A Norm-Governed Systems Perspective of Ad Hoc Networks

Alexander Artikis[1], Lloyd Kamara[2], and Jeremy Pitt[2]

[1] Institute of Informatics & Telecommunications,
NCSR "Demokritos", Athens, 15310, Greece
[2] Intelligent Systems & Networks Group,
Electrical & Electronic Engineering Department,
Imperial College London, SW7 2BT, UK
a.artikis@acm.org, {l.kamara, j.pitt}@imperial.ac.uk

Abstract. Ad hoc networks are a type of computational system whose members may fail to, or choose not to, comply with the laws governing their behaviour. We are investigating to what extent ad hoc networks can usefully be described in terms of permissions, obligations and other more complex normative relations, based on our previous work on modelling norm-governed multi-agent systems. We propose to employ our existing framework for the specification of the laws governing ad hoc networks. Moreover, we discuss a software infrastructure that executes such specifications for the benefit of ad hoc network members, informing them of their normative relations. We have been developing a sample node architecture as a basis for norm-governed ad hoc network simulations. Nodes based on this architecture consider the network's laws in their decision-making, and can be individually configured to exhibit distinct behaviour. We present run-time configurations of norm-governed ad hoc networks and indicate design choices that need to be made in order to fully realise such networks.

1 Introduction

An *Ad Hoc Network* (*AHN*) is a transient association of network nodes which inter-operate largely independently of any fixed support infrastructure [30]. An AHN is typically based on wireless technology and may be short-lived, supporting spontaneous rather than long-term interoperation [31]. Example AHNs are formed by the devices of consumers entering and leaving an 802.11 wireless hot spot covering a shopping mall (for buying/selling goods consumer-to-consumer style by matching potential buyers and sellers); by participants in a workshop or project meeting (for sharing and co-authoring documents); or by emergency or disaster relief workers, where the usual static support infrastructure is unavailable.

An AHN may be visualised as a continuously changing graph [30]: connection and disconnection may be controlled by the physical proximity of the nodes or it may be controlled by the nodes' continued willingness to cooperate for the formation, and maintenance, of a cohesive (but potentially transient) community.

An issue that typically needs to be addressed when managing and maintaining an AHN is that of routing. AHNs are not usually fully connected; participating nodes are often required to act as routers to assist in the transport of a message (packet) between sender and receiver nodes. Resource-sharing is another challenge that needs to be addressed during the life-time of an AHN; the participating nodes compete over a set of limited resources such as bandwidth. AHNs are often specifically set up for sharing a resource such as broadband Internet access, processor cycles, file storage, or a document in the project meeting example mentioned above.

The nodes of an AHN are programmed by different parties — moreover, there is no direct access to a node's internal state and so one may only make inferences about that state. It is possible, even likely, that the nodes of an AHN will fail to behave as they ought to — for example, a node acting as a router may move out of communication range or run out of power and may therefore be unable to forward packets. Furthermore, system components may intentionally mis-behave in order to seek unfair advantage over others. A 'selfish' node, for instance, may refuse to forward packets for other nodes while gaining services from these nodes [42]. The controller of a resource may grant access to the limited resource based on personal preferences rather than an agreed metric. Moreover, due to the (typically) wireless nature of AHNs, a participating node should be prepared to counteract against rogue peers. For all of these reasons, an AHN needs to be 'adaptable' — that is, it should be able to deal with 'exceptions' such as the ones mentioned above.

Within the EPSRC-funded Programmable Networks Initiative, we are investigating to what extent adaptability can be enhanced by viewing AHNs as instances of *norm-governed* systems, that is, systems in which the actual behaviour of the members is not always ideal and, thus, it is necessary to express what is permitted, prohibited, obligatory, and possibly other more complex normative relations that may exist between the members [18]. We have developed a framework for an executable specification of norm-governed multi-agent systems (ngMAS) that defines the social laws governing the behaviour of the members of such systems [1–3] (we will use the terms 'social law' and 'norm' interchangeably). We propose to use this framework as an infrastructure for the realisation of norm-governed AHNs (ngAHN)s.

The remainder of this paper is organised as follows. First, we give an overview of ngAHNs. More precisely, we review our work on specifying ngMAS and propose ways to apply this work to AHNs. Second, we discuss a software infrastructure for the realisation of ngAHNs. This infrastructure executes the specifications of ngAHNs to inform members of their permissions, obligations, etc, at any point in time. Third, to simulate ngAHNs, we discuss a sample node architecture. Nodes based on this architecture consider the network's social laws in their decision-making, and can be individually configured to exhibit distinct behaviour. Finally, we summarise the presented work and outline our current research directions.

2 Norm-Governed Ad Hoc Networks

In previous work [1–3] we presented a theoretical framework for specifying ngMAS in terms of concepts stemming from the study of legal and social systems. The behaviour of the members of a ngMAS is regulated by social laws expressing their:

(i) physical capabilities (that is, what actions members can perform in their 'environment');
(ii) *institutional powers* [19, 38], a characteristic feature of norm-governed systems, whereby designated participants have the institutional power (or are empowered) by the system to create/modify facts of special significance within the system, *institutional facts*, usually by performing a specified kind of act (such as when an agent awards a contract and thereby creates a set of normative relations between the contracting parties);
(iii) permissions, prohibitions and obligations;
(iv) sanctions, that is strategies countering the performance of forbidden actions and non-compliance with obligations.

Note that there is no standard, fixed relationship between physical capability, institutional power and permission. For example, being empowered to perform an action A does not necessarily imply being permitted to perform A or being capable of A. (For further discussion and references to the literature see [19,25].) The laws comprising level (ii) of the specification correspond to the *constitutive norms* that define the meaning of the agents' actions. Levels (i) and (iii), respectively, can be seen as representing the *physical* and *normative* environment within which the agent interactions take place.

We have employed our framework for specifying a protocol used to regulate the control of access to shared resources [1], a typical issue in AHNs. The protocol expresses the conditions under which a node can be said to have the institutional power to request to access the limited resources. Exercising this power causes the node eligible to be granted access to these resources. Whether a node is *permitted* or not to exercise this power is another aspect expressed by the protocol specification. Typically, the performance of a forbidden action leads to a sanction (irrespective of whether the node performing the action was empowered to do so). Similarly, the protocol expresses, among other things, the circumstances in which it is meaningful to say that a resource controller is obliged or simply permitted to grant/revoke access to the resource, and what the consequences are, and the conditions under which it is physically (practically) possible or permitted for a node to manipulate a shared resource, or obligatory to release the resource, and what the consequences are.

While being a member of an AHN typically implies being within communication range with at least one of the remaining members, being a member of a ngAHN additionally implies being governed by the ngAHN's social laws (see Figure 1), that is, having a set of institutional powers, permissions, and obligations. Consider, in the resource sharing example, a node N that is not a member of the ngAHN but is within communication range of the ngAHN's members. N can

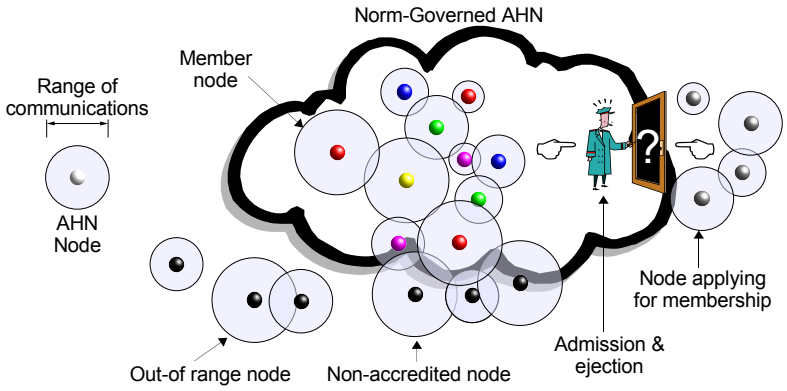

Fig. 1. An AHN Node (left) and a norm-governed AHN.

communicate a request for access to the shared resources, by means of sending a message of a particular form via a TCP/IP socket connection, for instance, but its request will be ignored, since N is not empowered to request the resources.

In order to become a member of a ngAHN, a node participates in a *role-assignment* protocol. If the node is accepted in the ngAHN then it will occupy a set of roles expressing its institutional powers, permissions and obligations while in that ngAHN. The decision-making procedure for awarding or denying a role is application-specific. Example procedures are *chair-designated* (an elected node assigns roles), *election* (the members of a role-assigning committee comprising existing member nodes or other elected nodes vote on role-assignment), *argumentation* (the members of a role-assigning committee debate on role-assignment) and *lottery scheduling* (role-assignment operates on a probabilistic basis). Formalisations of the first three types of procedure may be found in [1,2,32] respectively. In all cases, the decision-making procedure of the role-assigning authority is informed by, among other things, whether or not the applicant N satisfies the *role conditions* (for example, nodes acting as routers should have broad communication range), whether or not N has been banned from a ngAHN, or how many times it has been disqualified (banning and disqualification are discussed below).

As already mentioned, actuality does not necessarily coincide with ideality in ngAHNs, that is, a node may, inadvertently (due to network conditions or software errors) or maliciously, perform forbidden actions or not comply with its obligations. In this case sanctions may be enforced, not necessarily as a 'punishment', but as a way of adapting the network organisation. Sanctions may come in various forms; for example, a sanctioned node N may be:

- suspended, that is, N loses for a specified time period its institutional powers and permissions. In the resource sharing example, a suspended node loses access to the shared resources as its requests for access will not be serviced (since the node lost the institutional power to request the resources).

– disqualified; N loses its ngAHN membership and thus loses its institutional powers and permissions. However, N may re-apply to enter the ngAHN and, if successful, will regain its institutional powers and permissions.
– banned, that is, N loses its membership and may not re-apply to enter the ngAHN.

Other forms of sanction are possible, such as, for instance, 'bad reputation' (see [9, 16] for a few examples); the choice of a sanction type (when is a node penalised, what is the penalty that it has to face, who applies the penalty, etc) is application-specific.

Another possible enforcement strategy is to try to devise (additional) physical controls that will force nodes to comply with their obligations or prevent them from performing forbidden actions. When competing for hard disk space, for example, a forbidden revocation of the resource access may be physically blocked, in the sense that a node's account on the file server cannot be deleted. The general strategy of designing mechanisms to force compliance and eliminate non-permitted behaviour is what Jones and Sergot [18] referred to as *regimentation*. Regimentation devices have often been employed in order to eliminate 'undesirable' behaviour in computational systems (see, for example, *interagents* [35], *sentinels* [22], *controllers* [26], *guards* and *enforcers* [5]). It has been argued [18], however, that regimentation is rarely desirable (it results in a rigid system that may discourage agents from entering [34]), and not always practical. The practicality of regimentation devices is even more questionable when considering AHNs, due to the transient nature of these networks. In any case, violations may still occur even when regimenting a computational system (consider, for instance, a faulty regimentation device). For all of these reasons, we have to allow for sanctioning and not rely exclusively on regimentation mechanisms.

The process of joining or excluding a node from a ngAHN is a part of a *session control* protocol which further prescribes ways for inviting to join, or withdrawing from a ngAHN, changing the laws of a ngAHN, determining which resources are to be shared, and so on.

In order to realise ngAHNs, we need to provide mechanisms for informing member nodes of the social laws governing their behaviour — a discussion of such mechanisms is presented next.

3 Norm-Aware Nodes

We encode social laws specifications of norm-governed systems in executable action languages. We have shown how two such languages from the field of Artificial Intelligence (AI) may be used to express these specifications: the $\mathcal{C}+$ language [14, 15] and the Event Calculus (EC) [23]. The $\mathcal{C}+$ language, notably when used with its associated software implementation, the Causal Calculator (CCALC), already supports a wide range of computational tasks of the kind that we wish to perform on system specifications. A major attraction of $\mathcal{C}+$ compared with other action languages in AI is its explicit semantics in terms of

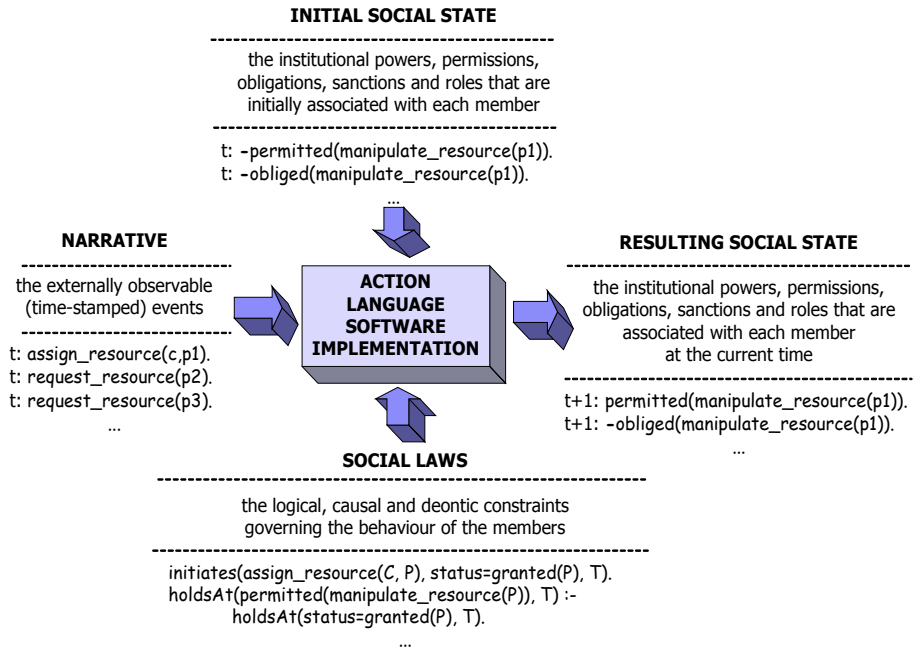

Fig. 2. Computing with Social Laws.

labelled transition systems, a familiar structure widely used in logic and computer science. EC, on the other hand, does not have an *explicit* transition system semantics, but has the merits of being simple, flexible, and very easily and efficiently implemented for an important class of computational tasks. It thus provides a practical means of implementing an executable system specification. The Society Visualiser (SV) [3] is a logic programming implementation that supports computational tasks on system specifications formulated in EC. A detailed discussion of both action languages and their software implementations can be found in [3].

Members of a ngAHN should be aware at any point in time of their institutional powers, permissions, obligations and sanctions. Such information may be produced by computing answers to 'prediction' queries on the social laws specification. This type of computational task, which is supported by each action language software implementation (ALSI), that is, CCALC and SV, may be expressed, in the context of ngAHNs, as follows. The input to an ALSI includes an initial *social state* — that is, a description of the institutional powers, permissions, obligations and sanctions that are initially associated with the ngAHN members, and a *narrative*, that is, a description of temporally-sorted externally observable events (actions) of the ngAHN. The outcome of a prediction query (if any) is the current social state — that is, the members' institutional powers,

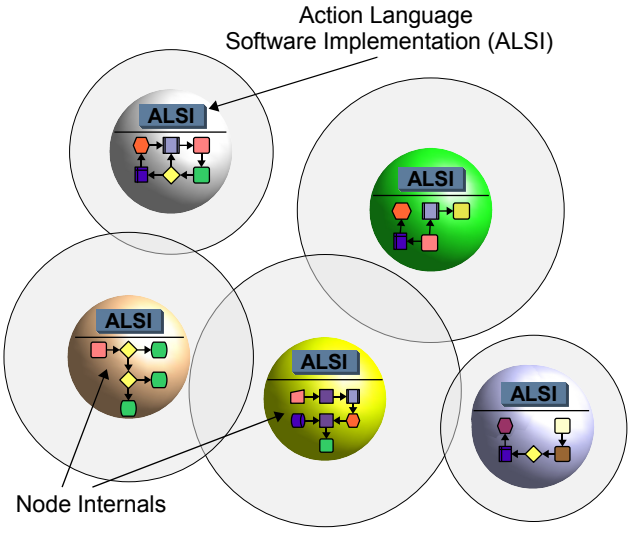

Fig. 3. Run-Time Mechanisms: Total Distribution.

permissions, obligations and sanctions that result from the events described in the narrative.

Figure 2 illustrates the computation of answer to a prediction query. Example narrative, social laws, initial and resulting social states, expressed in EC pseudo-code, concerning the resource sharing example, are presented.

The current social state may be available to (a subset of) the ngAHN members at run-time. Such run-time services may be provided by a central server or, as expected in an AHN, in various distributed configurations. We describe two example configurations below.

Each ngAHN member node (with a private internal architecture) could be equipped with an ALSI module, computing answers to prediction queries for the benefit of the node, informing it of its institutional powers, permissions, obligations and sanctions (see Figure 3). In this configuration the ALSI module of each node may not be aware of all events (that is, the narrative) necessary to compute a prediction query answer. For instance, a node may not be aware of the fact that a resource is no longer available and thus, its ALSI module may compute that the node is still empowered, say, to request access to the resource. A partial narrative could be enriched by 'witnessing' other nodes' actions or requesting from peer nodes to be updated about the events taking place in the ngAHN. In the latter case, however, the node could be, intentionally or not, misinformed which could result in an inaccurate computation of its institutional powers, permissions, and so on.

A design choice that needs to be made in this setting concerns the social laws available to the ALSI module of a node. Clearly a node N's ALSI module

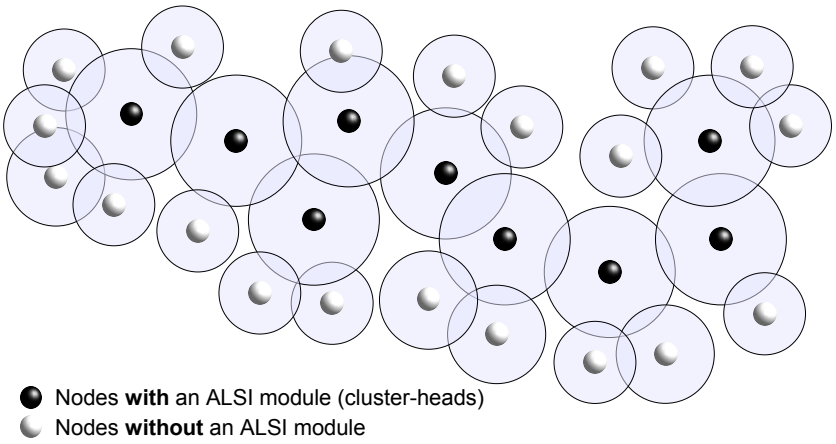

Fig. 4. Run-Time Mechanisms: Partial Distribution.

should include the laws relevant to the roles N occupies. In some applications, the permissions and obligations of a node need to be private to that node — in this case the ALSI module of each node should contain *only* the laws relevant to its roles. In other applications, a node's ALSI module could contain the complete set of social laws, thus enabling each node to compute information about any other member (provided that a node is aware of the complete ngAHN narrative).

Due to the limited resources (battery, for example) available to the nodes of a ngAHN, it may not be practical or feasible for each node to compute its own institutional powers, permissions and obligations. To address this issue, nodes with rich resources availability could be selected specifically for producing such information and publicising it to the members of a ngAHN. Consider the example topology shown in Figure 4. A network is divided into clusters and each cluster elects a 'cluster-head' (CH), typically a node with longer communication range, larger bandwidth and more power (again, such as battery). A CH's communication range covers all the nodes in the cluster. Links are established to connect CHs to a backbone network. (Such topologies have been proposed in the literature for achieving 'good' routing performance in AHNs — see, for example, [10,33,44].) Every message exchange, in this example topology, is carried out via the backbone network; therefore, CHs can compile, in cooperation, the narrative of events of the whole ngAHN. Moreover, each CH is equipped with an ALSI module and the complete set of the ngAHN's social laws. Consequently, a CH is capable of computing a node's institutional powers, permissions and obligations, and detecting non-compliance with obligations and performance of forbidden actions. Such information is publicised to a node upon request (a node requests such information from its CH). Different strategies may be followed for publicising information to nodes — a node's institutional powers could be pub-

licised to all members of a ngAHN, its permissions could be kept private to the node, etc.

It is possible that a CH will not behave as it ought to behave. For instance, it may move out of communication range or run short of resources, and thus be incapable of fullfiling the nodes' requests. Moreover, a CH may intentionally mis-behave — for example, giving incorrect information (informing a node that it is forbidden to perform an action when it is permitted to do so) or disclosing private information (publicising the obligations of a node when these should be private to the node). For these reasons, we may express a CH as role of a ngAHN, specifying the institutional powers, permissions and obligations associated with that role. Violation of these permissions or non-compliance with these obligations will result in 'sanctioning' a node acting as a CH. (Note that the detection of a CH's mis-behaviour may not be straightforward — consider, for example, the disclosure of private information.) A sanctioned CH is replaced by a new one. (Recall that a sanction is not necessarily a penalty; in the case of inadvertent mis-behaviour a sanction can be seen as a way of dealing with the network's exceptions.) The selection of a node as a CH, for the replacement of a 'sanctioned' CH or for the formation of a new cluster, is a typical issue of role-assignment (see Section 2).

Clearly, other topologies are possible for the realisation of a ngAHN. Grizard et al. [16], for instance, discuss an overlay network in which nodes in the overlay, called 'controller agents', monitor the behaviour of the underlying network nodes, called 'application agents', detect whether an application agent violated a norm, and publicise, upon request, details about norm violation (for example, how often each application agent violates a norm). In general, the aforementioned topologies were presented in order to give an indication of the design choices that need to be made for the realisation of a ngAHN.

Apart from mechanisms for informing nodes of their institutional powers, permissions and obligations, the realisation of ngAHNs further requires that nodes actually *consider* such information in their decision-making process. The next section discusses node architectures for ngAHNs.

4 Decision-Making in Norm-Aware Nodes

The functionality of a node within a ngAHN can be likened to that of an agent within a multi-agent system. We have been developing an architecture for a ngAHN node (Figure 5), drawing upon design principles for *agent architectures* [20, 21]. We give an overview of the architecture and focus, in this paper, on the architecture module defining how a node factors a ngAHN's social laws into its decision-making.

The architecture is conceptually organised into three main parts: a *control module*, *state & attributes* and an ALSI module. The control module includes a *communications* interface enabling each node to send and receive its own messages. (We assume error-free communications within simulations, although we note that the introduction of deliberate communications errors and correspond-

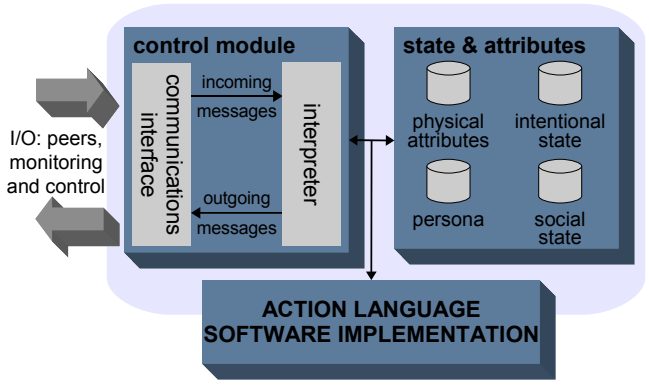

Fig. 5. A Sample Architecture for a Norm-Governed AHN Node.

ing error-handling mechanisms is a possible avenue of future investigation.) The communications interface can also be used by an operator or manager to externally monitor and instruct the node. That is, the same type of message exchange that occurs between nodes can also occur between a node and an 'external' entity, with emphasis on interrogative and imperative communications. In the simulation platform described later (Section 4.1), we show that in addition to supporting such input/output operations, the same interface can be used to interact with an 'environment'. In the case of a (ng)AHN simulation, the environment process represents the network topology. Communicative acts directed to this process are interpreted as queries or physical acts upon the environment, which can respond accordingly. In this way, the environment process may determine what the nodes perceive of their physical environment. The environment process can also be used to effect exogenous events (such as *time-outs*).

The *interpreter*, a part of the control module, is a focal point of the architecture, linking to all other components. This module is primarily characterised by a processing cycle that enacts, in turn, the perceptive, deliberative and responsive behaviour of the node. These amount to checking for incoming messages, consulting and updating state & attributes accordingly, formulating appropriate (re-)actions and creating optional outgoing messages.

The state & attributes grouping includes elements usually found in deliberative agent architectures. *Physical attributes* include information like a node identifier, location within the environment as well as platform and communications attributes such as battery power and communications range. Note that some of these attributes are beliefs rather than 'facts' as they may be based on (possibly outdated) information obtained from an environment process. The *intentional state* comprises current goals and which, of the methods available, the node in question has selected to achieve them.

While analogues of the preceding architectural elements can be readily found in most deliberative agent architectures, *persona* and *social state* are, perhaps, less likely to have direct counterparts. The former determines a node's social

outlook by defining how exactly social laws are factored into its behaviour. The latter is as explained previously in Section 3, with a subjective emphasis — that is, a record of the node's institutional powers, permissions, obligations and sanctions (as opposed to a record of every member's institutional powers, permissions, etc). Such information is produced by consulting an ALSI module, local or remote (for instance, consulting a cluster-head).

We believe that the design approach described above facilitates our modelling and study of ngAHNs, as it allows distinct node behaviours to be straightforwardly introduced through well-defined adjustment of a single module within a generic architecture (the persona module, for example). This has the outward appearance of introducing a different node architecture without the potential configuration and interoperability issues associated with doing so. In addition, the communications interface 'hides' the internal operation of one node from another, thus capturing the node heterogeneity property of a ngAHN.

4.1 Implementation

We have been developing a simulation base that allows us to model the entities and communications that might take place in a ngAHN [20, 21]. Implemented in *Prolog*, the simulation base enacts the processing cycles of multiple communicating nodes. To do so, it first consults the individual files in which the behaviours of these nodes are specified. These files define the initial configuration of state & attributes elements.

Following initialisation, the simulation base performs the processing cycles of the defined nodes consecutively. The environment cycle is always first to be enacted, while those of the remaining nodes occur in a randomised order. This avoids the development of unfavourable or predictable interaction patterns. Message exchange during processing cycles occurs when data structures, representing node-designated message queues, are consulted and updated.

A processing cycle consists of message exchange as described above, followed by update of the relevant node's state & attributes. During the update, decisions are taken on how to respond to received messages and perceived events — effectively, the deliberative stage of the cycle. This deliberation may produce specific changes to the state & attributes during the update. The realisation that a certain action or event has transpired, or that a particular condition holds, for example, can trigger the adoption of a new goal, attitude or of different means to achieve existing goals.

4.2 Resource-Sharing Protocol

In the following scenario, we provide Prolog-based pseudo-code further illustrating the implementation aspects described above. More precisely, we discuss ways of simulating a ngAHN by specifying different node personas. The scenario is based on an instance of the resource sharing protocol mentioned earlier. In this instance `node1` acts as the resource controller or 'chair', and `node2` and `node3` are the 'subjects', that is, they request access to the shared resource.

An ALSI module expresses, in this example, a logic programming Event Calculus (EC) encoding of the social laws of the resource-sharing scenario. This is used in the first instance to establish the roles occupied by nodes. To summarise the protocol, the chair determines who is the best candidate among the current resource requests and grants resource access accordingly. A grant is for a set amount of time, during which the allocated node derives some utility from the resource and may, at its discretion, either release the resource early or request an extension of the grant period. In addition, the chair can entertain additional resource requests from the other node during the grant period and may insist that (command) the node to whom the resource is currently granted release it immediately. When a violation of social laws occurs (and is detected), a node may be sanctioned. Further consideration of these aspects of the protocol (and variants) is given in [1].

In this example, we characterise `node2` as being willing to violate a social law if the utility of doing so outweighs the expected sanction. In contrast, `node3` will only contemplate social law violation if it has not already done so within an arbitrarily recent time-frame. We consider one potential violation where a node does not comply with the obligation to release a resource. Consider the following example:

```
(1)  process_message( node2, cmd_release_resource( node1, node2 ), T ) :-
(2)     holdsAt( obliged( node2, release_resource( node2 ) ) = true, T ),
(3)     holdsAt( status = granted(node2, TEnd), T ),
(4)     TLeft = TEnd - T,
(5)     TLeft > 0,
(6)     utility( TLeft, U ),
(7)     holdsAt( sanction( obligation_release_resource ) = Cost , T ),
(8)     (  compare(U, Cost) ->
(9)        send_msg( release_resource( node2 ) )
(10)       ;
(11)       true
(12)    ).
```

The `process_message/3` procedure presented above expresses the reaction of a node (in this example, `node2`) to an incoming `cmd_release_resource` ('command to release the resource') message — this message was sent at time `T` by `node1`, in this example. Upon receipt of a `cmd_release_resource` message, `node2` calculates whether or not it is obliged to release the resource (line 2 of the pseudo-code example). A message `cmd_release_resource` creates an obligation for the node holding the resource to release it only when the controller has the institutional power to issue such a command. The controller may not always have the power to command the release of a resource — for example, before the time allocated to the holder ends, the controller may only be empowered to command a release of the resource when it receives an 'urgent' request for the resource from another node. If `node2` is indeed obliged to release the resource, then it recalls the time (`TEnd`) until which access to the resource was originally granted (line 3), and calculates the time difference (`TLeft`) between `TEnd` and the current time

T (line 4). If TLeft is greater than zero, that is, the chair commanded node2 to release the resource before the allocated time ended, then node2 computes a subjective utility based on TLeft (line 6), that is, it computes the utility it would derive from manipulating the resource until the allocated time ends. It then (line 8) compares the calculated utility with the objective sanction associated with failure to comply with the obligation to release the resource (that is, the sanction expressed by the social laws of the resource sharing protocol). If the comparison favours keeping hold of the resource (in this case the compare/2 predicate fails), node2 will ignore the chair's command.

Note, in the above example, that if TLeft is less than or equal to zero then cmd_release_resource was issued after the time allocated for manipulating the resource ended — this case is dealt by another process_message/3 procedure.

While we do not provide a specification of the compare/2 predicate here, we note that it represents a key aspect of a node's (social) decision-making. The predicate provides a subjective assessment of the seemingly objective sanction cost relative to a node's internal utility function. It naturally follows that the value of either complying with or violating a social law is primarily determined by a node's private architecture. A welcome corollary is the ability to nuance node behaviour through the specification of different compare/2 predicates. A less welcome concern is the additional complexity that such specifications require.

holdsAt/2 is an EC predicate expressing the social laws of the resource sharing protocol. In other words, the occurrences of holdsAt/2 that appear in the code above (and below) represent calls to an ALSI module capable of reasoning about social laws. As already mentioned, an invocation of an ALSI module may be 'actual' (in the case where the node in question has its own module) or 'logical' (when the ALSI module is located in another node).

Consider another example node persona:

```
(1)  process_message( node3, cmd_release_resource( node1, node3 ), T ) :-
(2)    holdsAt( obliged( node3, release_resource( node3 ) ) = true, T ),
(3)    last_violation( obligation_release_resource, T2 ),
(4)    compliance_interval( CInt ),
(5)    (  ( CInt < ( T - T2 ) ) ->
(6)       (  retract( last_violation( obligation_release_resource, _ ) ),
(7)          asserta( last_violation( obligation_release_resource, T ) ),
(8)       )
(9)       ;
(10)      send_msg( release_resource( node3 ) )
(11)   ).
```

node3 maintains a record of the last time (T2) it violated the obligation to release the resource. It then compares the time difference T - T2 (T is the current time) with a 'compliance interval'. If the time elapsed since the last such transgression is greater than the compliance interval, node3 will not comply with the obligation to release the resource.

Clearly, there are other possible attitudes concerning compliance with obligations and performance of forbidden actions. Moreover, we may specify different

node personas by expressing the attitude of a node concerning its institutional powers. For instance, a node will not perform an action, say request a resource, if it is not empowered to do so (even if it is permitted to request the resource). Another node N may perform an action, although not empowered to do so, expecting that (some) peer nodes will not be able to tell whether or not N was empowered to perform the action (some peer nodes may not have access to an ALSI module).

By specifying different attitudes to institutional power, permission and obligation — that is, by adjusting the persona module of the proposed node architecture, we may introduce different node behaviours as required for our ngAHN simulations. (Varying node behaviours can be introduced by additionally adjusting the intentional state module of the node architecture.) This is our current research direction.

5 Related work

There are several approaches in the literature that are related to our work on specifying norm-governed multi-agent systems — [4,5,11–13,26–29,35,36,39–41, 45,46] are but a few examples. Generally, work on the specification of multi-agent systems does not distinguish between the constitutive laws and the normative environment within which the agent interactions take place. This is a key difference between our work and related approaches in the literature. Our specification of social laws explicitly represents the institutional powers of the agents, capturing the meaning of the agents' actions, and thus expressing the constitutive laws of a system. Moreover, our specification differentiates between institutional power, permission and physical capability, thus separating the constitutive laws from the normative and the physical environment within which a system is executed. A detailed discussion of research related to the framework for specifying norm-governed computational systems, and an evaluation of this framework, can be found in [2, 3].

Within multi-agent systems research, 'social awareness' can be seen as a development of cooperative behaviour (see, for example [17, 43]), which has been primarily based on the use of communication protocols. Castelfranchi *et al.* [8] note that agents using only fixed protocols are unable to deal with unexpected behaviour on the part of their environment and other agents. The authors consequently identify the need for *autonomous normative agents*: agents with an awareness of social laws and the capacity to both adopt and violate such laws. Castelfranchi *et al.* position social law-awareness as an influential, but not deterministic, factor in agent behaviour — to them, an agent's ability to choose to violate a recognised social law is equally important as its ability to choose to conform to the same. Our node architecture reflects this consideration by positioning the ALSI as an *advisory* module to whose 'output' the agent can react arbitrarily.

Several researchers have since proposed agent models and associated theory to address the needs identified by Castelfranchi *et al.* For example, the Beliefs-

Obligations-Intentions-Desires (BOID) architecture of Broersen *et al.* [6,7] introduces mechanisms to resolve conflicts between the representations of cognitive state in a deliberative agent and its obligations. This allows the characterisation of a number of abstract agent social perspectives in terms of the interdependencies, precedences and update strategies of beliefs, obligations, intentions and desires, in which obligations and desires are respectively treated as external (social) and internal motivational attitudes. We adopt a similar first-class view of normative relations in our node architecture from a *logical* perspective but note that the BOID approach differs slightly in its integration of these relations at an *implementation* level. It is through the introduction of a specialised implementation module (the ALSI) to our node architecture that an agent is able to perceive its normative relations.

Lopez *et al.* [24] characterise social laws in terms of their 'positive and negative effect' on the goals of agents and those of their 'society'. The approach of Lopez *et al.* is currently a formal model of agent behaviour and remains unimplemented.

Sadri *et al.* [37] incorporate social decision-making within a deliberative agent model enhanced by EC-based formulations of social laws, noting how these can change over a system's lifetime. They use an abductive logic-based proof procedure to compute an agent's time-constrained *social goals* — expressions encoding what the system expects of an agent. Distinct logic programs encode the *preference policy* of each agent — for example whether an agent puts social goals above its own goals. Social perspectives are identified (for instance, social, anti-social agents) based on the agents' preferences (personal goals versus social goals). Like the other reviewed approaches, the framework of Sadri *et al.* does not identify the institutional powers of agents, a concept on which we place emphasis.

6 Summary

Malfunctioning, either by intent or by circumstance, is to be expected in AHNs. How to identify and adapt to such situations is essential. By specifying the permissions, obligations, and other more complex normative relations that may exist between the members of an AHN, one may precisely identify 'undesirable' behaviour, such as the performance of forbidden actions (for instance, illicit use of a peer's processor cycles) and non-compliance with obligations (for example, denying access to one's processor cycles). Therefore, it is possible to introduce sanctions and enforcement strategies to adapt to such behaviour (for example, temporarily ejecting a mis-behaving or erroneous node from a network).

We have developed a framework for specifying the laws of norm-governed AHNs and executing these laws for informing the decision-making of the member nodes. We are currently developing such nodes in order to simulate norm-governed AHNs and thus investigate the practicality of, and in general evaluate, the presented approach.

Acknowledgements

This work has been supported by the EPSRC project "Theory and Technology of Norm-Governed Self-Organising Networks" (GR/S74911/01).

References

1. A. Artikis, L. Kamara, J. Pitt, and M. Sergot. A protocol for resource sharing in norm-governed ad hoc networks. In *Proceedings of Workshop on Declarative Agent Languages and Technologies (DALT)*, volume LNCS 3476, pages 221–238. Springer, 2005.
2. A. Artikis, M. Sergot, and J. Pitt. An executable specification of an argumentation protocol. In *Proceedings of Conference on Artificial Intelligence and Law (ICAIL)*, pages 1–11. ACM Press, 2003.
3. A. Artikis, M. Sergot, and J. Pitt. Specifying norm-governed computational societies. Technical Report 2006/5, Imperial College London, Department of Computing, 2006. Retrieved March 6, 2006, from `http://www.doc.ic.ac.uk/research/technicalreports/2006/DTR06-5.pdf`.
4. A. Bandara. *A Formal Approach to Analysis and Refinement of Policies*. PhD thesis, Imperial College London, 2005.
5. J. Bradshaw, A. Uszok, R. Jeffers, N. Suri, P. Hayes, M. Burstein, A. Acquisti, B. Benyo, M. Breedy, M. Carvalho, D. Diller, M. Johnson, S. Kulkarni, J. Lott, M. Sierhuis, and R. Van Hoof. Representation and reasoning about DAML-based policy and domain services in KAoS. In J. Rosenschein, T. Sandholm, M. Wooldridge, and M. Yokoo, editors, *Proceedings of Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pages 835–842. ACM Press, 2003.
6. J. Broersen, M. Dastani, J. Hulstijn, and L. Van der Torre. Goal generation in the BOID architecture. *Cognitive Science Quarterly*, 2(3–4):428–447, 2002.
7. J. Broersen, M. Dastani, J. Hulstijn, Z. Huang, and L. van der Torre. The BOID architecture: conflicts between beliefs, obligations, intentions and desires. In *Proceedings of Conference on Autonomous Agents*, pages 9–16. ACM Press, 2001.
8. C. Castelfranchi, F. Dignum, C. Jonker, and J. Treur. Deliberative normative agents: Principles and architecture. In *Proceedings of Workshop on Agent Theories, Architectures and Languages*, volume LNCS 1757, pages 364–378. Springer, 1999.
9. C. Dellarocas. Reputation mechanisms. In T. Hendershott, editor, *Handbook on Economics and Information Systems*. Elsevier Publishing, to appear.
10. X. Du. QoS routing based on multi-class nodes for mobile ad hoc networks. *Ad Hoc Networks*, 2(3):241–254, 2004.
11. M. Esteva, D. de la Cruz, and C. Sierra. ISLANDER: an electronic institutions editor. In C. Castelfranchi and L. Johnson, editors, *Proceedings of Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1045–1052. ACM Press, 2002.
12. M. Esteva, J. Padget, and C. Sierra. Formalizing a language for institutions and norms. In J.-J. Meyer and M. Tambe, editors, *Intelligent Agents VIII: Agent Theories, Architectures, and Languages*, LNAI 2333, pages 348–366. Springer, 2002.
13. D. Fitoussi and M. Tennenholtz. Choosing social laws for multi-agent systems: minimality and simplicity. *Artificial Intelligence*, 119(1-2):61–101, 2000.
14. E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153(1–2):49–104, 2004.

15. E. Giunchiglia, J. Lee, V. Lifschitz, and H. Turner. Causal laws and multi-valued fluents. In *Proceedings of Workshop on Nonmonotonic Reasoning, Action and Change (NRAC)*, 2001.
16. A. Grizard, L. Vercouter, T. Stratulat, and G. Muller. A peer-to-peer normative system to achieve social order. In *Proceedings of Workshop on Coordination, Organization, Institutions and Norms in Agent Systems*, 2006.
17. A. Haddadi. *Communication and Cooperation in Agent Systems — A Pragmatic Theory*. LNCS 1056. Springer, 1995.
18. A. Jones and M. Sergot. On the characterisation of law and computer systems: the normative systems perspective. In *Deontic Logic in Computer Science: Normative System Specification*, pages 275–307. J. Wiley and Sons, 1993.
19. A. Jones and M. Sergot. A formal characterisation of institutionalised power. *Journal of the IGPL*, 4(3):429–445, 1996.
20. L. Kamara, J. Pitt, and M. Sergot. Norm-aware agents for ad hoc networks: A position paper. In *Proceedings of the Ubiquitous Agents Workshop, AAMAS*, 2004.
21. L. Kamara, J. Pitt, and M. Sergot. Towards norm-governed self-organising networks. In *Proceedings of the NorMAS Symposium, AISB*, 2005.
22. M. Klein, J. Rodriguez-Aguilar, and C. Dellarocas. Using domain-independent exception handling services to enable robust open multi-agent systems: the case of agent death. *Journal of Autonomous Agents and Multi-Agent Systems*, 7(1–2):179–189, 2003.
23. R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–96, 1986.
24. F. Lopez, M. Luck, and M. d'Inverno. Normative agent reasoning in dynamic societies. In N. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *Proceedings of Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 732–739. IEEE Computer Society, 2004.
25. D. Makinson. On the formal representation of rights relations. *Journal of Philosophical Logic*, 15:403–425, 1986.
26. N. Minsky and V. Ungureanu. Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 9(3):273–305, 2000.
27. L. Moreau, J. Bradshaw, M. Breedy, L. Bunch, P. Hayes, M. Johnson, S. Kulkarni, J. Lott, N. Suri, and A. Uszok. Behavioural specification of grid services with the KAoS policy language. *Cluster Computing and the Grid*, 2:816–823, 2005.
28. Y. Moses and M. Tennenholtz. On computational aspects of artificial social systems. In *Proceedings of Workshop on Distributed Artificial Intelligence (DAI)*, pages 267–284, 1992.
29. Y. Moses and M. Tennenholtz. Artificial social systems. *Computers and Artificial Intelligence*, 14(6):533–562, 1995.
30. A. Murphy, G.-C. Roman, and G. Varghese. An exercise in formal reasoning about mobile communications. In *Proceedings of Workshop on Software Specification and Design*, pages 25–33. IEEE Computer Society, 1998.
31. C. Perkins. *Ad Hoc Networking*, chapter 1. Addison Wesley Professional, 2001.
32. J. Pitt, L. Kamara, M. Sergot, and A. Artikis. Voting in multi-agent systems. *Computer Journal*, 49(2):156–170, 2006.
33. J. Pitt, P. Venkataram, and A. Mamdani. QoS management in MANETs using norm-governed agent societies. In *Proceedings of Workshop on Engineering Societies in the Agents' World*, LNCS 3963, pages 221–240. Springer, 2006.
34. H. Prakken. Formalising Robert's rules of order. Technical Report 12, GMD – German National Research Center for Information Technology, 1998.

35. J. Rodriguez-Aguilar, F. Martin, P. Noriega, P. Garcia, and C. Sierra. Towards a test-bed for trading agents in electronic auction markets. *AI Communications*, 11(1):5–19, 1998.
36. J. Rodriguez-Aguilar and C. Sierra. Enabling open agent institutions. In K. Dautenhahn, A. Bond, L. Canamero, and B. Edmonds, editors, *Socially Intelligent Agents: Creating relationships with computers and robots*, pages 259–266. Kluwer Academic Publishers, 2002.
37. F. Sadri, K. Stathis, and F. Toni. Normative KGP agents. *Journal of Computational and Mathematical Organization Theory*, 12(2-3):101–126, 2006.
38. J. Searle. *Speech Acts*. Cambridge University Press, 1969.
39. Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In W. Swartout, editor, *Proceedings of Conference on Artificial Intelligence (AAAI)*, pages 276–281. The AAAI Press/ The MIT Press, 1992.
40. Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, 73(1-2):231–252, 1995.
41. M. Tennenholtz. On computational social laws for dynamic non-homogeneous social structures. *Journal of Experimental and Theoretical Artificial Intelligence*, 7:379–390, 1995.
42. Y. Wang, V. Giruka, and M. Singhal. A fair distribution solution for selfish nodes problem in wireless ad hoc networks. In *Proceedings of Conference on Ad Hoc, Mobile and Wireless Networks*, LNCS 3158, pages 211–224. Springer, 2004.
43. M. Wooldridge and N. Jennings. Formalizing the cooperative problem solving process. In *Readings in Agents*, pages 430–440. Morgan Kaufmann Publishers Inc., 1998.
44. K. Xu, X. Hong, and M. Gerla. Landmark routing in ad hoc networks with mobile backbones. *Journal of Parallel Distributed Computing*, 63(2):110–122, 2003.
45. P. Yolum and M. Singh. Reasoning about commitments in the event calculus: An approach for specifying and executing protocols. *Annals of Mathematics and Artificial Intelligence*, 42(1–3):227–253, 2004.
46. F. Zambonelli, N. Jennings, and M. Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 12(3):317–370, 2003.