2017 IEEE 37th International Conference on Distributed Computing Systems

The Internet of Things and Multiagent Systems: Decentralized Intelligence in Distributed Computing

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695, USA
singh@ncsu.edu

Amit K. Chopra
School of Computing and Communications
Lancaster University
Lancaster LA1 4WA, UK
amit.chopra@lancaster.ac.uk

***Abstract*—Traditionally, distributed computing concentrates on computation understood at the level of information exchange and sets aside human and organizational concerns as largely to be handled in an ad hoc manner. Increasingly, however, distributed applications involve multiple loci of autonomy. Research in multiagent systems (MAS) addresses autonomy by drawing on concepts and techniques from artificial intelligence. However, MAS research generally lacks an adequate understanding of modern distributed computing.**

In this Blue Sky paper, we envision *decentralized multiagent systems* as a way to place decentralized intelligence in distributed computing, specifically, by supporting computation at the level of social meanings. We motivate our proposals for research in the context of the Internet of Things (IoT), which has become a major thrust in distributed computing. From the IoT's representative applications, we abstract out the major challenges of relevance to decentralized intelligence. These include the heterogeneity of IoT components; asynchronous and delay-tolerant communication and decoupled enactment; and multiple stakeholders with subtle requirements for governance, incorporating resource usage, cooperation, and privacy. The IoT yields high-impact problems that require solutions that go beyond traditional ways of thinking.

We conclude with highlights of some possible research directions in decentralized MAS, including programming models; interaction-oriented software engineering; and what we term enlightened governance.

***Index Terms*—Governance; Multiagent systems; Decentralization; Sociotechnical systems; Norms**

I. Introduction

This is the age of distributed computing. Distributed computing has progressed dramatically since its inception and research has been booming of late. However, despite shifts in settings where distributed computing is applied—for example, a greater number of powerful edge devices and greater expectations for user control and privacy—the research area has not changed quite as much as it could have and should have to match these modern settings.

The thesis of this Blue Sky paper is that if we are to take Distributed Computing as a research community to the next level, we need to introduce new ways of thinking that better appreciate the new problems and opportunities that are at the heart of any research discipline. Specifically, we need to expand the scope of computation as understood within Distributed Computing to introduce human and organizational aspects such as of autonomy and accountability. This is not to step on the turf of humans-oriented academic disciplines but to develop a *computational* understanding of these high-level aspects that only Distributed Computing can offer, and to incorporate that understanding into our models and methods.

In particular, the arrival of the Internet of Things (IoT) forces problems upon us that require consideration of new abstractions and techniques for distributed computing. The main idea of the IoT, in simple terms, is to introduce *things* to the Internet—so that they can be monitored and controlled and information produced by them can be used by stakeholders for better decision-making. Currently, to design an IoT-enabled application (*Iota*, from here on, for brevity) is to figure out (1) how to gather the requisite information, (2) how to process it, and (3) how to take the appropriate action. Such a conception though misses the crucial fact that the typical Iota supports nothing but the enactment of a *social process* involving multiple autonomous parties. For example, the value of a health Iota does not lie in merely monitoring and recording various health variables but in enabling healthcare providers, patients, family members, and fitness instructors to take better-informed healthcare decisions and thereby provide better-informed services to each other. Notably, *information governance*, which has to with privacy and security in settings where stakeholders share resources, requires a computational conception of Iotas as social processes. In the absence of such a conception, these concerns can be addressed only in ad hoc ways, typically through manual negotiation and control by the parties involved. Providing a computational foundation for governance is crucial to the expansion of Iotas.

Some of the enabling abstractions and techniques for a social process-oriented conception of an Iota relate to multiagent systems (MAS), especially artificial intelligence. The idea of a MAS as a system of interacting agents, each of whom represents an autonomous party is a powerful one. So is the idea of high-level abstractions for capturing social relationships among agents. However, traditional MAS research is framed in ways that are not compatible with modern distributed computing. Some works extend single-agent abstractions such as beliefs, intentions, and goals to their multiagent (*joint*) counterparts; however, in doing so, they effectively assume a unitary perspective (usually undergirded by synchronous communication). Some works assume a global repository of

1063-6927/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDCS.2017.304

system state, making the system effectively centralized. In general, challenges related to asynchronous communication and decentralized enactments have been deemphasized in multiagent systems research. The gaps on both sides—distributed computing and multiagent systems—lead us to propose the notion of a *decentralized multiagent system* as a way to frame the new problems and their solutions.

More broadly, the objective of this paper is to motivate a conception of Iotas as social processes and to highlight the kinds of methodologies, including languages and infrastructures, required to realize to Iotas. Specifically, we propose that Iotas be realized as decentralized MAS constructed of heterogeneous and autonomous entities, namely, *agents*, that represent the autonomous participants. A computational model of social processes will enable us, as a research community, to systematically approach practical challenges that the IoT poses, such as coordination among devices and across organizations, and attendant concerns such as security and privacy, and to identify new research advances that will help address these challenges better than traditional computer science can.

Although the IoT raises several challenges [1], we concentrate in this paper on a few challenges that must be addressed in order to incorporate decentralized intelligence into the very heart of distributed computing. The contribution of this paper is the motivation of these challenges, an introduction of a new framework for thinking, and a discussion of promising approaches based on decentralized intelligence. The paper ends with a series of high-level research questions through which we hope to influence future research in the community.

II. IoT Primer

We briefly discuss the salient aspects of the IoT relevant for our present purposes.

A. Enabling Technologies

The IoT is enabled by a confluence of advances, primarily in low-level technologies. Sensors of a rich variety now exist and more are emerging. These sensors are able to obtain readings not only of obvious attributes such as temperature, air pressure, and location, but also detect specific chemicals (such as ozone), vibrations, salinity, and so on. Though sensors are inherently limited in the bandwidth that they support for sending data out and receiving commands, their capacities have been increasing. There have been major advances in low power and passive technologies. Passive technologies, such as RFID, rely upon an outside power source. It is not uncommon for active sensors to last years on a single battery. And, there are new technologies for power harvesting that can potentially last forever. For example, it is possible to extract energy from the temperature differential between the human skin and the ambient air temperature.

Although the IoT is a modern phenomenon, the component technologies that make it interesting have been developing for years. First, the IoT shows a natural synergy with mobile computing. Many low-power techniques were invented for mobile computing. In a typical usage setting, IoT nodes work in a low power mode and (high-power) mobile computing techniques are used to receive information from and command IoT nodes. That is, a conventional mobile computing node serves as a gateway to IoT devices. A particular instantiation of this pattern is where the IoT devices are wearables, embedded in the user's clothes or jewelry, and form a personal area network with the user's mobile phone, which stores and forward data and commands from and to the IoT devices. Likewise, research into ad hoc and fixed sensor networks, such as for sensing in a large area, has fed into the IoT. In the same spirit, typically in built environments, pervasive and ubiquitous computing provides a foundation for some aspects of the IoT.

B. Representative Applications

What makes the IoT interesting is not its technology, but the applications that it potentially sustains. These applications range over the domains of manufacturing, civil infrastructure, power systems, healthcare, and personal uses. Practically anything could be a thing—industrial equipment, lakes, glaciers, automobiles, roads, buildings, household appliances, personal artifacts, trees, even animals and humans. A typical approach is to place a networked device (sensor or actuator) on a thing so that it can be accessed online. The IoT enables precise monitoring of assets and other resources to help provide a basis for evidence-driven decision making. In healthcare, uses of the IoT include both the monitoring of infrastructure, such as operating theaters and equipment, and of patients.

Personal uses include IoT nodes in the environment and on a user's person, such as through wearables or medical implants. These applications involve providing an enhanced user experience based on inferring attributes of the user's current context. For example, knowing if the user is in a public place and in an environment with elevated ozone (as in smog) may suggest a medical intervention to avert an asthmatic attack (e.g., by advising the user to inhale some medication).

Another family of applications is of *participatory sensing* in which users share information that their sensors obtain [2]. Participatory sensing, which originated with users sharing sensors on their phones, is proving effective in monitoring our urban and natural environments. It can expand naturally to sharing information from sensors on things at large.

Moreover, the IoT gains value through emerging user interaction technologies. Specifically, augmented reality can make it possible for users to "visually browse" the world of things through augmentations of reality that are produced based on sensor data obtained from IoT nodes. For example, in monitoring a factory, an augmented reality interface may show the temperatures of different objects or when they were last repaired or how cracked they are. Such interfaces can help provide users with actionable intelligence without overwhelming them with raw data.

C. Architecture

Figure 1 illustrates some of the important architectural concerns dealing with IoT. This figure highlights the main components that undergird an IoT application. On one side, we place IoT devices, both sensors and actuators. On the other side, we place end users of the target application. A crucial feature of Iotas is their incorporation of subtle relationships between users and resources, which are more pronounced than in the case of traditional web applications, because Iota users have direct relationship with the deployment and operation of infrastructural elements, not only in the information exchanged.

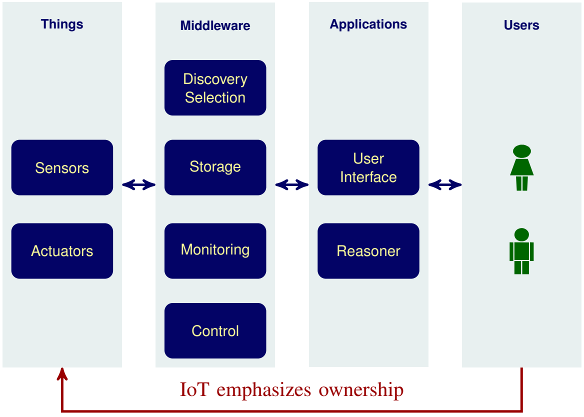

Fig. 1. Main architectural elements of IoT.

In simple terms, an Iota consists of a user interface and a reasoner, the latter being a locus of the decision making by the application. The Iota deals with IoT devices, not directly, but through extensive middleware. Some of the middleware provides abstractions for the IoT in the nature of monitoring (for the sensors) and control (for the actuators). Some middleware concerns storage and some concerns service selection, both of which we expand on below.

The main enablers for practical IoT applications include the following. First, data technologies, including storage, streaming, and analytics, are a major enabler for the IoT. Since each IoT node potentially produces data continually and there are envisioned to be tens of billions of them, they can together generate a lot of data. Of course, each specific environment would have only a few IoT nodes, but the data generated by them can add up to large volumes. Second, reasoning about such data quickly enough to inform intelligent decision making can be nontrivial but is becoming ever more tractable because of improved technology and advances in algorithms. For this reason, cloud computing, which provides elastic storage and compute capacity to contend with the scale and variations in the scale of IoT applications is also a key enabler. Third, information semantics technologies, including linked data, as a basis for sense making of the large amount of otherwise uninterpreted data obtained from sensors. Fourth, to engineer and maintain IoT applications requires ways of programming, such as service abstractions for things and data.

III. What the IoT Needs

IoT needs ways for sharing, fusing, revising information that can keep with multiple IoT devices continually providing data, sometimes mutually contradictory and sometimes conveying revisions of previously provided data. Handling such systematically requires evidential reasoning as well as ways of capturing domain knowledge.

However, as we explained above, we would like to focus here on challenges that are not the obvious ones of data handling and analytics and instead lie at the core of distributed computing. In that spirit, we next describe decentralized perspectives, decoupled enactment, and governance.

Currently, an Iota is realized via a conceptually *unitary* machine—a single logical locus of control. We adopt the term *agent* for a unitary machine to draw attention to the fact that it performs application-level reasoning. Essentially, the agent orchestrates computations as specified by its designer. In general, it would run some variant of a sense-reason-act loop. The agent is unitary despite the fact that its components may be distributed and that it might be sitting on top of infrastructure that is decentralized. In traditional computer science, entire applications are typically modeled and implemented as unitary machines, that is, as a single agent.

For concreteness, we return to our healthcare scenario. We may imagine many agents, each corresponding to an independently designed Iota. The health service's agent enables the recording of a patient's health indicators via wearable technology and facilitates the automated delivery of medications based upon analysis carried out by the health service. A fitness center's agent accesses its subscribers' health indicators via a service interface to the health service's repository and automatically calibrates exercise machines to fit a subscriber's health profile whenever he or she goes to the fitness center. Other parties such as insurance companies, family members, and research organizations have their own agents. The crucial observation to make here is that although there are many agents, there is *no* multiagent system—no two agents interact with each other. Instead, following current software engineering practice, each agent sits atop a shared database, in our example, of health indicators and other information.

We emphasize that the health service's agent clearly has a distributed implementation: it gathers information from a variety of distributed sensors, stores and processes it in the health service's data center, and actuates the delivery of the appropriate medication to the patient. However, it is a single locus of control.

A. Decentralization

The Iota computational model in Figure 1, while hitting the highlights, is overly simplistic. What truly characterizes Iotas and sets them apart from conventional applications is that they tend to involve multiple administrative domains, or parties. In general, each party contributes (information from) a few things and consumes information (from things) contributed by others and provide services to others. The point is that the parties interact with each other on the basis of the applicable *rules of encounter*, that is, *interaction protocols*. Under this doctrine, the application is modeled via the interactions between the autonomous entities that feature in it, each of which is modeled as an agent, that is, a unitary machine (although it may recursively exhibit internal structure). The interaction protocol in fact specifies the MAS by capturing the social process that the parties engage in via their agents. In the health scenario, there may be rules of encounter that capture a patient's expectation that the health service share information only with authorized third parties; a patient's expectation of safe and effective drug delivery; a regulator's expectation that health services monitor for potentially dangerous states of indicators and intervene appropriately; and so on.

Specifying an application in terms of a protocol is the key to decentralization. Each party that plays a role in the protocol can implement its own agent to automate its decision-making and facilitate its interactions with other parties. Whereas following current methods, we would realize an Iota as a single agent, in the interaction-oriented viewpoint, an Iota is characterized by an interaction protocol in which agents representing real-world parties adopt roles. In other words, each Iota instantiation is inherently multiagent. As Figure 2 shows, IoT's inherent decentralization calls for a multiple machine, that is, a multiagent architecture. Here, each agent represents an autonomous party and contains a reasoner. We defer discussion of agent-agent interaction and the Org to the next two subsections.

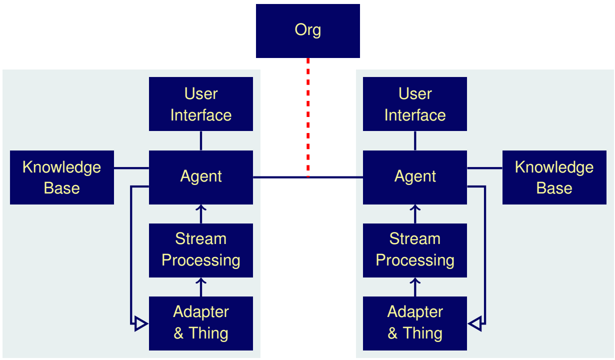

Fig. 2. Toward a multiagent architecture for IoT.

B. Asynchrony and Decoupled Enactment

As Figure 2 shows, decentralization requires that agents interact with one another. That is, the protocols we refer to must be enacted through arms-length communications—in other words, via messaging between the agents. Correct decentralized enactment of protocols via messaging is a major challenge in distributed computing [3]–[6]. The challenge arises from the fact that messages may be lost, delayed, and reordered in transmission; duplicates may be received; and in general different agents would observe different messages in incompatible orders. Further, concurrency means that distinct instances (enactments) of a protocol may be interleaved. These aspects entail that agents may potentially end up in incompatible states during a protocol enactment, leading to a failure in interoperability.

In current research, discussions of asynchrony in Iota mostly arise in connection with low-level protocols, such as MQTT [7] or CoAP [8]. At the application level, discussions of asynchrony are limited to event processing architectures [9]. There is little discussion, however, of protocol enactments where the end points are maximally decoupled from each other. What the IoT needs in particular is a way of supporting decoupled enactment that provides a high-level information model characterizing the adopted protocol.

C. Governance: Security, Accountability, Privacy

Since a crucial feature of Iotas is their multiple stakeholders, accommodating their diverse requirements is a major challenge. Accordingly, we identify the need for governing interactions to gain some assurance that a collaboration is proceeding successfully, that is, in a manner that meets stakeholder requirements (assumed to be suitably reconciled when mutually inconsistent). Governance requires some measure of social control. For concreteness, as Figure 2 shows, we imagine that agent interactions take place within the scope of the Org that regulates those interactions.

An Org provides a computational basis for representing autonomy and accountability, which are the two fundamental concepts in understanding how stakeholders relate with each other in an Iota. Autonomy means each party is free to act as it pleases; accountability means that a party may be called upon to account for its actions. Mamdani and Pitt [10] use autonomy and accountability as bases for distinguishing stakeholders from their computational agents. In general, balancing autonomy and accountability is crucial for ensuring that an Iota would not devolve into the extremes of chaos or tyranny.

Accountability is classically understood, e.g., in political theory [11] and healthcare [12], in terms of the standing of one party—the *account-taker*—to expect certain behavior from another—the *account-giver*. Although accountability is increasingly recognized as an important theme, existing computational approaches for accountability disregard its core intuitive basis. Some approaches labeled "accountability" in the traditional distributed computing and networking literatures, e.g., [13], [14], address *traceability* of actions instead. Traceability is an important mechanism for holding someone accountable, but is neither necessary nor sufficient for accountability. Traceability is not necessary because accountability holds even without adequate traceability and could be adequate depending upon assumptions of trustworthiness. For example, a patient may hold a health service accountable for loss of privacy even if the loss was caused by an untraceable attacker or by equipment failure. Traceability is not sufficient because even perfect traceability can be circumvented through external interactions. For example, if Alice circumvents traceability by getting Bob to act on her behalf, she remains accountable. A challenge is to align our computational models of rich concepts such as accountability with what users expect, not merely to develop artificial computational models that may be easier to implement but miss the point of the original concept.

To this end, it is crucial to develop computational representations of the governance of stakeholder interactions as these interactions reflect their respective requirements and support successful interoperation. Such high-level representations would provide a basis for validating the technical policies through which IoT devices and gateways carry out their decision making in regards to how they interoperate.

Security and privacy are crucial concerns in the IoT since devices gather valuable personal or corporate information. An IoT device being compromised can have significant repercussions on the privacy of those concerned and possibly on their physical security. One aspect of this challenge falls under the rubric of conventional (technical) cybersecurity. Another aspect, dealing with privacy and acceptable use of IoT resources is more a matter of governance, especially accountability, than of conventional cybersecurity—we include this aspect in our present scope. Along the same lines, the representations of interest here lie above constructs such as identity, including ways to pair devices [15].

IV. What Decentralized MAS can Offer

Multiagent systems provide important constructs that help address the major challenges confronting Iotas.

A. Agents as Service Endpoints

First, MAS provides ways of dealing with autonomy and heterogeneity. Autonomy refers to the idea that stakeholders, especially IoT device owners, data aggregators, and end users, are able to decide and act independently. Heterogeneity refers to the idea that devices and datasets and data streams pertaining to the devices are independently structured. In essence, the information models underlying these resources are not necessarily in agreement. In addition, MAS enables discovery and selection of IoT devices and data resources. A usual approach to deal with heterogeneity is through semantic service descriptions and agents who adopt and apply a common ontology as a basis for information interchange [16].

Second, MAS has developed approaches for trust. One category of approaches deals with gathering, maintaining, and disseminating reputation information. These approaches can be adapted for IoT devices and their owners. The other category engenders trustworthiness through incentives to align autonomous parties' interests with those of each other, e.g., so that self-interested providers of IoT devices behave in a trustworthy manner. Such incentives can induce those who own and configure IoT devices to, first of all, engage in a particular application by contributing data from their devices and, second, where appropriate to configure them to expend the requisite power to obtain and transmit high-quality observations.

Third, we understand *collective intelligence* as how multiple parties can jointly exhibit knowledge and decision making superior to any of them individually. Collective intelligence, which generalizes on crowdsourcing, applies in some important parts of the IoT application lifecycle. For example, we might need to collectively maintain service descriptions, specify protocols, create norms that address multiple stakeholder concerns; and guide emergence of prosocial behaviors. Along these lines, MAS provides approaches such as from *social choice theory*. Specifically, we would apply judgment aggregation to fuse not the raw data but the decisions made using that data. Likewise, preference aggregation applies in IoT where there is need for a collective decision, e.g., in adopting standards for communication or determining what attributes to sense and at what precision to sense them. It may be achieved through various voting mechanisms.

B. Decentralization via Information Protocols

Because of the importance of autonomy and heterogeneity, to realize IoT applications presupposes an ability to construct decentralized MAS. Here, a *decentralized MAS* refers to a MAS in which the agents maintain their goals separately and privately and autonomously act as they deem appropriate. In addition, we avoid assumptions of the infrastructure that would interfere with the autonomy of the agents. In particular, the agents do not wait for one another because of the infrastructure though they may wait if the application demands waiting. This leads to the idea of *asynchronous* communication where an agent may send a message to another agent whenever it wants to and an agent may work on a received message anytime after it has received it.

Engineering a system under these constraints requires that we precisely specify *interaction protocols* to enable interoperation despite the decentralization. To this end, formal models, languages, and techniques for protocols are valuable. The established approaches (e.g., [17], [18]) are generally focused on control flow structures, which are generally not well suited to the modeling of IoT applications for two reasons. One, in settings of autonomous parties, no one can directly control the actions of another. Two, control flow abstractions such as loops and conditionals are fine for sequential programs where an instruction executes fully before another the execution of another begins; however, they are incompatible with asynchronous enactments where the "instructions" are messages, which may be lost or inordinately delayed, with the result that every party may potentially see a different order of instructions.

The only currency in a computing system is information. Therefore, all correctness requirements must be based on how information flows or fails to flow. For decentralized systems, we further separate the concern of information flow between parties from the concern of information processing within a party's IT resources. The former concern is captured by protocols. Even when the control flow is prominent in the modeling, it must be mapped to information flow between decentralized components in order to realize it and to verify that it is correctly realizable. The components are referred to as *local perspectives* (the control flow specification being the global one) or *role skeletons* [3]–[6]. The control flow approaches often apply sophisticated techniques but they fail to contend with the basic limitation that there are two competing standards: the control flow specification and the information flow realization.

In contrast, information-based protocols [19] make the modeler directly responsible for specifying *causality* and *integrity* as expressed in terms of information. The associated control flow is *any* control flow that is compatible with causality and integrity. Further, as causality and integrity determine the most general set of correct flows, an information protocol would typically be more flexible than any correct handcrafted control-flow specification.

We now give a flavor of a declarative information protocol. Listing 1 shows a simple request response protocol in BSPL [19], an information-based protocol language.

Listing 1. A request response protocol in BSPL.

```
Request Response {

 role Requester, Responder
 parameter in ID key, out query, out answer

 Requester ↦ Responder: request[in ID, out query]

 Responder ↦ Requester: response[in ID, in query, out
     answer]
}
```

We highlight the salient features of Listing 1. *Request Response* is the name of the protocol; *request* and *response* are message schemas. The key ID uniquely identifies instances of *Request Response*, *request*, and *response*. This protocol is not enactable standalone because at least one parameter is ⌜in⌝, meaning that its binding must come from composition with another protocol. An instance of *request* must precede any instance of *response* with the same ID because query is ⌜out⌝ in *request* but ⌜in⌝ in *response*. The autonomy of the parties means that no message need occur. And, *response* must occur for *Request Response* to complete because it binds answer which is a parameter of *Request Response*.

The basic idea behind information-based protocols is that an agent can send any message whose parameter bindings it either knows or can generate. Notably, there is no specification of control flow in Listing 1 (or in Listings 2 and 3 for that matter).

Listing 2 shows a more complex sensor subscription setup protocol. This protocol may be composed with the sensor data transfer protocol in Listing 3. Composition is a strength of BSPL.

Listing 2. A sensor subscription setup protocol in BSPL.

```
Subscription Setup {

 role Publisher, Subscriber
 parameter out sID key, out metadata, out rID

 Subscriber ↦ Publisher: request[out sID, out metadata]

 Publisher ↦ Subscriber: accept[in sID, in metadata, out
     rID]

 Publisher ↦ Subscriber: reject[in sID, in metadata, out
     rID]
}
```

Listing 3. A sensor data transfer protocol in BSPL.

```
Sensor Subscription {

 role Sensor, Subscriber
 parameter in sID key, out req key, out dataitem key

 private end

 Subscriber ↦ Sensor: start[in sID, out req]

 Subscriber ↦ Sensor: stop[in sID, in req, out end]

 Sensor ↦ Subscriber: dataResponse[in sID, nil end, in
     req, out dataitem]
}
```

C. Norms as a Computational Foundation

A norm captures an expectation, a standard for correct interaction in a social setting [20]. Norms can be descriptive (what is established practice) and regulative (what is expected) [21]. We consider regulative norms, each of which involves at least two parties: the account-taker and the account-giver [22], [23]. Norms in this sense are a way to capture accountability relationships between two parties. For example, we might capture that Alice is committed to Bob that if Bob accepts her offered price, she will provide her sensor stream to him. Another example is of the prohibition placed on the health service by patients on sharing data with third parties, which means that the service is accountable to patients for the privacy of their information.

Norms are important to Iotas because although we cannot force autonomous parties to act in any particular way, norms capture how each of them ought to act. That is, for a decentralized system, instead of asking if the system is behaving correctly, we ask which participant is behaving correctly, that is, respecting the norms that apply to it. A norm is often associated with a sanction, positive or negative, for complying with or violating it [24].

A key idea is to treat norms, not merely as documentation for the specification of an Iota, but as the key high-level specification for an Iota.

D. Meaning-Based Protocols

A higher-level challenge than information protocols is to assign meaning to the information exchanged. To this end, formal semantics is an important and obvious requirement: decentralization demands not only that we model interaction but that we do so in rigorous terms.

Accordingly, we motivate meaning-based protocols, which capture the social meaning associated with the messages to be exchanged. Meanings are commonly expressed in terms of norms, e.g., commitments [25] or norms such as authorization, prohibition, and power [26].

Listing 4 shows a meaning-based protocol, specifically, a commitment specification, in Cupid [27]. A Cardio Care commitment from a nurse (the debtor) to a physician (the creditor) is *created* when the nurse takes charge of a patient, that is, upon the occurrence of TakeCharge and *detached* if a CardiacEvent above the specified threshold happens for this patient within 180 minutes of taking charge. If it is not detached then the commitment *expires*. The commitment is *discharged* if CPR on this patient happens within five minutes of the Cardiac Event, else it is *violated*. TakeCharge, CardiacEvent, and CPR may all be messages specified in an information protocol. Thus, meaning specifications may be layered on top of information protocols. The idea is that the state of a commitment instance progresses as messages are observed and recorded in an agent's local information store; the state can be unambiguously computed from the observed messages.

Listing 4. A commitment specification in Cupid.

```
commitment CardioCare nuID to phID
create TakeCharge
detach CardiacEvent [, TakeCharge + 180]
  where ceMagnitude >= tcThreshold
discharge CPR [,CardiacEvent + 5]
```

V. Promising Directions for Research

We have seen how the IoT is useful, what it relies upon, what challenges it faces, and how decentralized multiagent systems may help. Although MAS promises interesting ideas, our understanding of decentralized intelligence is far from complete. The IoT exposes new challenges that current research does not address adequately.

These limitations suggest new directions for research, which we frame as research questions, written in italics, below.

A. Programming Models for Agents

We posit a programming model that enables programmers to conceive of agents in terms of how they process norms. Each agent acts on behalf of a stakeholder. For example, an agent may schedule the satisfaction of two commitments to others in order of their deadlines (nearest deadline first) or by the identity of the creditor (serving the most important creditor first). Such a programming model has the potential to vastly simplify programming by promoting high-level abstractions. For example, the health service's agent would be designed to process the norms that the health service is party to. An agent may be designed to satisfy norms. In general though, a party would design an agent according to its preferences, which may entail violating some norms. For example, the health service may prefer to violate a prohibition on sharing information with a third party if it determines doing so serves a legitimate patient interest.

Our proposal represents a departure from mainstream agent programming ideas. Agent programming languages, e.g., 2APL [28], [29], remain largely based on concepts that describe the cognitive (internal) states of agents and do not support programming agents from specifications of protocols. In general, works on declarative programming approaches for agents, e.g., [28], [30], [31], are valuable. Some approaches [32] provide mappings from commitments to plans based on cognitive constructs such as beliefs, desires, and intentions. These approaches suggest the promise and feasibility of a norm-based programming model.

> *What are suitable abstractions for programming agents in an Iota that support normative reasoning and facilitate accountability?*

Each agent computes the state of any norm instance relevant to its decision making based solely on the events it has observed, and the semantics of the norm in question. In general, in the spirit of decentralization, we expect that the agents would make different observations and may disagree on states of some norm instances. For example, the user of some lab testing equipment may infer that it has released the resource in time, thereby satisfying its commitment to do so; however, the owner may infer that the commitment was violated. Such a misalignment reflects a potential interoperability problem [33]. Therefore, a crucial requirement for interoperation is to ensure that either such disagreements between agents are not significant to their interoperation or can be resolved through additional communication and computation. Ensuring correct decentralized enactment for meaning-based protocols is crucial to realizing the full value of norms as a human-level architectural abstraction.

> *For what expressive Iota specifications can we automatically and efficiently ensure alignment and under what assumptions regarding decentralized enactment?*

B. Interaction-Oriented Software Engineering

In modeling interactions, protocols capture the interactive, that is, the *public* part of the application logic. Languages such as Cupid and BSPL are promising starting points for meaning-based and information protocols, respectively. Specifying realistically complex Iotas would require the development of more expressive languages, e.g., those that naturally support the streaming and aggregation of information. For example, in auctions of sensor-acquired data, the fact that the highest bidder is declared winner is public and would ideally be specified in a protocol. Splee [34], an extension of BSPL, exhibits features such as dynamic role binding, multiple agents playing a role, and multicast communication to all agents playing a role.

> *What are expressive information protocol specifications that capture the needs of realistic Iotas while retaining ease of decentralized implementation?*

Further, meaning-based protocols would be enacted via information protocols. A rich direction of study is to study the connection between the two. Ideally, we would like to *generate* information protocols from social-level specifications of causality and meaning. At the social level, causality means that a piece of information (e.g., the binding for some parameter) must be *declared* [35] by some agent before it can be used by any agent. The generated information protocols would ideally guarantee meaning alignment. Success in this direction would ensure that much of the technical architecture required to support an Iota could be generated from specifications written using abstractions much closer to stakeholder vocabulary. By contrast, most MAS approaches today deal only with synchronous communication where each of the communicating parties must wait for the other [36]–[38]. Not only does synchrony conflict with autonomy (one party waiting for another to progress), it also yields poor scalability—important since there are potentially large numbers of devices communicating with others in an Iota.

> *How can we reduce or eliminate the impedance mismatch between information and meaning-based protocols, specifically, supporting asynchrony while providing a veneer of a system-wide view of an Iota's state?*

We would need to develop a methodology around the protocol languages that would support (1) refining stakeholder requirements into protocols, e.g., as in [39], [40]; (2) protocol verification tools, e.g., to verify safety and liveness of information protocols [34], [41] and to verify the satisfiability and falsifiability of norms [27]; (3) middleware to ensure correct decentralized enactment [42]; and (4) interaction-oriented programming models for agents, as described above. We term this overall methodology Interaction-Oriented Software Engineering (IOSE) [43] to distinguish it from Agent-Oriented Software Engineering (AOSE). Whereas AOSE emphasizes specifying and composing agents to build systems, IOSE emphasizes specifying and composing interactions to build systems.

> *How can we capture stakeholder requirements in a manner that exposes the norms that impinge upon the experience of others in their Iota while encapsulating internal decision making?*

C. Enlightened Governance

As mentioned earlier, regulative norms yield accountability relationships, whereas descriptive norms describe what principals *normally* do in an Iota. The descriptive norms of an Iota may deviate from its regulative norms. For example, third parties are accountable for deleting patient data when they decommission resources that have internal storage; however, it may turn out that in practice they do not do so.

Descriptive norms inform adaptation of the Iota. Specifically, a motivation for adaptation would be to close the gap between the descriptive and regulative norms. This is what we refer to as *enlightened governance*. Enlightened governance requires advances in many areas. We would need to be able to infer the descriptive norms of an Iota from observations or records of interactions [44], [45]. Further, once the descriptive norms are known, stakeholders would decide whether and how to close the gap. Many options may be available. The stakeholders may drop some of the norms (if they are obsolete with respect to current requirements) or introduce stronger incentives (positive or negative) to promote their satisfaction. The stakeholders could be supported in their reasoning by simulations of what may emerge in the adapted Iota [46]. Deliberation technologies, e.g., to support participation in discussion and debate and to extract actionable information would also be highly useful [47], [48].

> *How can we evaluate the quality of the norms underlying an Iota in terms of their support for stakeholder requirements and their resilience to "attacks" by noncooperative participants?*
>
> *How can we support deliberation about Iotas leading to consensus regarding any proposed design or redesign?*

Research on operating systems has taught us much about the management of resources in unitary machines, but it has little to say about governance, which concerns the administration of shared resources by autonomous stakeholders. To support enlightened governance, what we need is an "operating system" that provides the computational infrastructure (e.g., messaging services and norm stores), services (e.g., for recognition of descriptive norms and determining gaps between the regulative and descriptive norms), and protocols to support stakeholder deliberation and judgment aggregation. Such an operating system would be nothing like a traditional operating system though because its "processes" would be autonomous parties.

> *What are possible architectural abstractions for governance that support a computational representation of Iotas and associated services to automate governance in Iotas?*

VI. Conclusions

In a nutshell, the emergence of the IoT challenges the field of distributed computing to support multiple stakeholders engaging in complex interactions sometimes over highly constrained resources. Efforts on networking, data management, and analytics will not be adequate by themselves. We need new ways to support flexible reasoning, enactment, and governance in the social sphere. To do so would require a deeper treatment of how decentralized intelligence can be embedded into distributed computing. This means not merely patching existing approaches but placing decentralized intelligence constructs such as norms at the heart of modern distributed systems.

It is worth considering what makes the IoT an important setting for the proposed research into distributed computing. First, the IoT differs from previous IT advances such as federated databases, the WWW, and service-oriented computing in key respects. First, the IoT forces a distribution of resources. Distribution is nominally demonstrated by the above-mentioned application areas but mainly as a convenience. In practice, distribution has been reduced in system architectures. Instead of true distributed computing, it has been economical to develop semicentralized architectures such as cloud computing.

Second, the IoT is conducive to independent ownership and independent operation of resources. This is because IoT devices are physically distributed and cross jurisdictional boundaries and are therefore well aligned with business models in which some of the ownership is likewise spread over the stakeholders. Increasing recognition of privacy risks with the

IoT brings up the need for incorporating governance within an Iota, which is possible only if we develop computational representations of the social sphere in which an Iota exists.

Acknowledgments

Munindar Singh was partially supported by the US Department of Defense for support through the Science of Security Lablet at NC State University and by the NCSU Laboratory for Analytic Sciences. Amit Chopra was supported by the EPSRC grant EP/N027965/1 (*Turtles*).

REFERENCES

[1] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[2] T. C. King, Q. Liu, G. Polevoy, M. de Weerdt, V. Dignum, M. B. van Riemsdijk, and M. Warnier, "Request driven social sensing (Demonstration)," in *Proceedings of the 13th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. Paris: IFAAMAS, May 2014, pp. 1651–1652.

[3] S. Basu, T. Bultan, and M. Ouederni, "Deciding choreography realizability," in *Proceedings of the 39th annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. Philadelphia, PA, USA: ACM, 2012, pp. 191–202.

[4] M. Carbone, K. Honda, N. Yoshida, R. Milner, and S. Ross-Talbot, "A theoretical basis of communication-centered concurrent programming," October 2006, http://www.w3.org/2002/ws/chor/edcopies/theory/note.pdf.

[5] M. Charalambides, P. Dinges, and G. Agha, "Parameterized concurrent multi-party session types," in *Proceedings of the 11th International Workshop on Foundations of Coordination Languages and Self Adaptation*, ser. Electronic Proceedings in Theoretical Computer Science, vol. 91. Open Publishing Association, 2012, pp. 16–30.

[6] N. Desai, A. U. Mallya, A. K. Chopra, and M. P. Singh, "Interaction protocols as design abstractions for business processes," *IEEE Transactions on Software Engineering*, vol. 31, no. 12, pp. 1015–1027, December 2005.

[7] OASIS, "MQTT 3.1.1 specification document," *OASIS Standard*, Oct. 2014, previously known as the Message Queuing and Telemetry Transport; http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf.

[8] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," Internet Engineering Task Force (IETF), Tech. Rep. RFC 7252, Jun. 2014, proposed standard; https://tools.ietf.org/html/rfc7252.

[9] A. Artikis, O. Etzion, Z. Feldman, and F. Fournier, "Event processing under uncertainty," in *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, 2012, pp. 32–43.

[10] E. A. Mamdani and J. Pitt, "Responsible agent behavior: A distributed computing perspective," *IEEE Internet Computing*, vol. 4, no. 5, pp. 27–31, Sep. 2000.

[11] R. W. Grant and R. O. Keohane, "Accountability and abuses of power in world politics," *American Political Science Review*, vol. 99, no. 1, pp. 25–43, Feb. 2005.

[12] L. L. Emanuel, "A professional response to demands for accountability: Practical recommendations regarding ethical aspects of patient care," *Annals of Internal Medicine*, vol. 124, no. 2, pp. 240–249, Jan. 1996.

[13] K. Argyraki, P. Maniatis, O. Irzak, S. Ashish, and S. Shenker, "Loss and delay accountability for the Internet," in *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, Beijing, Oct. 2007, pp. 194–205.

[14] A. Haeberlen, "A case for the accountable cloud," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 52–57, Apr. 2010.

[15] XMPP, "XMPP Internet of Things," XMPP Standards Foundation; http://www.xmpp-iot.org/, 2015.

[16] M. P. Singh and M. N. Huhns, *Service-Oriented Computing: Semantics, Processes, Agents*. Chichester, United Kingdom: John Wiley & Sons, 2005.

[17] ITU, "Message sequence chart (MSC)," Apr. 2004, http://www.itu.int/ITU-T/2005-2008/com17/languages/Z120.pdf.

[18] WS-CDL, "Web Services Choreography Description Language, version 1.0," Nov. 2005, http://www.w3.org/TR/ws-cdl-10/.

[19] M. P. Singh, "Information-driven interaction-oriented programming: BSPL, the Blindingly Simple Protocol Language," in *Proceedings of the 10th International Conference on Autonomous Agents and MultiAgent Systems*, 2011, pp. 491–498.

[20] G. H. Von Wright, *Norm and Action: A Logical Enquiry*, ser. International Library of Philosophy and Scientific Method. New York: Humanities Press, 1963.

[21] R. B. Cialdini, R. R. Reno, and C. A. Kallgren, "A focus theory of normative conduct: Recycling the concept of norms to reduce littering in public places," *Journal of personality and social psychology*, vol. 58, no. 6, pp. 1015–1026, 1990.

[22] G. H. Von Wright, "Deontic logic: A personal view," *Ratio Juris*, vol. 12, no. 1, pp. 26–38, Mar. 1999.

[23] M. P. Singh, "Norms as a basis for governing sociotechnical systems," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 1, pp. 21:1–21:23, Dec. 2013.

[24] A. R. Radcliffe-Brown, "Social sanction," in *Encyclopaedia of the Social Sciences*, E. R. A. Seligman, Ed. Macmillan Publishers, 1934, vol. XIII, pp. 531–534, reprinted in *Structure and Function in Primitive Society*, chapter 11, pages 205–211, The Free Press, Glencoe, Illinois, 1952.

[25] P. Yolum and M. P. Singh, "Flexible protocol specification and execution: Applying event calculus planning using commitments," in *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems*. ACM Press, 2002, pp. 527–534.

[26] A. K. Chopra and M. P. Singh, "Custard: Computing norm states over information stores," in *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. Singapore: IFAAMAS, May 2016, pp. 1096–1105.

[27] ——, "Cupid: Commitments in relational algebra," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015, pp. 2052–2059.

[28] R. H. Bordini, J. F. Hübner, and M. J. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason*. Chichester, United Kingdom: John Wiley & Sons, 2007.

[29] M. Dastani, "2APL: A practical agent programming language," *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, vol. 16, no. 3, pp. 214–248, Jun. 2008.

[30] M. Baldoni, G. Boella, V. Genovese, R. Grenna, and L. van der Torre, "How to program organizations and roles in the JADE framework," in *Proceedings of the 6th German Conference on Multiagent System Technologies*, ser. Lecture Notes in Computer Science, vol. 5244. Kaiserslautern, Germany: Springer, Sep. 2008, pp. 25–36.

[31] N. Alechina, M. Dastani, and B. Logan, "Programming norm-aware agents," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'12)*. Valencia: IFAAMAS, 2012, pp. 1057–1064.

[32] M. Winikoff, "Implementing commitment-based interaction," in *Proceedings of the 6th International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. Honolulu: IFAAMAS, May 2007, pp. 868–875.

[33] A. K. Chopra and M. P. Singh, "Generalized commitment alignment," in *Proceedings of the Fourteenth International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS, 2015, pp. 453–461.

[34] A. K. Chopra, S. H. C. V, and M. P. Singh, "Splee: A declarative information-based language for multiagent interaction protocols," in *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. São Paolo: IFAAMAS, May 2016.

[35] J. L. Austin, *How to Do Things with Words*. Oxford: Clarendon Press, 1962.

[36] J. Odell, H. V. D. Parunak, and B. Bauer, "Representing agent interaction protocols in UML," in *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering (AOSE 2000)*, ser. Lecture Notes in Computer Science, vol. 1957. Toronto: Springer, 2001, pp. 121–140.

[37] U. Endriss, N. Maudet, F. Sadri, and F. Toni, "Protocol conformance for logic-based agents," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2003, pp. 679–684.

[38] T. Miller and J. McGinnis, "Amongst first-class protocols," in *Proceedings of the 8th International Workshop on Engineering Societies in the Agents World (ESAW 2007)*, ser. Lecture Notes in Computer Science, vol. 4995. Springer, 2008, pp. 208–223.

[39] A. K. Chopra, F. Dalpiaz, F. B. Aydemir, P. Giorgini, J. Mylopoulos, and M. P. Singh, "Protos: Foundations for engineering innovative sociotechnical systems," in *Proceedings of the 18th IEEE International Requirements Engineering Conference*, 2014, pp. 53–62.

[40] A. Günay, M. Winikoff, and P. Yolum, "Dynamically generated commitment protocols in open systems," *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, vol. 29, no. 2, pp. 192–229, 2015.

[41] M. P. Singh, "Semantics and verification of information-based protocols," in *Proceedings of the 11th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. Valencia, Spain: IFAAMAS, Jun. 2012, pp. 1149–1156.

[42] ——, "LoST: Local State Transfer—An architectural style for the distributed enactment of business protocols," in *Proceedings of the 9th IEEE International Conference on Web Services (ICWS)*. Washington, DC: IEEE Computer Society, Jul. 2011, pp. 57–64.

[43] A. K. Chopra and M. P. Singh, "From social machines to social protocols: Software engineering foundations for sociotechnical systems," in *Proceedings of the 25th International World Wide Web Conference*, Montréal, 2016, pp. 903–914.

[44] J. Morales, M. López-Sánchez, J. A. Rodriguez-Aguilar, M. Wooldridge, and W. Vasconcelos, "Automated synthesis of normative systems," in *Proceedings of the 12th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. St. Paul, Minnesota: IFAAMAS, May 2013, pp. 483–489.

[45] D. Avery, H. K. Dam, B. T. R. Savarimuthu, and A. K. Ghose, "Externalization of software behavior by the mining of norms," in *Proceedings of the 13th International Conference on Mining Software Repositories (MSR)*. Austin, Texas: ACM, May 2016, pp. 223–234.

[46] B. T. R. Savarimuthu and S. Cranefield, "Norm creation, spreading and emergence: A survey of simulation models of norms in multi-agent systems," *Multiagent and Grid Systems*, vol. 7, no. 1, pp. 21–54, 2011.

[47] A. K. Chopra and M. P. Singh, "Colaba: Collaborative design of cross-organizational business processes," in *Proceedings of the International Workshop on Requirements Engineering for Systems, Services, and Systems of Systems (RES^4)*. Trento, Italy: IEEE, Aug. 2011, pp. 36–43.

[48] M. Lippi and P. Torroni, "Argumentation mining: State of the art and emerging trends," *ACM Transactions on Internet Technology (TOIT)*, vol. 16, no. 2, 2016.