ECAI 2012
Luc De Raedt et al. (Eds.)

doi:10.3233/978-1-61499-098-7-714

Institutionalised Paxos Consensus

David Sanderson and **Jeremy Pitt**[1]

Abstract. We address the problem of maintaining consistency in systems that are open, decentralised and resource-constrained, where the system components are highly mobile and/or 'volatile'. An example of these systems is found in vehicular ad hoc networks (VANET). Self-organising and norm-governed electronic institutions have been proposed to address these issues, but the problem of maintaining the consistency of conventionally-agreed values (institutional facts) arises due to the fragmentation/aggregation of component clusters, role failure, and revision of agreed values.

In this paper, we specify IPCon, an algorithm for Institutionalised Paxos Consensus, which is an extension of the well-known Paxos consensus algorithm for maintaining consistency in static distributed databases. The 'classic' Paxos algorithm is modified to accommodate role-based institutionalised power and extended to allow for dynamic clusters that may change, merge or fragment. We further extend IPCon to allow for the revision of previously agreed parameter values. A proof of correctness for IPCon is given, along with details of an axiomatisation and executable specification of the algorithm in the Event Calculus. These results show that IPCon is a viable method for coordination, consensus formation and collective-choice in self-organising multi-agent systems using electronic institutions.

1 INTRODUCTION

Systems that are open, decentralised, resource-constrained and highly mobile present unique challenges in their organisation. Vehicular networks that can provide a platform for many kinds of communication application or physical optimisation of transportation are an example of this kind. Such networks have been proposed as the basis of sensor networks for monitoring pollution or traffic density [8], and as the basis for physical structures called 'road trains' or 'platooning' which can optimise fuel and road-space usage [1].

Both sensor networks and physical constructs require autonomous and heterogeneous agents (associated with each car, or network node) to form opportunistic alliances, or clusters. These clusters have to be managed by forming a consensus on certain values and parameters; for example, the agreed top speed and separation distance in a platoon. However, this consensus is susceptible to temporary or permanent node failure, circumstantial changes like cluster aggregation and fragmentation, and environmental change beyond the control of the agents (e.g. congestion requiring a reduced top speed).

Maintaining consistency in open decentralised systems, especially those such as vehicular networks, requires robust algorithms for consensus formation. We manage consensus formation in such networks by using self-organising electronic institutions (cf. [18, 2]). The institution members (agents) are required to maintain mutable institutional facts in the face of inadvertent failure, institutional fragmentation/aggregation, and environmental change. Thinking of an electronic institution as a distributed database where the authority to write to institutional facts is role-dependent suggests the use of the well-known Paxos consensus algorithm [12, 14] developed for fault-tolerant management of distributed databases. However, Paxos is capable of addressing only some of these problems, so we adapt it for use with electronic institutions, dynamic roles and cluster membership, and mutable values. We design and specify IPCon, an Institutionalised Paxos Consensus algorithm, that provides resilient collective-choice arrangements in self-organising electronic institutions in the face of cluster fragmentation/aggregation, role failure in situations where authority is role-dependent, and the need to revise an agreed value.

The paper is structured as follows; in Section 2 we state the problems we address, as contextualised for VANETs, before briefly explaining classic Paxos in Section 3. Section 4 describes the derivation of the IPCon algorithm from Paxos and goes on to give details of its axiomatisation using the Event Calculus [11] as well as explaining resilience to role failures and institutional fragmentation/aggregation. Our proof of correctness is given in Section 5 by relating the requirements for safety to the properties of classic Paxos we have implemented in IPCon. Section 6 provides a summary of related and further work. We conclude in Section 7 that IPCon provides role-based institutionalised power that allows for dynamic clusters that modify previously agreed institutional facts and is a viable method for coordination, consensus formation and collective-choice in self-organising multi-agent systems using electronic institutions.

2 PROBLEM SPECIFICATION

In contrast to related work on adaptation in electronic institutions by machine learning, we use institutional power [10] for consensus on conventionally agreed institutional facts showing that safety properties can be preserved in a cooperative open system, especially as clusters aggregate and fragment. The inclusion of institutional power presents challenges that we address in addition to the problems of maintaining consistency in static distributed systems; these are given below and indicated diagrammatically in Fig 1 when taken in the context of vehicular networks (note that although we use the motivating example of vehicular networks throughout, these problems are common to any electronic institution, self organised system, or multi-agent system requiring groups to mutually agree on values):

- Fragmentation/Aggregation — Dynamic open systems of the sort we are interested in will entail cluster aggregation and fragmentation as agents join and leave the clusters, and clusters merge and split over time. In any mobile ad-hoc network, nodes in the network may physically move out of range of some networks and into range of other networks. Even in non-mobile networks, there may be reasons for the membership of the network to change.

[1] ISN Group, EEE Dept, Imperial College, Exhibition Road, London, England, SW7 2BT. {dws04|j.pitt}@imperial.ac.uk

- Revision of conventionally agreed values (parameter change) — The value that is chosen for an institutional fact may not always be the best value, so we must deal with its mutability in a consistent manner. In VANETs for example, the vehicle platoon may agree on a speed to travel at and then come to an area of road that is more congested; this would then require them to alter the previously agreed speed to something lower.
- Role failure — We must ensure robustness against role failure as the loss of a role-fulfilling agent may impede the proper functioning of the institution. In a voting system where only one member has the authority to declare that a motion has passed, no motions can be passed if that member fails; having a single point of failure is undesirable in many different types of systems.

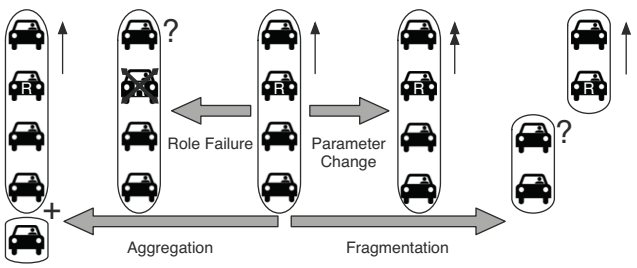

Figure 1. Problems to address

A source for a potential solution to the issues of resilience and fault-tolerance in open systems is in the domain of distributed systems. For example, a well-known algorithm for designing fault-tolerant distributed systems under certain conditions is Paxos.

3 PAXOS

Paxos is an algorithm for implementing a fault-tolerant distributed system using the state machine approach [12] as shown in Fig 2. Fault tolerance is achieved by executing an infinite sequence of separate independent *instances* of the Paxos algorithm (where each instance is a run of the algorithm to decide a command to be sent to the distributed state machine) which contain numbered *ballots*. Each ballot decides on the value of an instance; it is orchestrated by a *leader*, voted on by *acceptors*, and *learners* act as redundant storage. Values are proposed by *proposers*. Each leader may choose ballot numbers from an unbounded but individual set of natural numbers; it is unbounded to allow 'infinite' ballots to occur, and individual so that it is impossible for another leader to begin a ballot that has already taken place. The Paxos consensus algorithm operates on the undecided commands before they are sent to the "state machine" to ensure that all the distributed nodes in the fault-tolerant system agree on the value of those commands. Once they have been decided, they are unalterable and each node can execute the same commands in the same order, thus ensuring that each distributed node has the same state.

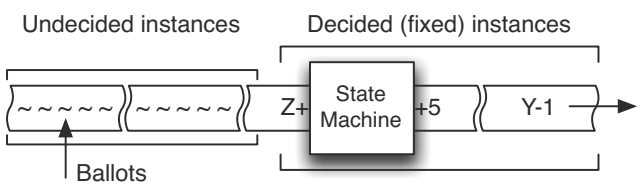

Figure 2. Paxos state machine approach

Different versions of Paxos exist to deal with different sets of constraints and to address different types of fault; here we explain the most basic form, "classic Paxos" [12, 13]. The algorithm assumes that agents operate asynchronously at arbitrary speed, may fail by stopping, and may restart, but may not lie or impersonate other agents. Furthermore, messages can take arbitrarily long to be delivered, can be duplicated, and can be lost, but cannot be corrupted – this is termed "non-Byzantine communication" by Lamport.

As a consensus algorithm, Paxos has three safety requirements:

S1 Only a value that has been proposed may be chosen,
S2 Only a single value is chosen, and
S3 A process never learns that a value has been chosen unless it actually has been.

To maintain safety, all versions of Paxos rely on the fact that all possible *quorums* have at least one member in common [12, Sec 2]. This results in the assurance that f failures can be tolerated in a system of $2f + 1$ processes. This defines the term *quorum* in the context of all Paxos derivatives.

Paxos maintains the following 'normative' properties:

P1 An acceptor can vote for value v in ballot b only if v is *safe at* b.
P2 The acceptors may not vote for different values in one ballot.
P3 A value v is safe at ballot number b if no value other than v has been, or ever can be, chosen in any ballot numbered less than b.

Once a quorum of acceptors has voted for a value v in a ballot b, the value is said to have been *chosen* and should be communicated to any additional non-acceptor *learners* if this is relevant. Once a value has been chosen, then in any further proposals in that instance, the leader will be restricted to the chosen value by P3 and the restriction that all quorums must have at least one acceptor in common. This means that further proposals in that instance will retrieve the chosen value, rather than choose one.

Although Paxos provides fault-tolerant consensus in a static distributed system, current versions are limited in their application in opens systems, in so far as they presume a static set of nodes where the set of decision-makers (that is, the cluster) does not change, aim for consensus on a single value for each instance that will not and cannot change, and presume temporary failures with the possibility of restart where the cluster size remains constant. Our solution is to design a new algorithm to overcome these limitations by explicitly representing different dynamic clusters of agents, allowing conventionally agreed values to be changed in a coherent manner without causing confusion between different nodes, and providing resilience against permanent role failures and the departure of an agent, which is equivalent to a permanent failure. This new algorithm is IPCon, an algorithm for Institutionalised Paxos Consensus.

4 IPCON SPECIFICATION

Paxos operates by deciding commands that are to be sent to a state machine; the static set of nodes means that all of the decision makers are also maintaining a copy of the state machine, so the state of the machine itself (the values in the database) doesn't need to be expressed explicitly by the Paxos algorithm. This is because all the nodes are guaranteed to start with the same state and have the same changes applied to them.

Our requirement for dynamic clusters means that we work on two levels of abstraction at the same time; the changes to send to the database and the database itself. We therefore explicitly represent

the values and operate on them, not on the commands to be sent. We begin by mapping an *instance* of the Paxos algorithm to an *issue* in the IPCon algorithm to allow multiple separate instances of Paxos to be related and reasoned about together in one run of IPCon.

Changing a value in the database is no longer simply a case of issuing a new command to change it; it now requires modifying an agreed value, so we wrap it in a new concept of a *revision* of an issue. This adds to our mapping between Paxos and IPCon by relating a Paxos instance to a revision of an issue in IPCon. This requires a new method for revising a previously agreed value.

While Paxos is designed to deal with temporary node failure, the dynamic membership of clusters and the explicit representation of institutional power requires new methods for dealing with cluster aggregation and fragmentation and permanent role (node) failure.

Contrast the presentation of Paxos in Fig 2 with that of IPCon in Fig 3. Rather than instances of Paxos operating on commands that are yet to be sent to the "machine", the IPCon issues are now institutional facts inside the "machine" that the agents operate on. Despite this change, the independence of Paxos instances from each other means that we maintain the properties that are internal to each instance of Paxos for each issue in IPCon.

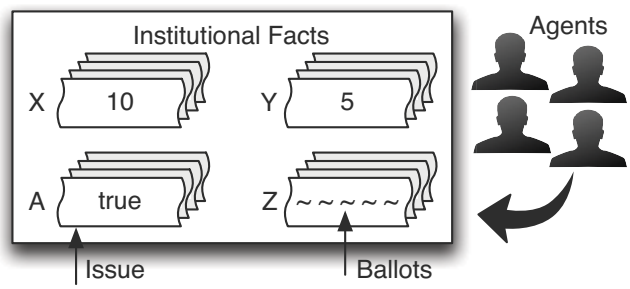

Figure 3. IPCon institutional approach

4.1 Definitions

Cluster Agents are organised into clusters that represent self-organising electronic institutions. In our work on Intelligent Transportation Systems, these clusters are groups of vehicles travelling along the road together for some period of time.

Quorum Any set of agents in the cluster is considered a quorum if it has at least one member in common with every other possible quorum. In a cluster of N agents, any set of agents that has $> \frac{N}{2}$ members is therefore a quorum.

Issue Clusters have a number of institutional facts to agree on; in our motivating example of a vehicular network examples could be the agreed speed of the group, the space to be left between vehicles, the order in which the front vehicle in a road-train should rotate, and so on. These issues are specific to each cluster.

Value Each issue has a value associated with it at any given time that is agreed upon by the members of the cluster. These are the values that we use consensus to agree on. In our example, for the issue of `speed` the value could be `70mph`.

Ballot In Paxos and IPCon, the values for each issue are decided using a sequence of numbered ballots that are orchestrated by a leader. In each ballot, the leader submits a value for the issue and the acceptors may either vote for it, or abstain. The process by which a ballot proceeds is given in Section 4.3.1.

Roles Leader, Acceptor, Learner, and Proposer are the roles in Paxos and they are also required in our institutions; each role has different associated powers, permissions, and obligations. We have introduced powers to open the system, and constrained the actions of the agents with permissions and obligations.

We introduce the following concepts not present in any existing Paxos variant:

Revision The value that is chosen for an issue may not always be the most suitable value. For example, over time the ideal speed for a group of vehicles may change due to any number of reasons. As Paxos is explicitly designed to prevent agreed values changing, we introduce the concept of a revision to allow values to change safely. Each revision is unaffected by the previous revisions and represents a 'clean slate'. See Section 4.3.4 for more detail.

Self organisation The cluster requires a leader to coordinate the process of the algorithm; this central role should be appointed from within the system by the autonomous components themselves using self-defined rules. The leader needs to manage the cluster by granting or removing roles from agents, as well as adding or removing them from the cluster. We provide actions allowing any agent to arrogate or resign the role of leader. See Section 4.3.5 for more details.

4.2 Properties and Assumptions

IPCon maintains the assumptions of asynchronous communication, possibility of failure and non-Byzantine communication. Indeed, as our motivating example is that of Intelligent Transportation Systems, it does not make sense to consider the possibility of deceit in systems that are clearly safety-critical. It is another problem to consider a Byzantine Institutionalised Paxos Consensus algorithm.

Unlike Lamport, we give a precise specification of the requirements for liveness and progress; the dynamic nature of our application means that clusters may change frequently, but we aim for a system that will eventually result in a chosen value being learned. A non-faulty quorum is required for liveness, as the algorithm cannot change state without a quorum of agents. Progress is guaranteed by ensuring that eventually a single leader exists as the only one trying to issue proposals, and it can communicate successfully with a quorum of (non-faulty) acceptors [13, Sec2.4]. The liveness and progress requirements are given as L1 and L2. We assume that if a non-faulty leader exists, it can communicate successfully with the non-faulty quorum of acceptors, by virtue of them all being non-faulty.

L1 At least one non-faulty quorum of acceptors must exist, and
L2 Eventually (only) one non-faulty leader must exist.

Lamport describes the design of the Paxos consensus algorithm as following "almost unavoidably from the [Safety] properties we want it to satisfy" [13, Sec1]. We design IPCon such that it maintains institutional 'normative' versions of the properties maintained by classic Paxos. The clear difference is that we explicitly include multiple issues and clusters, and allow values to be *revised*. The properties are derived from Paxos by mapping "ballot number b" to "ballot number b on revision r of issue i in cluster c", and terming it "b'". P1 and P2 are now normative rules, but remain the same, whilst P3 is broken into three in translation to normative terminology. P3a, 3b, and 3c are a definition of a *safe* value expressed through normative states.

P3a If no empowered acceptor in the quorum voted in a ballot numbered less than b', all values are *safe at* b'.
P3b If an empowered acceptor in the quorum has voted, let c be the highest-numbered ballot less than b' that was voted in. The value voted for in ballot c is *safe at* b'.

P3c A value v is *safe at* b' if no empowered acceptor in the quorum has voted for any value other than v in a ballot less than b in a revision equal to r of issue i in cluster c. Likewise, all values are safe at b if no empowered acceptor has voted for any value in any revision equal to r or greater.

We use a conceptual construct of the "highest numbered ballot" (**hnb**) to explain which values are safe for given (usually quorum-sized) sets of agents. The **hnb** for any set of agents Q is the ballot and value that was voted for in the highest-numbered ballot in which an agent in Q has voted. The set of **hnb** for a set C of overlapping quorums (ie, a cluster) is simply the set containing the **hnb** of the quorums $Q_{1...n}$ in C. It is important to note that as both the ballot number and the value that is voted for determine which values are safe, the **hnb** encapsulates a $\langle ballot, value \rangle$ pair.

4.3 IPCon algorithm

In this section we will describe the functioning of IPCon by detailing the parts of it algorithmically using the Event Calculus implemented in Prolog. Lamport observes in [14, Sec 8] that voting is a refinement of consensus. We take advantage of this to modify classic Paxos to create a single collective choice algorithm, IPCon.

4.3.1 Main message flow

This gives the standard flow of the protocol that decides on a value for an issue as adapted from classic Paxos ([13, Sec 2]).

0A The proposer sends a `request0a` message to the cluster leader requesting to know the value of an issue if it has one, or to propose a value if it does not.

1A The leader sends a `prepare1a` message to all empowered acceptors with a ballot number b' it has chosen.

1B On receipt of a `prepare1a` message from the leader, an empowered acceptor responds with a `respond1b` message containing the number of the **hnb** that they voted in, and the value they voted for. If they have not yet voted, then they indicate this by replying with a `null`. This message also represents a promise from the acceptor not to participate in any ballot numbered lower than b'. If the leader receives `respond1b` messages stating that some acceptors have voted in higher-numbered ballots than b', this indicates that it should retry with a higher ballot number.

2A Once it has received `respond1b` messages from a quorum of empowered acceptors, the leader chooses a value v that has previously been proposed and is safe at b' (see Section 4.2) and sends a `submit2a` message to the empowered acceptors asking them to vote for this submitted value.

2B On receipt of a `submit2a` message from the b' leader, an empowered acceptor either votes for the value v in ballot-b' by sending a `vote2b` message, or abstains by sending no message. An acceptor cannot vote in a ballot if it has already voted in a higher-numbered ballot on the same issue. A value is chosen once a quorum of agents have voted for it.

4.3.2 Threats to safety

Cluster fragmentation and cluster aggregation are related problems; in both cases the set of acceptors will change and this will cause a change in the set of **hnb** for the cluster; we serialise changes for simplicity. We provide functions for agents to leave the cluster, and for the leader to grant and remove agents' roles.

When two clusters aggregate or a new acceptor joins, there is at least one extra agent in the set of **hnb** that has not voted. If a value has not already been chosen, then this is not a problem because a consensus that does not exist cannot be violated. If however a value has been chosen, we must take care that further ballots cannot be made for values that were not safe before the agent joined. We give all new agents a choice to 'sync' with the cluster's agreed values as will be explained in Section 4.3.3 to alleviate this potential problem.

It is only possible for safety to be violated if a new quorum-sized group is created that does not have the chosen value as its **hnb**. The definition of a quorum means that this is only possible if there is already a group of size $new\_quorum\_size - 1$ that doesn't have the chosen value as its **hnb** (so this new agent will make a group of new_quorum_size). This matches with our intuition by only occurring when you have equally sized groups that voted *for* the value, and *not for* the value; the tipping point is when both groups are of size $new\_quorum\_size - 1$ (quorumsize is $\lfloor \frac{N}{2} + 1 \rfloor$ for N processes).

When a cluster fragments or an acceptor leaves the cluster, its votes no longer count. If a value had been chosen and the leaving acceptor had voted for it, the value may be 'unchosen' by causing another value to become safe. A value is safe if a $quorum\_size$ sized group of acceptors has it (or "null") as their **hnb**, so when an acceptor that voted for the currently chosen value leaves the cluster, and if $\#votes\_for = \#votes\_notFor = old\_quorum\_size - 1$ before the removal, it will be the case that $\#new\_votes\_for + 1 = \#votes\_notFor = new\_quorum\_size$ after the removal. This removes a quorum-sized group of agents that voted for the previously chosen value. Similarly to the addition of acceptors, this is a 'tipping point' that we watch for.

Both tipping points initiate an Event Calculus fluent as shown in Fig 4 to indicate that we are at risk of violating safety; when an acceptor does actually join or leave, these fluents are checked. If they hold at the time, then an obligation to revise is initiated. The procedures in which this occurs are given algorithmically below, and revision is then explained in further detail.

```
holdsAt(possibleAddRevision(R, I, C) = true, T) :−
    holdsAt(sync(_, Val, R, I, C) = true, T),
    numberOfVotesForValue(V, R, I, C, T, For, Against),
    For = Against.
holdsAt(possibleRemRevision(V, R, I, C) = true, T) :−
    highestVote(V, R, B, I, C, T),
    numberOfVotesForValue(V, R, I, C, T, For, Against),
    For = Against.
```

Figure 4. Code fragments

4.3.3 Acceptor addition and loss

When an acceptor joins the cluster the consistency of any previously chosen values must be maintained. The following procedure ensures this; if no value has been chosen then this is not required.

1. A new acceptor joins the cluster, either due to a merge or by being given the role of acceptor by the leader.
2. The leader sends a `syncReq` message to the new acceptor indicating that a value v has previously been chosen.
3. The new acceptor replies to the leader with a `syncAck` message.
   - If the acceptor agrees with v, then they include it in their reply and no further action is required.

– If the acceptor replies with "*no*", they do not agree with the chosen value and the process continues.

4. As the acceptor has not agreed with the choice of v as the value, the value of `possibleAddRevision` (Fig. 4) must be checked; if it holds then it might be possible for the new agent to violate safety so an obligation to revise is initiated.

As with addition, the loss of an acceptor requires our attention when a value has been chosen previously. As the acceptor is leaving however, it is simpler to deal with as no interaction is required. If no value has been chosen, then this process is not required.

1. An acceptor leaves the cluster; either by choice, due to the cluster fragmenting, or by having the role of acceptor removed.
   – If it had not voted for the chosen value, there is no possibility of safety being violated, so no further action is required.
   – If it had voted for the chosen value, then the value of `possibleRemRevision` (Fig. 4) must be checked; if it holds then it might be possible for safety to be violated, so an obligation to revise is created.
2. In either case, all previous votes by the acceptor are erased from the system or otherwise marked as "no longer valid".

4.3.4 Revision

By ballot ordering, the promise in **1B** of Section 4.3.1, and the restriction on voting in **2B**, lower-numbered ballots cannot progress once a higher-numbered ballot has been interacted with by a $quorum\_size$ sized group of acceptors. Our method of tagging ballot numbers with revision numbers extends the ballot numbering system so that 'older' revisions of an issue are explicitly out of date. We can therefore say that revising an issue safely 'un-choses' a value by resetting the **hnb** set so that all values are safe. Revision can then be seen as a reset switch that does not affect safety except in the way that we want it to; namely by allowing us to change our minds on the chosen value. Revising also closes all `sync` fluents and active ballots.

4.3.5 Leadership

The details of electing a process to the role of leader are left as an implementation detail by Lamport [13, Sec 2.4] under the assumption that a single agent will eventually have the role of leader. Our method has no impact on the values that are chosen as no values are referenced by the commands, so has no effect on safety. We provide the commands `arrogateLeadership` and `resignLeadership`; the former allows any agent to claim the leadership of the cluster, and the latter allows any agent with the role of leader to resign. We presume some mechanism or agreement whereby the agents will organise a solution resulting in only one leader at any given time. As explained by Lamport in [13], duelling leaders can only impede progress, and we allow agents to leave a cluster; this means that if a duelling leaders situation occurs, agents may leave the cluster and reform without them in order to continue progression. Once a "new" cluster is formed by fragmentation, one of the agents should arrogate the leadership so that new values can be chosen.

5 CORRECTNESS PROPERTIES

The specification of IPCon is in Prolog, hence it is it own implementation (i.e. executable specification) and we can run it before submitting queries for given algorithmic states or situations, and then check that the result is what we expect. We have run it with clusters of agents with valid and invalid narratives (sequences of events) and exhaustive testing provides empirical evidence supporting the correctness proof shown below. The full sourcecode for the executable specification is available online[2].

The proof of correctness is too verbose to include fully due to space constraints; we sketch the proof here with a focus on the important points. Lamport gives a proof by derivation, as a derivation is "an implementation proof written backwards" [14, Sec 1]. He uses P1–3 to implement a solution that guarantees S1–3 on the assumption of L1&2 imprecisely stated. We have implemented the core of our algorithm almost identically to classic Paxos, maintaining the same message flow and adhering to P1–3; we therefore will show only that P1–3 results in S1–3 and explain how our additions do not violate S1-3 and assist in the fulfilment of L1&2.

Proof of S1: Only a value that has been proposed may be chosen. A value can only be chosen by being voted for (by the definition of 'chosen'). A value can only be voted for in a ballot by empowered agents (by P1 and the definition of 'vote'). A value has to have been proposed by an agent with the role of proposer to be submitted to a ballot (by step **2A** in Section 4.3.1). □

Proof of S2: Only a single value is chosen. A value can only be chosen by being voted for (by the definition of 'chosen'). A value can only be voted for in a ballot by empowered agents (by P1 and the definition of 'vote'). A ballot must have a "safe" value (by P1). A safe value cannot have been chosen before, and once a value has been chosen that is the only safe value (by P3, so can't choose different values in two ballots). Ballots can only have a single value (by P2, so can't choose multiple values in the same ballot). □

Proof of S3: Only chosen values are learned. Assume that agent L learns that value V was chosen when it has not actually been chosen. If it has not been chosen, there is no group of agents of size Q that have voted for it (by the definition of 'chosen' where Q is the quorum size). By the definition of a 'quorum' and institutional membership requirements, all quorums have at least one member in common. By the definition of 'learning', L must have observed at least Q acceptors voting for V. This is a contradiction. □

Correctness of revision. Our concept of revision by adding another part to the ballot number can be seen as equivalent to starting a new instance of Lamport's classic Paxos. In classic Paxos there is no possibility of different instances interfering with each other, so we only have to consider the requirement for agents to know when a value has been revised. This can be accomplished by broadcasting the act of revision to all agents - this is in fact required, because the acceptors must be informed that they now have the role of acceptor in a new revision of the issue. If an agent does not receive this broadcast for whatever reason, we must still ensure that they are informed of the change. This does not require any additions to the protocol, as all messages from **1A** onwards contain the ballot number, which includes the revision number. Therefore, it is not possible for a functioning agent to be unaware of a revision having taken place, or for a lower-numbered revision to continue (Section 4.3.4).

6 FURTHER & RELATED WORK

Issues of teamwork, consensus, and coalition formation have been well studied in the multi-agent systems literature, for example from

[2] https://s3-eu-west-1.amazonaws.com/dws04-academic-storage/IPCon.zip

perspectives of evolutionary game theory [6], opinion formation in social networks [9], rules and agent attitudes [5], computational social choice theory for design and evaluation of preference selection mechanisms [3], and learning norms in agent institutions [15]. IPCon is the first algorithm to leverage a distributed systems algorithm with well-established safety properties, use it as the foundation for a self-organising system using institutionalised power and dynamic norm-governed specifications [2], and show that those safety properties are preserved in an open system as clusters aggregate and fragment.

Similarly, the idea of 'platooning' in VANETs has received much attention [1, 8, 17], motivated by the desire to reduce fuel consumption and lower carbon emissions. However, all these approaches take either a centralised approach, or rely on emergence based on local interactions with respect to physical rules and 'brute' facts. Our approach uses a form of organised adaptation which uses local communication and computation with respect to physical and conventional rules, lifting the 'brute' facts to conventionally-agreed institutional facts. This provides a greater degree of flexibility and fine-grained control over cluster parameters, and allows a 'memory' of potentially advantageous $\langle issue, value \rangle$ pairs to be preserved as clusters fragment and aggregate, rather than having to re-emerge the values each time a cluster forms or fragments. This 'memory' could also be used as the basis for machine learning or case-based reasoning algorithms to generate new institutional regulations based on experience [15].

In further work, we intend to extend this research by integrating it as the meso level of a three tier micro-meso-macro constitution of Intelligent Transportation. At the micro-level there are interacting agents, at the macro-level there are global system properties, while institutions operate at the meso-level to encourage trust, reciprocity and collective action which map micro-level behaviour into desired macro-level outcomes [16]. We note that a similar concept has been proposed in the work of Dopfer [4] on micro-meso-macro level economic analysis. To accomplish this, we need to implement the IPCon algorithm in an agent architecture with anticipatory expectations and embed this in a network animation tool which can simulate metropolitan area networks with a large population of agents.

A more theoretical refinement of IPCon concerns judgement aggregation [7]. Currently agents converge on, maintain and update values for what are, logically, multi-valued fluents. We propose investigating the use of Paxos in converging on values for propositional *formulas*, where the agreed truth of a formula in a distributed system is not necessarily given by summing the truth values of component propositions and applying the traditional meaning of the connective to the summed values.

7 SUMMARY & CONCLUSIONS

This paper has presented IPCon as a method for robust collective-choice in electronic institutions. After identifying a problem in open, decentralised and resource-constrained systems, we have modified Paxos and applied it at different levels of abstraction in a new domain to create the IPCon algorithm. IPCon includes an explicit representation of both institutional roles and power and of mutually agreed mutable institutional facts; this allows the internals of the electronic institution to be reasoned about by the member agents. We have allowed the clusters comprising the electronic institutions to themselves be dynamic in that their size and membership may change permanently; this has been achieved by providing a set of methods for the automated resolution of clusters fragmenting or merging.

We have also provided a proof of correctness adapted from Lamport's proof for the classical Paxos algorithm to show that the original properties and their normative equivalents still hold. The preservation of the normative properties in IPCon is due to the essential robustness of the underlying Paxos algorithm. By giving a formal definition of IPCon in the Event Calculus, we substantiate the proof of correctness through an executable specification. This implementation can be animated to demonstrate that original safety properties and their normative equivalents still hold and that it provides resilience to role failures and institutional fragmentation and aggregation.

Paxos is a remarkable achievement; its longevity and wide applicability is indicative of its core quality and this allows us to leverage it without structural (syntactic) changes. Our innovative contribution to the Paxos canon is generalising it to institutional settings in which empowered agents perform designated acts to assert facts, making it applicable to the class of open distributed systems of particular importance to multi-agent systems research.

In conclusion, this research contributes to the general field of electronic institutions, but differs from those of others [18] in its explicit representation of institutional power, which means our work is applicable to open distributed systems in which reasoning about roles, counts-as and context is necessary and significant. The work is also a part of an ongoing research program on the formalisation of socio-economic principles for enduring institutions, inspired by those given by Ostrom [16]. IPCon in particular encapsulates the third principle that requires the participation of the institution members; nothing can be done unless a quorum of members agrees on it.

ACKNOWLEDGEMENTS

This research was supported by an IBM Faculty Award.

REFERENCES

[1] The SARTRE Project http://www.sartre-project.eu/.
[2] A. Artikis, 'Dynamic Specification of Open Agent Systems', *Journal of Logic and Computation*, (2011).
[3] Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet, 'A short introduction to computational social choice', *SOFSEM 2007: Theory and Practice of Computer Science*, 51–69, (2007).
[4] K. Dopfer, J. Foster, and J. Potts, 'Micro-meso-macro', *Journal of Evolutionary Economics*, **14**(3), 263–279, (2004).
[5] B.M. Dunin-Keplicz and R. Verbrugge, *Teamwork in Multi-Agent Systems: A Formal Approach*, Wiley, 2010.
[6] J.M. Epstein and R. Axtell, *Growing artificial societies: social science from the bottom up*, The MIT Press, 1996.
[7] S. Hartmann and G. Pigozzi, 'Aggregation in multi-agent systems and the problem of truth-tracking', *AAMAS vol*, **5**, 674–676, (2007).
[8] T. He, K.W. Lee, N. Sofra, and K.K. Leung, 'Utility-based Gateway Deployment for Supporting Multi-domain DTNs', *SECON*, (2010).
[9] R. Hegselmann and U. Krause, 'Opinion dynamics and bounded confidence models, analysis, and simulation', *JASSS vol*, **5**(3), (2002).
[10] A. J. I. Jones and M. Sergot, 'A Formal Characterisation of Institutionalised Power', *Logic Journal of IGPL*, **4**(3), 427–443, (1996).
[11] R. Kowalski and M. Sergot, 'A logic-based calculus of events', *New generation computing*, **4**(1), 67–95, (1986).
[12] L. Lamport, 'The part-time parliament', *ACM TOCS*, **16**(2), 133–169, (1998).
[13] L. Lamport, 'Paxos made simple', *ACM SIGACT News*, **32**(4), 18–25, (2001).
[14] L. Lamport, 'Byzantizing Paxos by Refinement', *DISC*, 211–224, (2011).
[15] J. Morales, M. López-Sánchez, and M. Esteva, 'Using Experience to Generate New Regulations', in *22nd IJCAI*, 307–312, (2011).
[16] E. Ostrom, *Governing the commons*, CUP, 1990.
[17] K. Sampigethaya, L. Huang, M. Li, and R. Poovendran, 'CARAVAN: Providing location privacy for VANET', *Constraints*, 1–15, (2005).
[18] C. Sierra, JA. Rodriguez-Aguilar, and P. Noriega, 'Engineering multi-agent systems as electronic institutions', *European Journal for the Informatics Professional*, **4**, (2004).