\langle r,t\rangle \models \bigcirc^{5f}\mathtt{K}_k(n_k \succ n_j)$ (by AS6, 6) (28)

$\langle r,t\rangle \models \bigcirc^{5f}\mathtt{K}_j(n_j \prec n_k)$ (by AS6, 13) (29)

$\langle r,t\rangle \models \bigcirc^{5f}\mathtt{K}_i(n_i \succ n_k)$ (by AS6, 20) (30)

$\langle r,t\rangle \models \bigcirc^{5f}\mathtt{K}_k(n_k \prec n_i)$ (by AS6, 27) (31)

$\langle r,t\rangle \models \bigcirc^{5f}(\mathtt{K}_k(n_k \succ n_j) \wedge \mathtt{K}_j(n_j \prec n_k) \wedge \mathtt{K}_i(n_i \succ n_k) \wedge \mathtt{K}_k(n_k \prec n_i))$ (by TP, 28, 29, 30, 31) (32)

$\langle r,t\rangle \models \bigcirc^{5f}(n_i \Cap n_j)$ (by definition of $\Cap$) (33)

The last formula represents the statement of this Lemma.

Lemma 7. *Let a Chord network contain a stable pair and let a node between them leave the network. Then, there is a finite period of time after which the starting pair will be stable again, if no other nodes are trying to join in the meanwhile.*

Proof. We will show that 2 rounds of the stabilization process (i.e., $2f$ steps of an execution of the Chord protocol) will be enough to guarantee that the considered pair of nodes becomes stable.

Let us assume that $n_i, n_j, n_k \in \mathcal{A}(\langle r,t\rangle)$ and $n_i \Cap n_k$, i.e. $(n_i \succ n_j) \wedge (n_j \prec n_i) \wedge (n_j \succ n_k) \wedge (n_j \prec n_k)$ and that n_j tries to leave that stable pair. Let us denote

$$\alpha = \bullet(n_i \Cap n_k) \wedge \rho_{L,j} \bigwedge_{n_l \in I} \bigwedge_{t=0}^{2f} \bigcirc^{t}\neg n_l, I = \{n_l | n_l\mathtt{M}\langle n_i, n_j\rangle, n_k \neq n_l, n_j \neq n_l\}.$$

$\rho_{L,j}$: $\bullet(n_j \wedge n_i \Cap n_k) \wedge n_j\mathtt{M}\langle n_i, n_k\rangle \wedge \neg n_j$
We have,

$\langle r,t\rangle \models \alpha$ (0)

$\langle r,t\rangle \models \mathtt{K}_i(n_i \succ n_j) \wedge \mathtt{K}_k(n_j \prec n_k) \wedge \neg n_j$ (by simplification α) (1)

$\langle r,t\rangle \models \bigvee_{l=0}^{f} \bigcirc\mathtt{K}_i(n_i \succ n_k)$ (by 1, $\rho_{S3,i}$) (2)

$\langle r,t\rangle \models \bigvee_{l=0}^{f} \bigcirc\mathtt{K}_k(u \prec n_k)$ (by 1, $\rho_{S4,k}$) (3)

$\langle r,t\rangle \models \bigcirc^{f}\mathtt{K}_i(n_i \succ n_k)$ (by AS6,2) (4)

$$\langle r,t\rangle \models \bigcirc^{f}\mathtt{K}_k(u \prec n_k) \quad \text{(by AS7, 3) (5)}$$

$$\langle r,t\rangle \models \bigvee_{l=0}^{f} \bigcirc^{f}\mathtt{K}_k(n_i \prec n_k) \quad \text{(by MP, 4,5, } \rho_{S2,i,k}\text{) (6)}$$

$$\langle r,t\rangle \models \bigcirc^{2f}\mathtt{K}_k(n_i \prec n_k) \quad \text{(by AS7, 6) (7)}$$

$$\langle r,t\rangle \models \bigcirc^{2f}\mathtt{K}_i(n_i \succ n_k) \quad \text{(by AS6, 4) (8)}$$

$$\langle r,t\rangle \models \bigcirc^{2f}(\mathtt{K}_k(n_i \prec n_k) \wedge \mathtt{K}_i(n_i \succ n_k) \quad \text{(by TP, 7,8) (9)}$$

$$\langle r,t\rangle \models \bigcirc^{2f}(n_i \Cap n_k) \quad \text{(by definition of } \Cap\text{) (10)}$$

The last formula represents the statement of this Lemma.

Bojan Marinković is Research Assistant Professor at Mathematical Institute of the Serbian Academy of Sciences and Arts. He received his PhD student at The Faculty of Techical Sciences University of Novi Sad, Serbia in 2014. During 2009, he spent three months as a visiting researcher at INRIA Sophia Antipolis, France. His research interests concern: distributed systems, applications of mathematical logic in computer science, automated theorem proving and digitization of cultural and scientific heritage.

Zoran Ognjanović is a research professor at the Mathematical Institute of the Serbian Academy of Sciences and Arts. He received his PhD degree in mathematical logic from University of Kragujevac, Serbia, in 1999. He has authored or coauthored over 70 (chapters of) monographs and technical papers in major international journals and conferences. His research interests concern: applications of mathematical logic in computer science, artificial intelligence and uncertain reasoning, automated theorem proving, applications of heuristics to satisfiability problem and digitization of cultural and scientific heritage. He is a recipient of the Serbian Academy of Sciences and Arts Award in the field of mathematics and related sciences for 2013 and the annual award of Serbian Ministry of Science for results in fundamental research in 2004.

Paola Glavan, PhD, is currently research and teaching assistant at the University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture-FSB. Her main scientific interests include using logic and logical methods in analyzing distributed processes and protocols. In particular, she is interested in Abstract State Machines and their application to semantics of programming languages and distributed protocols and the use of temporal epistemic logic in describing and verifying distributed protocols.

Anton Kos received his Ph.D. in electrical engineering from University of Ljubljana, Slovenia, in 2006. He is an assistant professor at the Faculty of Electrical Engineering, University of Ljubljana. He is a member of the Laboratory of Information Technologies at the Department of Communication and Information Technologies. His teaching and research work includes communication networks and protocols, quality of service, dataflow computing and applications, usage of inertial sensors in biofeedback systems and applications, signal processing, and information systems. He is the (co)author of more than thirty papers appeared in the international engineering journals and of more than fifty papers presented at international conferences.

Anton Umek received his Ph.D. in electrical engineering from University of Ljubljana, Slovenia, in 1999. He is currently an assistant professor at the Faculty of Electrical Engineering, University of Ljubljana. He is a member of the Laboratory of Information Technologies at the Department of Communication and Information Technologies. He is a member of the research program Algorithms and optimization methods in telecommunications that was two years in a row the best research program financed by the Slovenian research agency. Since last year he is the leader of industrial research and development projects in designing of sensor based smart sport equipment and sensor based forestry machinery. His teaching and research work includes signal processing, digital communication, secure communications , access network technologies and design of sensor supported sport training systems. He is the (co)author of eight papers appeared in the international engineering journals and of more than thirty papers presented at international conferences. Anton Umek is a member of IEEE and between 2015 and 2018 he was the Slovenian section ComSOC chapter chair.

Received: November 15, 2018; Accepted: August 8, 2019.