Decentralized Governance of Networked Systems: from Access Control to Interaction Control

Naftaly Minsky

Rutgers University

March 22, 2009

Contents

Abstract

This paper introduces an abstract model for mechanisms for the governance of large, heterogeneous, and open networked systems. This, so called *interaction control* (IC), model goes well beyond conventional access control, along a number of dimensions. In Particular, the IC model: (1) is inherently decentralized, and thus scalable even for wide range of stateful policies; (2) is very general, and not biased toward any particular type of policies; thus providing a significant realization of the age-old principle of *separation of policy from mechanism*; and (3) enables flexible, composition-free, interoperability between different policies.

The IC model, which is an abstraction of a mechanism called LGI, has been designed as a minimalist reference model that can be reified into a whole family of potential governance mechanisms that may support different types of communication, with different performance requirements, and for different application domains. Such IC mechanisms can also be extended with important features such as: *obligation*, treatment of *exceptions*, and the ability to organize policies into *conformance hierarchy.*

1 Introduction

The economy and security of modern society depends increasingly on the software that supports complex institutions, such as banking systems, health care organizations, government agencies, and commercial enterprises. And this software is undergoing a relentless transition from monolithic systems, constructed according to a single overall design, into loosely coupled, networked, and heterogeneous systems, whose component parts may be written in different languages, may run on different platforms, and may be designed, constructed, and even maintained under different administration domains. We will refer to such systems as *open*[1], in part because their component may change dynamically, or disappear, while new components may be added to a system in an unpredictable manner. The emerging concept of *service oriented architecture* (SOA) is a prominent example of this trend.

Unfortunately, while there are powerful reasons for this transition to open systems—including the tendency of societal institutions to interoperate and to form federations, such as *grids*, *virtual organizations* (VOs), and *supply chains*—the heterogeneous and semi-anarchical nature of such systems engenders serious obstacles to their manageability, dependability and security.

These obstacles include, but are not limited to, the following: (1) The difficulty to protect an open system against attacks by unruly or malicious *actors*[2], from within the system or from the outside—a difficulty which is exacerbate by the inability of a geographically distributed networked system to protect itself by hiding within its local intranet, behind the *Maginot line* of their firewalls (to use Bill Wulf's analogy[3]). (2) The difficulty to dynamically manage—i.e., monitor and control—an open system; in particular, because the would be managers have little, if any, sway over components of such a system, which may be dispersed all over the Internet, and whose often heterogeneous code may not be available, or even known, to the managers. And (3) the difficulty to facilitate safe and harmonious coordination between the system's disparate actors, which may have little or no knowledge of each other's behavior.

We claim that these difficulties can be alleviated by suitable governance of the interaction between the disparate actors of a system, and without assuming any knowledge of, or control over, the structure or internal behavior of the interacting actors themselves. We will refer to such governance as *interaction control* or IC, rather than the traditional "access control," because we believe that it is necessary to govern the dynamic process of interaction between actors, and not just to deny access to resources from those that do not have the *right* for them—which is the main objective of conventional access control (AC) mechanisms.

One may doubt the usefulness of governance that regulates *only* the interactions between

[1]Note that the term "open system," as used here, has nothing to do with the concept of open source.

[2]By an "actor" we mean either an autonomous software component, or a person operating via some software interface.

[3]An unpublished address by W. Wulf at an NSF meeting on the cyberTrust program, 2004.

the actors (or components) of the system being governed. But the critical role that essentially such governance plays in societal systems, suggests that it could be similarly useful when applied to software systems as well. The transportation system, for example, is governed by traffic laws that regulate such things as: to whom should one give the right of way, and how should one react to the changing traffic lights. Such laws are oblivious of the intentions of individual drivers, yet they enable drivers to coordinate harmoniously with each other, and to negotiate their passage through intersections with relative safety. This, despite the fact that the drivers themselves are an heterogeneous bunch, with little if any knowledge of each other.

But we believe that for IC kind of governance to be able to address the difficulties inherent in open systems, the governance mechanism needs to satisfy the following set of principles.

1.1 The IC Principles of Governance of Open Software Systems

We propose here five principles that a governance mechanism needs to satisfy to be effective for open software systems, and will explain briefly the rationale of each of them. We call them the *IC principles.* We will elaborate on these principles, and on their broader implications, throughout this paper.

Principle 1 (statefulness) *System governance needs to be sensitive to the history of interaction between the actors operating in the system. And for such sensitivity to be scalable, the governance mechanism must be able to maintain* state *that represents relevant functions of the interaction history. Such a mechanism is called* stateful.

History sensitivity is obviously necessary for regulating coordinations, and for supporting management, both of which are highly dynamic processes. And it is also necessary for the security of systems, which can be endangered by improper dynamic behavior of actors, such as sending message in an improper order, or sending too many messages, thus causing denial of service. The precise relationship between such sensitivity and satefulness will become evident in due course.

Principle 2 (locality and decentralization) *A governance mechanism should be local, in the following sense: the constraints imposed on the interactive behavior of a given actor can depend only on the history of* its own interactions *with the rest of the world. Such a mechanism is also called* decentralized.

This is important because without locality one would need to employ a central mediator (or *reference monitor* to mediate the interactions between all actors. And the use of a central mediator is problematic for several reasons. First, it makes the governance unscalable, particularly when dealing with stateful policies[4]. (This is the case even if the mediator is replicated, because, as has been shown in [22], replication tends not to improve scalability under stateful regulation.) Second, centralized mediation tends to distort the inherent concurrency, and the independence, of the interactive activities of distributed actors. Finally, local (and thus decentralized) governance mechanism is obviously necessary for the regulation of the interaction between actors communicating via *mobile ad hoc networks* (MANETs).

Principle 3 (global sway) *Despite its local (decentralized) nature, a governance mechanism should have global sway over the system under its jurisdiction.*

What we mean here by "global sway" will be explained in Section 3.2.

Principle 4 (flexible interoperability) *A governance mechanism should provide flexible means for actors operating under different policies to interoperate, without having to compose their respective policies into a single policy.*

Interoperability is critical due to the growing need for actors operating under disparate policies to interact. This is the case, in particular, for B2B commerce, for virtual organizations of various kinds, and for supply chains. The need to carry out such interoperability without having to compose the policies of the interacting parties is due to the fact that composition is computationally hard [18], and tends to be very flexible, as argued in [2].

Principle 5 (separation of policy from mechanism) *A governance mechanism should not be biased towards any particular type of policies.*

This is the age-old principle proposed more than thirty year ago [30] for access control, but never fully satisfied. It means, essentially that a governance mechanism should not be designed specifically for any particular type of policies—such as a particular ways for doing delegation and revocation of rights. But it should instead be able to support a wide range of policy types, in a uniform manner. This principle is becoming increasingly important because modern complex systems tend to require a multitude of diverse policies, regarding various types of system activities, and various system divisions. Using different mechanisms to implement such policies would be very hard, and it would preclude interoperation between actors subject to different types of policies.

[4]The term "policy" is used in this paper either informally, in its dictionary sense, or in its sense under traditional access control, whenever access control is being discussed.

1.2 The Objective of this Paper

The main goals of this paper are the following: (a) to introduce an abstract reference model of governance mechanisms that satisfy the IC principles—it is called the *IC model*; and (b) to explore the resulting structure of this model. This reference model, is intended to serve as a basis for a whole family of concrete *IC mechanisms* that would satisfy our principles, but may be designed for different types of communication, different performance requirements, and different application domains; and which may extend the minimalist IC model in various ways. The IC model itself is an abstraction of the implemented *law governed interaction* (LGI) mechanism [20]. But it features only those aspects of LGI that are essential for the support of our principles, and is not encumbered by the many details of this concrete mechanism.

The rest of this paper is organized as follows. The IC model is introduced in Section 2. Section 3 explores various properties of this model, and elucidate the manner in which it satisfies our principles. This analysis of the IC model will be done in comparison with conventional *access control* (AC)—the currently dominant approach to the regulation of distributed systems—which, as we shall see, does not, on the whole, satisfy the IC principles, nor has it been designed to do so. The paper concludes in Section 4.

2 The Interaction Control (IC) Model

We introduce here an abstract model of mechanisms for the governance of the interactions (via message exchange) between distributed actors. This model, which is based on the IC principles, is oblivious of the internal structure and behavior of the interacting actors, and independent of the languages in which these actors are programmed, and of the type of host on which they run. In other words, the actors whose interactions are to be regulated are viewed as black boxes that send and receive messages. (This enable the IC model to deal with program modules and people in completely uniform manner, treating them all as actors.)

The IC model is abstract in the following respects. First, it does not specify the type of message passing being used for inter-actor interactions; which may, in particular, be asynchronous, synchronous (via RPC of some kind), or both. Second, this model makes no assumptions about the reliability and latency of the message passing, or about the underlying fabric of the communication network—which may, in particular, be the Internet, or some form of ad hoc wireless communication. Third, this model leaves many details unspecified, which makes this model very lean.

The abstract nature of this model enables it to be reified into a whole family of concrete regulatory mechanisms—called *IC mechanisms*—one of which being the existing LGI mechanism, of which this model is an abstraction. Such mechanisms can be formed by completing the elements of this model, which were not fully specified, and possibly by extending them in various ways, as we will point out in due course.

The most basic elements of this model are: (a) the concept of a *interaction law*, (or, simply a *law*) representing a constraint (or a set of constraints) on the sending and receipt of messages; (b) the concept of an *$\mathcal{L}$-agent*, which, broadly speaking, is an actor engaged in the sending and receiving messages, subject to a specific law $\mathcal{L}$ (we often use the term *IC-agents*, or simply an *agent*, when the law under which it operates is assumed to be known, or if it is of no immediate concern); and (c) the concept of *law-based trust*, or *L-trust* for short, induced by an IC mechanism.

This section is organized as follows. After introducing the concepts of law and of agent in the following two sections, we discuss in Section 2.3 the dual mediation mechanism between any pair of IC-agent, implied by this model. We then illustrate these concepts in Section 2.4 via an example. In Section 2.5 we introduce the concept of L-trust, and show how it is used for the enforcement of IC laws. Finally, in Section 2.6 we discuss possible techniques for establishing L-trust.

2.1 The Concept of Law

The function of a law $\mathcal{L}$ under this model is to decide what should be done in response to the occurrence of certain events at any agent x that operates under this law (i.e., at any $\mathcal{L}$-agent). The events in question belong to a predefined set E of what we call *regulated events*, which may occur at any IC-agent. For any such event e occurring at x (called the *home* of this event) the law mandates a response to be carried out locally at x. This mandated response, called the *ruling* of the law, is a sequence of zero or more operations, which are members of a predefined set O of what we call *control operations*. This ruling may depend on the event e that triggered it, as well as on what we call the *control-state* (or simple *state*) of the home x of e.

The sets E and O, and the structure of the control-state, are features of the IC model, and are therefore the same for all laws, and for all IC-agents. These concepts are defined abstractly below, and they need to be reified, and can be extended, by any IC mechanism based on this model.

The Set E of *Regulated Events*: This is the set of events that may occur at any agent and whose disposition is subject to the law under which this agent operates. And every event is associated with the local time of its occurrence. This set contains the following three types of events, described here with little, if any, detail of their structures: (1) The `arrived` event, which represents the arrival of a message at a given agent, and contains a identifier of the law under which this messages has been sent. (2) The `sent` event, which represents an attempt by the actor of a given agent to send a message (the distinction between an agent and its actor will be clarified later). And (3) the `birth` event, which is the very first regulated event in the lifetime of every agent, signaling its moment of formation.

The *Control-State* of an Agent: The control-state CS_x is the state maintained by agent x—it is distinct from the internal state of the actor of this agent, of which this model is oblivious. The control-state, or simply "state," is an unbounded bag of terms whose structure is not specified by this model (As we shall see, this state, which is initially empty, can change dynamically in response to the various events that occur at it, subject to the law under which this agent operates.)

The Set O of *Control Operations*: These are the operations that can be included in the rulings of laws. The set O contains the following two subsets: the set O_s of *state-operations*, and the set and O_c of *communication-operations*. O_s, contains operations that change the state of the home agent, by adding and removing terms from it. And O_c contain the following operations, which affect the exchange of messages between the home agent and its peers: (1) the `forward` operation, which forwards a message to another agent; and (b) the `deliver` operation, which delivers a message to the actor of the home agent—in effect, enabling this actor to read this message.

2.1.1 A Definition of a Law:

We are now in a position to give a more formal definition of the concept of law, as follows.

Definition 1 (law) *Given the sets E and O, as defined above, and a set S of all possible* *control-states of an agent, a law $\mathcal{L}$ is a function:*

$$\mathcal{L} : E \times S \rightarrow O^* \tag{1}$$

In other words, a law maps every possible $(event, state)$ pair into a sequence of zero or more control operations, which constitute the *ruling* of the law. Several aspects of this concept of law are worth pointing out here.

1: There is an interplay between the fixed law that governs a given agent, and its dynamically changing state: on one hand, the ruling of the law may depend on the state, and on the other hand the content and evolution of the state is regulated by the law. This interplay is an important characteristic of the IC model, which, as we shall see in Section 3, is instrumental in satisfying Principles 1 and 3.

2: *A law is local* to every agent x it governs, in that the ruling of the law is defined over the local events and state at x, and it can mandate only operations that can be carried out locally at x. Thus the law that governs an agent x can be complied with, or enforced, locally at x, without any access to other agents, and without any knowledge of the coincidental state of others—as required by Principle 2. It is this locality which makes the IC-model decentralized.

Moreover, as we shall see in Section 3, the locality of the law facilitates flexible interoperability between agents subject to different laws, as required by Principle 4; and it causes no loss of generality, as required by Principle 3.

3: A law cannot be inconsistent[5]; that is, its ruling is never ambiguous. Indeed, by its definition, a law maps any given $(event, state)$ pair to a *single* ruling. This is worth mentioning, in particular, because many conventional policy mechanisms admit policies with inconsistencies, which needs to be resolved, somehow. The main reasons for inconsistencies under these mechanisms, is the wish to admit independent contributions to a single policy by different stakeholders. But this need can be satisfied under IC, without introducing inconsistencies, by extending the IC model with the concept of *law-hierarchy*, (cf. [2]).

4: Finally, note that this abstract definition of the concept of law does not specify a language for writing laws (which we call a law-language). This, for several reasons. First, because despite the pragmatic importance of choosing an appropriate law-language, this

[5]This does not mean, of course, that a law cannot be ill formed, or incorrect, in the sense that it does not function as its writer intended.

choice has no impact on the semantics of the model itself, as long as the chosen language is sufficiently powerful to specify all possible functions of the form defined above. Second, by not specifying a law-language one provides the freedom to employ different law-languages for different applications domains, possibly under the same mechanism. Finally, leaving the language out of this model simplifies it, and facilitates the analysis of its underlying nature. Note, however, that for the sake of the examples used in this paper we will introduce an informal pseudo code for such a language.

2.2 The Concept of an $\mathcal{L}$-agent

We have broadly defined an $\mathcal{L}$-agent as an actor engaged in the sending and receipt of messages, in compliance with law $\mathcal{L}$. Here we elaborate on this definition. First, by introducing an abstract model of an $\mathcal{L}$-agent, second by defining the meaning of compliance with a given law, and concluding with a brief discussion of cross-interaction between agents operating under different laws.

A Model of an $\mathcal{L}$-Agents We define an $\mathcal{L}$-agent x to be a pair $x = \langle c_x, T_x^{\mathcal{L}} \rangle$, where c_x is an *actor*, and $T_x^{\mathcal{L}}$ is a device that mediates the interactions of c_x with others, in compliance with law $\mathcal{L}$. The mediator $T_x^{\mathcal{L}}$ is called the *private controller* (or simply the controller) of agent x, and the actor c_x, whose internal structure is irrelevant to this model, is said to *animate* agent x. (Figure 1 depict two such agents, operating under possibly different laws, and interacting with each other.)

Note that this abstract model does not specify the "geographical" relationship between an actor and the controller that mediates its interaction with others. Indeed, as we shall see in Section 2.6, different realizations of this model may have the controller either running on a different host than the actor it serves, placed anywhere on the Internet, or running on the host of the actor, or even be part of the actor itself. Also, we point out that the controller $T_x^{\mathcal{L}}$ should not be viewed as a *wrapper* of actor c_x. In particular, because we do not rule out the possibility that an actor animates, concurrently, several agents, possibly under different laws; and may, at the same time, communicate in a manner not regulated under IC.

Now, it is the controller $T_x^{\mathcal{L}}$ which is the locus of the *regulated events* at agent x. In particular, an `arrived` event occurs at controller $T_x^{\mathcal{L}}$ when a message addressed to x arrives at $T_x^{\mathcal{L}}$. Similarly, a *sent* event occurs at $T_x^{\mathcal{L}}$ when a message sent by its actor c_x arrives at it, on its way to its target. It is also the controller $T_x^{\mathcal{L}}$ that maintain the *control-state* of agent

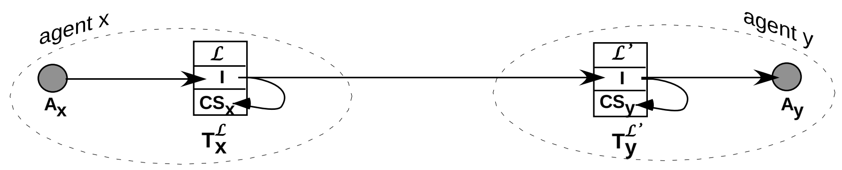

Figure 1: *A pair of interacting agents, operating under possible different laws*

x, and carries out the ruling of the law.

Structurally, a controller can be described as a triple $T_x^{\mathcal{L}} = \langle \mathcal{L}, I, CS_x \rangle$, where $\mathcal{L}$ is the law under which this particular controller operates; CS_x is the *control-state* of agent x; and I is a generic mechanism that complies with any given law $\mathcal{L}$ by interpreting this law, and carrying out its rulings, in the manner defined below.

Compliance with a Law: We say that an agent x complies with a given law $\mathcal{L}$ if the following two conditions are satisfied: (1) The ruling of law is carried out[6], *atomically*, in response to the occurrence of every regulated event at x, and in the order of the occurrence of these events; while events that occur simultaneously are handled sequentially, in arbitrary order. And (2) control operations are *not* carried out at x unless mandated by its law—thus, in particular, no messages would be forwarded from x to anybody, without it being mandated by the law.

$\mathcal{L}$-Messages, and Cross-Interaction: By definition, messages sent by an $\mathcal{L}$-agent comply with law $\mathcal{L}$—they are thus called $\mathcal{L}$-messages. But an $\mathcal{L}$-agent may be permitted, by its law $\mathcal{L}$, to receive $\mathcal{L}$'-messages—that is, messages sent by an $\mathcal{L}$'-agent—where law $\mathcal{L}$' is different from $\mathcal{L}$. This amounts to *cross-interaction* (or "interoperation") between agents operating under different laws. Moreover, an $\mathcal{L}$-agent may be permitted, by its law $\mathcal{L}$, to receive even *unregulated messages*, that is, messages send by an actor not regulated by an IC mechanism[7].

2.3 The Dual Mediation of Communication Under the IC model

One of the significant aspects of the IC model is that it involves duel mediation of every exchange of messages between IC-agents: one on the side of the sender of a message, and one on the side of its receiver. Specifically, the passage of a message from an actor c_x of

[6]If the ruling of the law for a given event is empty, then nothing extra will be done in response to this event; in other words, this event is effectively ignored by the law.

[7]For simplicity, we mostly ignore in this paper the latter possibility, which is quite important in practice.

an $\mathcal{L}$-agent x to an actor c_y of an $\mathcal{L}'$-agent y, must be mediated first by the controller $T_x^{\mathcal{L}}$ associated with c_x, and then by the controller $T_y^{\mathcal{L}'}$, associated with c_y, as is illustrated in Figure 1. This is a direct consequence of the locality of IC laws, which requires both the sender and receiver to individually comply with the law under which each of them operates. This duel mediation is in contrast with the conventional AC mechanisms, which use a single *reference monitor* to mediate the exchange of messages. Such a reference monitor is usually placed at the server side, or is used as a central *policy decision point* (PDP) for an entire system, as under XACML [13].

The dual mediation under IC has several important implications, not the least of them is that it facilitates interoperability by providing flexible control over cross-interaction between agents operating under different laws, as is farther discussed in Section 3.5. Moreover, as has been shown in [22], the duel control turns out to be more efficient than centralized control, in many circumstances. A simple illustration of the nature of dual mediation, and some of its consequences, is provided by the following example law.

2.4 Budgetary Control Over Messaging—an Example:

We introduce here a rather synthetic law that budgets messaging activities. More specifically, this law, called $\mathcal{BC}$, for "budgetary control," limits any actor that operates under it to sending at most 1000 messages, and to receiving no more than 2000 messages.

This, and other example laws in this paper, is written here in an informal pseudo-code, which is broadly modeled after the logic-based law-language used by the LGI mechanism [20]. We first introduce this pseudo code, and then law $\mathcal{BC}$ written in it.

An Informal Law-Language: A law written in this language consists of a sequence of *event-condition-action* rules of the following form:

```
    UPON e IF c DO [o],
```

where `e` is a regulated event (or a pattern that could match several such events); `c` is an optional condition defined over the event e, and over the state (CS) of the *home agent*; and `[o]` is the sequence of control-operations that constitute the ruling of the law (i.e., the "action" part of the rule).

Whenever some event e' occurs at some agent x, the ruling of its law is computed by the controller of this agent in the following way: The sequences of rules that constitute the law are evaluated from top to bottom, until a rule is found whose e part matches event e', and whose condition c is satisfied. The action $[o]$ of this successful rule is then defined as the ruling of the law. If no rule succeeds for a given event and state, then the ruling of the law for this event is defined as empty, causing this event to be simply ignored.

Here are some additional details about this informal law-language: (1) the control-state of the home agent can be checked via a predicate of the form $\exists$`t`, which means "there exists a term `t` in the control state of the home agent"; (2) capitalized symbols represent variables that can match any string; "_" represents a don't-care symbol; and (3) the phrase `+t` is a control operation that adds the term `t` to the state, and `-t` remove such term from the state. These details, along with the comments provided with each example law in this paper should be sufficient for the broad understanding of these laws. Finally, all our examples in this paper deal with groups of agents that operate under a common law, so we do not show any syntax for allowing interoperation between different laws.

Law $\mathcal{BC}$: This law, displayed in Figure 2, consists of three rules, whose effect is described below. Rule $\mathcal{R}1$ of this law is triggered by the `birth` event—the first event in the lifetime of every agent—and its ruling is to add the terms `sBudget(1000)` and `rBudget(2000)` to the *CS* of the newly created agent. As we shall see, these terms represent the upper bounds on the number of messages that every $\mathcal{BC}$-agent is allowed to send and receive, respectively.

Rule $\mathcal{R}2$ is triggered by the sending of any message, with the following result: If the message sender has in its *CS* the term `sBudget(B)` with a positive `B`—representing its current sending-budget—this budget would be decremented by 1, and the message would be forwarded to its destination. But if `B` is 0 then the sender would get a "message blocked" response, and its message would not be forwarded. Finally, Rule $\mathcal{R}3$ provides the analogous treatment of the receipt of messages, which is limited by the term `rBudget(B)`, representing the current budget of every $\mathcal{BC}$-agent for receiving messages.

Note the importance of the dual mediation under IC, which allows separate control over the sending of messages and over their receipt. We will make additional observations about this law in due course.

2.5 Law Enforcement, and the Concept of Law-Based Trust

By "enforcement" of laws over the communication between distributed actors we do not mean that the actors in question are forced to observe any particular law—indeed it is virtually impossible to force a set of disparate actors, dispersed throughout the Internet, to

```
R1. UPON birth DO [+(sBudget(1000)), +(rBudget(2000))]

R2. UPON sent(_,_)
            IF ∃sBudget(B) AND (B > 0) DO [decr(sBudget(B)), forward]
            ELSE DO [deliver(‘‘message blocked’’)]

R3. UPON arrived(_,_)
            IF ∃rBudget(B) AND (B > 0) DO [decr(rBudget(B)), forward]
            ELSE DO [deliver(‘‘message blocked’’)]
```

Figure 2: Law $\mathcal{BC}$ of Budgetary Control

conform to any given constraint, beyond those implicit in the standard Internet protocols—although it is possible to do so over an intranet, as has been demonstrated in [14]. This section describes what we mean by law-enforcement under the IC model. We start with the definition of a concept of trust, which we call *law-based trust*, or L-trust, for short. In Section 2.5.1 we introduce an example that illustrates some implications of such trust. In Section 2.5.2 we define our concept of law enforcement, and compare it to the conventional view of enforcement of access control policies.

Definition 2 (law-based trust) *Consider an IC-agent y, and an actor c (which may or may not operate as an IC-agent). We say that c* has a law-based trust in *y if and only if the following two conditions are satisfied when communicating with y (i.e., getting a message from y, or attempting to send a message to it): (1) c recognizes that y is an IC-agent, and (2) c identifies the law under which y operates. Also, we say that there is a* mutual L-trust *between two IC-agents if and only if the actor of each of them has an L-trust in the other*[8].

Of course, the concept of trust over the Internet—as in “trust management” or in “trusted computing base”—is notoriously ambiguous, and somewhat shaky. This is true for our concept of L-trust as well. Nevertheless trust is a necessary concept for the Internet, but it must be carefully justified to be meaningful and dependable. In Section 2.6 we will discuss briefly means for justifying L-trust. Here we turn to the implication of this kind of trust, assuming that it is justified.

[8]We sometimes use the term law-based trust whether it is mutual or not, expecting the ambiguity to be resolved by the context.

The Consequences of L-Trust, and the The Concept of $\mathcal{L}$-Community: The basic consequence of L-trust can be stated as follows. Consider an agent x that operates under law $\mathcal{L}$, sending a message to an agent y operating under law $\mathcal{L}$', which may or may not be equal to $\mathcal{L}$. The existence of mutual L-trust between x and y has the following consequences: (a) x can be confident that the effect of the arrival of this message at y would be in compliance with law $\mathcal{L}$' under which y operates; and (b) y can be confident that the message it got from x has been sent in compliance with the law $\mathcal{L}$, under which x operates.

L-trust is particularly significant for a set of agents operating under a common law $\mathcal{L}$, which we call an *$\mathcal{L}$-community*. Such a common law could enable the members of an $\mathcal{L}$-community to collaborate harmoniously and to compete safely with each other—depending on the law that governs them all—even with no knowledge of the nature and intention of each other. This is analogous to the manner in which social laws can create harmonious societies. We will illustrate this phenomena via the following example.

2.5.1 Making Electronic Theater Tickets Trustworthy—an Example

Consider a theater called *globe* that issues electronic tickets of the form `ticket(d)`, where d represents the date of a performance (assuming there is just one performance a day, and that the number of such tickets issued is equal to the number of seats). The theater then transfers these tickets to its clients, which may be merchants or theater goers. And clients can transfer these tickets to each other any number of times. Finally, the theater admits a client only in exchange of a ticket transferred to it. And suppose that all these transfers of tickets are to be done electronically.

This scheme can work well—that is, the holder of a ticket can depend on getting a seat in this theater in the specified date—if these tickets cannot be forged. That is, if these tickets can be created only by the *globe* theater, and that once created, they can be transfered, but not copied or modified. We now show how these assurances are provided by a law called $\mathcal{TU}$, for "ticket unforgeability."[9]. This law, displayed in Figure 3, consists of three rules discuss below.

Rule $\mathcal{R}1$ enable the theater *globe* (note that the symbol `Self` denotes a variable bound to the id of the home agent) to create any number of tickets in its own control state, simply by sending the message `createTiket(d)`, with any argument `d`. Indeed, the sending of such

[9]This law is simplified, in particular, in that it deals only with the transfer of tickets, but does not handle the payment that often (but not always) accompanies such a transfer.

a message by *globe* triggers this rule, whose ruling is to add the term `ticket(d)` to the *CS* of the sender, i.e., of the *globe* theater. (Note that his law has no rule to match the `birth` event, which mean that no control operation is carried out when a $\mathcal{TU}$-agent is being created.)

```
R1. UPON sent(createTicket(D),_) IF (Self=globe) DO [+ticket(D)]

R2. UPON sent(ticket(D)),_) IF (∃ticket(D)) DO [-ticket(D), forward]
            ELSE DO [deliver(‘‘illegal message’’)]

R3. UPON arrived(_, ticket(D)) DO [+ticket(D), deliver]
```

Figure 3: Law $\mathcal{TU}$ of Ticket Unfogeability

The other two rules enable any $\mathcal{TU}$-agent to transfer a ticket it has to any other $\mathcal{TU}$-agent[10], simply by sending it a message `ticket(d)`. Specifically, Rule $\mathcal{R}2$, which is triggered by the sending of a message `ticket(d)`, has two possible rulings: If the sender has the term `ticket(d)` in its *CS*, then the ruling would be to remove this term from the CS, and to forward this message to its address. Otherwise, the ruling for this event would be to deliver the message `‘‘illegal message’’` to the sender, and do nothing else.

Finally, Rule $\mathcal{R}3$, which is triggered by the arrival of a message of the form `ticket(d)` at any $\mathcal{TU}$-agent, has two operations in its ruling: first, to add the term `ticket(d)` to the *CS* of the home agent (i.e., the agent that received this message), and second to deliver the message itself to the actor of the home agent, to alert it about the arrival of this ticket. (For simplicity, we are not providing here the needed support for the case that the agent receiving such a message is the theater. This would require, in particular, a rejection of a ticket if it is for a wrong date.)

Discussion: The presumed L-trust among the member of this community induces a kind of *communal trust* of the following nature: When a theater *globe* transfers a ticket to some $\mathcal{TU}$-agent, it can be confident that whatever happened to this ticket, there can be no more than one copy of it, anywhere among the members of the $\mathcal{TU}$-community. Correspondingly, when a member of this community receives a ticket from another member, it (or she or he) can be confident that it got the only existing copy of this ticket—regardless of whom did

[10]We assume that no interoperability is permitted by this law, and by the other example laws in this paper.

it get the ticket from—and thus be confidant to be admitted to the theater in the specified date.

This communal trust is particularly remarkable, because the membership of the $\mathcal{TU}$-community is quite open and indefinite. Indeed the $\mathcal{TU}$ law imposes no restriction on adopting this law, thus joining this community. Therefore, anybody can join the $\mathcal{TU}$-community at anytime, and there is no way to tell who belongs to it at any given moment in time. (However, as shown in [20], it is quite possible to write laws that can regulate community membership in various ways.) We will consider other aspect of this law below, and in Section 3.

2.5.2 Virtual Enforcement of IC Laws via L-Trust

As has already been pointed out, the IC model does not coerce any actor to exchange $\mathcal{L}$-messages, under any specific law $\mathcal{L}$, or to engage in IC-regulated interaction in any other way. Such an engagement is purely voluntary.

Nevertheless, due to L-trust, an actor may often be *virtually compelled* to operate under a particular law $\mathcal{L}$. Broadly, this is the case when one wishes to use services provided only under this law. For example—returning to law $\mathcal{TU}$ above—consider an actor c who wishes to get a ticket for the *globe* theater. He or she can get such a ticket, which would be acceptable by the theater, only from he theater itself, which operates under law $\mathcal{TU}$, or from some other $\mathcal{TU}$-agent. In either case c would have to operate under law $\mathcal{TU}$ to get such a ticket.

Due to this compelling effect of L-trust, we occasionally refer to a mechanism that supports L-trust as a "law enforcement" mechanism. Such, somewhat loose, usage of the term "enforcement" is quite common in the access-control literature. For example, the XACML mechanism [13] relies for its "enforcement" on each server within its purview to voluntarily operate via a *policy enforcement point* (PEP). This PEP is expected to consult a *policy decision point* (PDP) regarding every request sent to it, and then to carry out the decision of the PDP. But no means are provided for forcing servers to operate via such PEPs, and no such enforcement is possible in an open system. In fact, the level of enforcement under an IC mechanism is stronger than in conventional access control, due to the existence of L-trust.

2.6 Establishing Law-Based Trust

We start by considering the conditions for a pair of IC-agents to have mutual L-trust in each other, according to Definition 2 of such trust. This is followed by a discussion of how these conditions can be satisfied for large numbers of IC-agents.

Consider a pair $x = \langle c_x, T_x^{\mathcal{L}} \rangle$ and $y = \langle c_y, T_y^{\mathcal{L}'} \rangle$ of IC agents (see Figure 1), exchanging messages under laws $\mathcal{L}$ and $\mathcal{L}'$ respectively—where these laws may be different or equal. The definition of Mutual L-trust between such agents requires the actors c_x and c_y to have L-trust in their counterpart agents. We maintain that such trust would exist if the two actors can trust the pair of controllers $T_x^{\mathcal{L}}$ and $T_y^{\mathcal{L}'}$ to satisfy the following three conditions: (a) each of the controllers operates in compliance with its law, according to the definition of compliance in Section 2.2; (b) the controllers identify to each other the laws under which they operate, via the *one way hash* of these laws (this is sufficient identification if the text of these laws is made public in some well known repository); and (c) the messages exchanged between the controllers, and between them and the actors they serve, are transmitted securely.

A pair of controllers can be thus trusted by the actors that employ them, if they are maintained by a trusted third party (TTP) acceptable to both actors, and if this third party certifies that they satisfy the above three requirements. This assertion is based on a well established tradition in computer research and practice.

Note that it is reasonable to require that the controllers operate on different hosts than the actors served by them, lest they can be corrupted by the actors. However, it may be possible for controllers to reside on the same host with the actors they serve, if the controllers are implemented over TPM [5].

Moreover, to support large numbers of IC agents, which may operate under a variety of laws, one need to provide a large set of *generic* controllers that can be widely trusted to operate in compliance with any valid law loaded into them. Such a collection of controllers would serve the role of a *trusted computing base* (TCB) for IC-based communication. But unlike the traditional TCB, which is usually centralized, our collection of controller is designed to be decentralized, and is thus referred to as "decentralized TCB", or DTCB. Such a DTCB needs to be provided by a reliable service, called a *controller service* (CoS), which creates, maintains, continuously tests, and certifies a distributed collection of controller.

Such a CoS can be operated by an enterprise for its own internal use; by a federation of enterprises for the use of its members; or by a regional authority, such as a municipality, for the use of actors in its range. The current implementation of LGI provides an experimental version of a CoS. And there is an ongoing research on techniques for protecting the controllers maintained by a CoS from various kinds of attacks; in particular, by intrusion detection, by

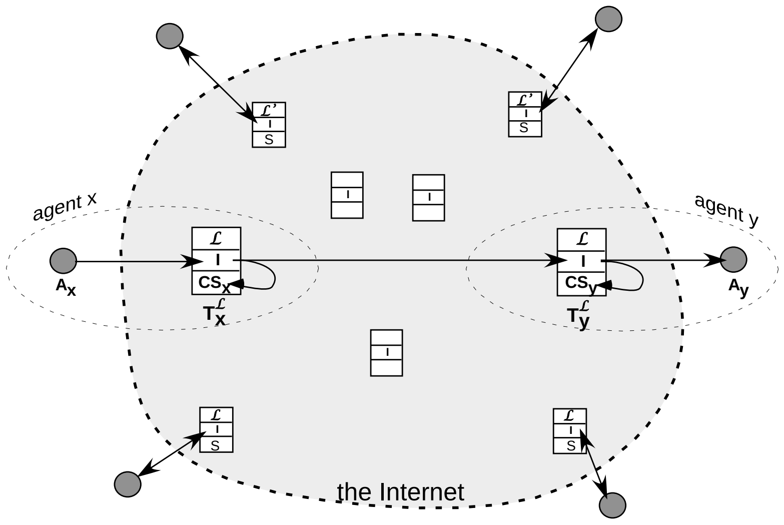

Figure 4: *IC-Based Communication: Actors are depicted by circles, interacting across the internet (lightly shaded cloud) via their controllers (boxes) operating under law L. Agents are depicted by dashed ovals that enclose (actor, controller) pairs. Thin arrows represent messages, and thick arrows represent modification of state.*

making it harder to target specific actors, or specific law; and by other means.

It should be pointed out that if an IC mechanism is to be used by arbitrary actors all over the Internet, then the CoS needs to be managed by a reputable commercial company or governmental institution, whose business it is to provide its customers with trustworthy controllers. This organization must be willing to vouch for the trustworthiness of its controllers, and to assume liability for their failures. Such an Internet wide CoS is yet to be established.

Selective Decentralization of Mediation: Although the mediation of inter-agent interaction under IC is decentralized in principle—each actor operating via its own "private" *controller*—it lends itself, in practice, to a whole range of partially centralized implementations, which can be adapted to different applications.

This is due to the fact that private controllers can be hosted, by the so called *controller pool*, each of which capable of hosting a large number[11] of controllers, thus serving several

[11]Under the current implementation of LGI, for example, each controller pool can host several thousand of private controllers, depending on the complexity of the law, and on the nature of the application at hand.

different agents, possibly subject to different laws. The above mentioned CoS is assumed to a distributed collection of such controller pools.

The availability of controller pools allows one to run any given sub-group of controllers on a single *controller-pool*, thus creating a whole spectrum of possible implementations of mediation—anything between complete decentralization, where each controller of a given community operates on a separate controller-pool, to a complete centralization, where all controllers are hosted by a single controller pool. The potential for performance optimization inherent in such selective centralization of mediation is yet to be fully explored; for some initial ideas about this matter see [20].

3 Analysis of the IC model

The purpose of this analysis is to identify the main characteristics of the IC model, and to explore the manner in which it satisfies the IC principles we proposed. We will do this, in part, by comparing this model with the traditional access control, attempting to highlight the main structural differences between the two. This comparison is important because AC is the predominant approach to regulation of distributed systems. But although access control is likely to be familiar to most readers, a few comments about it are worth making before we start.

The root of the concept of access control is a model formulated more than thirty years ago, to regulate the use of resources by processes residing in a single host. But AC technology has come a long way from this root, by developing various means for dealing with distributed systems. These include cryptographic authentication, the *public-key infrastructures* (PKIs), *trust management* [10], *delegation certificates* [6], and *roles* [26]. Nonetheless, access control still retain some key aspects of its origin. First, by and large[12], current AC mechanisms share the rather limited objective of the original AC model, which is, broadly speaking, *to permit access only to those that have the right for it*, but without any concern about the dynamic behavior of those that get such access. This is analogous to having traffic laws that require only that every driver has a driving license, without any concern about the dynamic process of driving. Second, policies are still enforced via virtually centralized[13] *reference monitor*. And third, current AC mechanism are strongly influenced by the original

[12]With some exceptions to be discussed in Section 3.1.

[13]"Virtually," because the reference monitors is often replicated.

dichotomy of mandatory and discretionary (MAC and DAC) policies, which, as we shall see, is unwarranted and even harmful. These aspects of conventional AC figure prominently in the differences between it and the IC model, and are responsible for the fact that the IC principles are largely unsatisfied by AC mechanisms.

We start our analysis in Section 3.1, by describing the manner in which scalable sensitivity to the history of interaction is achieved under IC. In Section 3.2 we discuss the global sway that a law can have over the community governed by it—this despite the inherently local nature of laws; and, more generally we argue that the locality of laws under IC does not reduce their expressive power. In Section 3.3 we show that the IC model unifies the DAC and MAC dichotomy of traditional access control; and in Section 3.4 we show that this unification is instrumental in satisfying the principle of separation of policy from mechanism. In Section 3.5 we show how decentralization facilitates flexible, composition-free, interoperation. And we conclude in Section 3.6 by showing that despite appearances to the contrary, IC laws can be sensitive to the context in which individual agents operate.

3.1 Scalable Sensitivity to the History of Interaction

This section discusses the the manner in which the IC model manages to be sensitive to the history of interaction, as required by our Principle 1, and the nature of the scalability of this sensitivity. We conclude by considering the treatment of these issues under conventional AC.

Generally speaking, history sensitivity is accomplished under IC not by maintaining the complete history of interaction, or even a selected parts of this history, in order to be examined when needed. But by maintaining, in the state of each agent x, a relevant *function of the local history* of the interaction of x with other agents. (Note that it is possible, under the IC model, to maintain the entire history of interaction, if necessary, but it is rarely if ever necessary, and it would be very inefficient and unscalable, as we shall see below.)

A simple example of this technique is provided by the budgetary control law $\mathcal{BC}$ iintroduced in Section 2.4. The purpose of this law is to impose bounds on the number of messages that each agent can send and receives. This is done by (a) initializing the state of each agents with budgets for sending and receiving messages, (b) maintaining the balances of these budgets while each agent exchanges messages, and (c) blocking messages when the corresponding balances are zero. These dynamically changing budgets are the functions of the history of interaction maintained by the law in the state of every $\mathcal{BC}$-agent.

What makes this technique possible under IC are mainly the following two properties of this model. The first is the fact that the ruling of the law is not restricted to accepting messages or blocking them, but that it can mandate arbitrary change of the local state of the home agent, thus enabling the computation of any desired function of the history of interaction—like the balances of the messaging budgets under law $\mathcal{BC}$. The second property of IC that is critical to this technique is, of course, the fact that the law can be made sensitive to the state of each agent. The lack of one, or both, of these properties in most conventional AC mechanism, precludes them from providing reasonably scalable sensitivity to history, as we shall see in Section 3.1.2.

3.1.1 On the Scalability of History Sensitivity Under IC

We consider here two quite distinct senses of of the term "scalability": (1) scalability with respect to the length of the process of interaction, and thus length of the history; and (2) scalability with respect to the number of interacting agents, and the volume of messages exchanged between them.

Law $\mathcal{BC}$ is scalable in both of these senses. It is scalable with respect to sense (1) because the incremental computation of the balances of the two budget is essentially fixed throughout the interaction history. And it is scalable with respect to sense (2) because the budgetary constraint is defined over the *local* history at each agent. Therefore, this constraints over the behavior of a given agent x depends only on the messaging activities of x.

Of course, not every history-sensitive constraint can be implemented under IC in such a scalable manner. One can imagine a non-local constraint, such as the requirements that the total number of messages sent by all members of the community in question does not exceed a certain limit. As argued in Section 3.2, all such constraints can be handled under IC. But the scalability of enforcing such constraints may vary. It is worth noting, however, that many kinds of important history-sensitive constraints are implementable very scalably under IC. This includes, in particular, the well known *Chinese wall policy* and *dynamic separation of duties* policies, as has been shown in [3].

3.1.2 Sensitivity to History Under Access Control

In a very limited sense, history sensitivity has been part of AC from its inception. Because the distribution of rights among the various subjects, which can be changed by *delegation*, can be viewed as a state of a policy. Moreover, several AC researchers noticed early on that there is a need for additional, and very specific, forms of sensitivity to history, including the so called “Chinese wall” policies and *dynamic separation of duties* policies [26]. But, the first comprehensive attempt at such sensitivity under AC seems to be due to Jojodia et al., in their 2001 paper [15]—a decade after such sensitivity was introduced, essentially under IC, in [19]. But this approach does not scale as it maintains essentially the entire history of interaction in its global set of rules.

The SPL mechanism introduced by Ribeiro et al. [25]. is more sophisticated in this respect. It provides an automatic “compilation algorithm” that reduces the amount of historical information to be saved, and attempts to optimize the way that information is to be queried for a given constraint to be evaluated over the history. The technique is clearly more scalable than that of the Jajodia model, but it is not scalable enough. In particular, in order to implement under SPL our simple example of messaging budget, with initial budget of `N`, one will have to keep at least part of the history of up to `N` messages—where `N` is the initial messaging budget—and this history will have to be kept in a central place, which makes it quite unscalable for large values of `N`.

The underlying reasons for this lack of scalability of SPL, and of the Jajodia’s model, are that both of them retain the traditional AC model, with its centralized enforcement mechanism; and under which a policy can only rule to permit or deny the transfer of a message, but cannot also change any state, as is done under law $\mathcal{BC}$ above.

3.2 On the Global Sway and Expressive Power of IC Laws

The local nature of IC laws raises the following questions: (a) can such laws establish global properties of a system consisting of many agents; and (b) is this ability to establish global properties as general as that of laws that are not restricted to be local? The answer to both of these questions turns out to be affirmative. We will address this issue here for an $\mathcal{L}$-community, that is, for a set of agents that operate under a common law $\mathcal{L}$. But our conclusions are valid more generally, as has been shown in [2].

We start with some terminology. We will refer to a global property of a multi-agent system as *regularity*—employing the following dictionary[14] definition of the term “regularity”: “conformance [by every part of the system] to a given rule.” And we distinguish between

[14]See the Random House Dictionary, for example

two kinds of regularities of a given multi-agent system S, which we call *local* and *non-local*. A local regularity of system S can be expressed as a universally quantified formula of the form $\forall x \in S\ P(x)$, where $P(x)$ is a local property of agent x; that is, $P(x)$ is defined over the local state of x, and over local events at it. For example, $P(x)$ can be a given limit of the number of messages that an agent x can send (as established by law $\mathcal{BC}$ of Section 2.4).

A non-local regularity is one which cannot be expressed in this manner. For example, consider the following statement: *"the aggregate of all messages sent by all members of system S does not exceed 1000."* This is a regularity in that it restricts the behavior of each member of S—in particular, no agent is allowed to send the 1001th communal message. And this regularity is non-local in that this constraints on message sending depends on the coincidental state and behavior of all members of S.

Now, local regularities are the direct consequences of the IC model, due to the fact that all members of an $\mathcal{L}$-community conform to the same law $\mathcal{L}$. This has been demonstrated by our example laws $\mathcal{BC}$ in Section 2.4, and $\mathcal{TU}$ in Section 2.5.1.

The case of non-local regularities is less obvious. Clearly, non-local regularities can be established by a non-local law enforced via a central reference monitor that mediate all messaging activities of members of S, and can maintain the global control state of the entire system (like the number of messages sent so far by all members of S.) The question is: can non-local regularities be established via the local IC laws?

The affirmative answer to this question has been provided in [22], via the following result:

> *Any non-local constraint on interaction that can be implemented via a central reference monitor can be* localized*, and thus implemented via a local IC law.*

This important result—whose proof is quite straightforward, but too long to be repeated here—is due to the ability of IC laws to mandate the movement of relevant information from one agent to another, getting such information to the appropriate decision points.

This ability to localize non-local regularities means, essentially, that *the locality of IC laws does not reduce their expressive power.* Of course, such localization involves certain amount of aggregation of information, which may reduce its scalability. But we have found that in practice, many useful non-local constraints involve modest amounts of aggregation, which has only a marginal effect of scalability. For more about this matter, see [20].

In conclusion, we can state that the locality of laws does not impede their ability to establish global properties, and it does not reduce their expressive power.

3.3 The Unification of Mandatory and Discretionary Controls

We discuss here a critical structural aspect of the IC model: its unification of mandatory and discretionary aspects of governance, which are generally considered independent elements of access control. Indeed, the 1999 report “Trust in Cyberspace,” by the National Academy of Science, characterizes the conventional access control technology as follows ([27] page 114): *the two basic types of access control policies that have dominated the computer security work for over two and a half decades [are the] discretionary access control [DAC] and mandatory access control [MAC]* . DAC policies, this report continues to explain, are discretionary in the sense that a subject can pass (via delegation) its access permissions on to another subject; and MAC policies enforce access control on the basis of fixed set of rules mandated by a central authority. (Note that MAC is frequently identified with *multi level security* (MLS) policies, like that of Bell-Lapadula, but we adopt here the general interpretation of MAC). In other words, DAC and MAC are viewed as a dichotomy, i.e., as two independent, and distinct types of policies. And an AC mechanism may support one, or the other, or both—in particular RBAC is claimed [24] to support both, (although that paper takes the narrow interpretation of MAC, as an MLS policy).

We maintain that the MAC/DAC dichotomy is problematic from a theoretical viewpoint, and that it has an unfortunate practical consequence. The theoretical difficulty with this dichotomy is that every DAC policy gets its semantics from an underlying MAC policy. Consider for instance a *capability-based AC*—a common example of a “pure DAC policy”. The very meaning of capabilities, as well as their origin, delegation and revocation, are all defined by an essentially mandatory (MAC) policy built into the very fabric of the AC mechanism at hand, and are not subject to the discretion of the subjects. It is this *implicit nature* of the mandatory counterpart of the discretionary distribution of capabilities that makes it seem to be a pure DAC, which it is not.

The practical consequence with this dichotomy is that it encourages making the mandatory counterpart of what is viewed as a pure DAC policy implicit in the mechanism, as we have seen above. This reduces the flexibility and generality of AC mechanisms, because any change in this implicit mandatory policy—like a change of the delegation process—would involve changing the mechanism itself, or replacing it with another. We will return to this matter in the following section.

The IC model, on the other hand, is based on two complementary and inextricably intertwined elements: (a) the explicitly stated fixed *law*, which is mandatory for all members of the community in question; and (b) the *state* that can be changed at the discretion of the various actors, subject to the law. So, while the law and the state correspond roughly to the concepts of MAC and DAC, they cannot be viewed as a dichotomy, but are, rather, the elements of a unified governance mechanism. Indeed, the ruling of the law depends on the state; and the state changes in response to events that occur mostly at the discretion of some actors, but only according to the mandatory rules laid out by the law. They are, thus, unavoidably intertwined. To illustrate these points we introduce below an implementation under IC of a version of capability based access control.

3.3.1 Capability Based Access Control Under the IC Model—an Example

We introduce here a law, called $\mathcal{CB}$ (for "capability based"), which establishes the following, informally stated, provisions. First, under this law a capability is a term $cap(y, d)$, which, when contained in the CS (i.e., control state) of some agent x, signifies the right of x (its holder) to send messages of the form $msg(m)$ (with any content m) to agent y. Second, a capability $cap(y, d)$ is *delegateable* only if d is 1. And the delegation of a $cap(y, 1)$ capability to any agent z can be done by the holder of this capability, by sending to z the message $delegate(cap(y, d'))$, where d' can be either 1 or 0. The arrival of this message at z would provide z with a delegateable or an undelegateable version of this capability, depending on whether d' is 1 or 0. (Note that this is just one of many possible ways for doing delegation; we will return to the general topic of delegation later.) Finally, every new member x of the $\mathcal{CB}$-community is initialized with the delegateable capability $cap(x, 1)$ *for itself*—which according to the above, it would be able to delegate to any other member.

A law $\mathcal{CB}$ that specifies this policy under the IC model is displayed in Figure 5. This law consists of five rules. Rule $\mathcal{R}1$ is triggered by the *birth* event, and its ruling is to add to the state (CS) of the home agent a term `cap(Self,1)` representing a delegateable capability for itself. This means that every agent operating under this law would be initialized in this way.

Rule $\mathcal{R}2$ is triggered by the sending of any message of the form `msg(_)` to any agent `Y`. This rule produces the ruling to forward the sent message only if the condition $\exists$cap(Y,_) is satisfied; that is if the sender has a `cap` term for `Y` in its state. Otherwise, this rule would produce a ruling to deliver to the sender an "illegal message" warning. When a message sent under this rule arrives at its destination it is delivered without further ado by Rule $\mathcal{R}3$. Note that these two rules provide the `cap(...)` terms with the semantics of a *right to send*

```
R1. UPON birth DO [+cap(Self,1)]

R2. UPON sent(msg(_),Y)
          IF ∃cap(Y,_) DO [forward]
          ELSE DO [deliver(``illegal message'')]

R3. UPON arrived(_,msg(_)) DO [deliver]

R4. UPON sent(delegate(cap(Y,D)),_) IF (∃cap(Y,1)) DO [deliver]

R5. UPON arrived(_,delegate(cap(Y,D))) DO [+cap(Y,D), deliver]
```

Figure 5: Law $\mathcal{CB}$ of Capability Based Control

messages, i.e., the semantics of capabilities.

Rule $\mathcal{R}4$ is triggered by an agent sending a message `delegate(cap(Y,D))` to any agent Z. The ruling for such sending would be to forward this message, but only if the sender has the term `cap(Y,1)` in its CS, representing a delegateable capability. Finally, when this `delegate` message arrives at its destination Z, it would trigger Rule $\mathcal{R}5$ at Z, which would cause the sent capability to be added to the CS of the receiver.

Discussion: This example clearly demonstrates the manner in which mandatory and discretionary aspects of regulation are represented under IC by means of the law and the state, respectively; and the manner in which these two elements of IC are intertwined. It is law $\mathcal{CB}$ that defines the semantics of capabilities, their initial distribution, and the manner in which they can be delegated. But the eventual distribution of the capabilities in the state of the various agents is determined by the `delegate` messages sent at the discretion of the various actors, but subject to the law.

Moreover, the regime defined by law $\mathcal{CB}$ is only one of many types of capability-based regimes that can be implemented under IC. One can, in particular, replace $\mathcal{CB}$ with a law that employs a different initialization of the states of agents, and a different manner for delegating capabilities. The significance of this flexibility of the IC model is discussed in the following section.

3.4 Separation of Policy from Mechanism

As has already been pointed out, one of the original principles of secure computing has been that of *separation of policy from mechanism*, formulated in the early 1970s, by Wulf et al. [30], and by others. What this principle means, essentially, is that an AC mechanism should be as general as possible, without a bias toward any particular kind of policies, thus accommodating a wide range of them. But despite the obvious importance of this principle—particularly for complex systems that need to use multiple policies—it has not been satisfied to a significant degree by conventional AC mechanisms, (the importance of this principle has been recently noted in [11], but without proposing a significant support for it.) As we have seen in Section 3.3, this is arguably because of the failure of these mechanisms to make the mandatory aspects of a policy explicit.

The situation is very different under the IC model because its law represents mandatory constraints explicitly. The lack of bias of this model toward any particular type of policy is evident from our $\mathcal{CB}$ law. Although the IC model has no built-in concept of capability, this law creates a certain interpretation of these concepts, as it can easily create others. We now demonstrate the degree to which IC satisfies the principle of separation of policy from mechanism, via brief case studies of the treatment of two well known aspects of control—*delegation*, and *role*—under conventional AC mechanisms, and under IC.

3.4.1 Delegation—a Case Study

The basic concept of delegation is very simple: it is the giving to somebody else a right that one has for itself. But there are various possible ways for delegation to be carried out, and various kinds of constraints that one may want to impose on the process of delegation. For example, one may specify such things as who is permitted to delegate which privileges to whom, and under which circumstances; and how such delegation should affect the privileges of the delegator itself. We call a specific choice among such possibilities a *delegation-pattern.* and here is a sample of possible patterns of this kind: (1) The requirement that certain rights shall be delegated only *by transfer*; that is, the delegator will give up its own right, when delegating it to somebody else, as we have done with theater ticket under law $\mathcal{TU}$. (2) Some restriction on the number of times a given right may be delegated, if at all. (3) Making delegation of a privilege operate as loans, requiring the privilege to be returned within a specified time period. And (4) Establishing mechanisms for the revocation of delegated privileges.

Despite the need of different applications to employ different delegation patterns, most conventional AC mechanism employ a fairly simple, and quite rigid, built-in patter. According to [7], *"The standard approach to delegation is binary: either delegation is possible, and* *then no substantial further control over the way it is used is possible, or else no delegation is permitted."* Recently, several more sophisticated delegation patterns have been proposed. In particular, Bandmann et al. [7] proposed a model of *constraint delegation* which allows one to control the possible shapes of delegation chains. Also, Li et al. [17] permitting a fixed upper bound to be imposed on the depth of delegation chains. But each of these delegation patterns requires a new AC mechanism to be constructed—contrary to the principle of separation of policy from mechanism.

Under the IC model, on the other hand, the nature of delegation is specified by the law of a community, as it is the case with law $\mathcal{TU}$ of Section 2.5.1, and law $\mathcal{CB}$ of Section 3.3. And it is easy to see how other types of delegation patterns can be similarly defined. For a demonstration of how sophisticated delegation that combines several of the above mentioned patterns can be defined by a law under an IC mechanism, see [4, 1].

3.4.2 Role Base Access Control (RBAC)—a Case Study

An important extension of the traditional matrix-based AC model has been the introduction of the "role-based access control" (RBAC) model, by Sandhu and his colleagues [26]. This model views roles—which represent such things as the position that a given individual holds in a given organization—as a set of permissions, and defines an AC-policy essentially as a pair of mappings: a mapping of users to sets of roles, and a mapping of roles to sets of permissions. The RBAC model also provides for a hierarchical organization of roles, and includes certain constraints on the combinations of roles that a given individual can possess at one time.

The RBAC model has been widely adopted by the industry and been broadly accepted by the research community. But despite its considerable structural complexity (the official RBAC has four "reference models" of increasing sophistication) RBAC proved to be overly rigid, which gave rise to a host of *RBAC-based models* that generalized RBAC in various ways. These variants of RBAC include, but are not limited to: (1) parameterization of the roles of a subject by its other attributes [12]; (2) making authorization dependent on some dynamic relationship between the subject, and the object on which it operates [8]; (3) making the rights associated with a role dependent on the *context*, such as time and location [23]; and (4) providing for dynamic, and temporal, control over role-permission activation and deactivation, and over user-role assignment and activation (TRBAC [9] and GTRBAC [16]).

Now, all these models are worthy generalizations of the original RBAC. The problem is that each of them requires a different AC mechanism for its support—again, in a sharp breach of the principle of separation of policy from mechanism. Under IC, on the other hand, we have shown in [3] that the original RBAC model itself, and every one of the above mentioned extensions of it, can be defined simply by writing a suitable law. And, thus, they can all be realized via a single IC mechanism. And this, despite the fact that IC has no built-in concept of roles.

3.5 Interoperability

We define interoperability under IC as the ability of two agents operating under different laws to interact, without violating their own rules. Although this is analogous to interoperability under access control, the mechanisms of interoperability under these two control regimes is substantially different.

The conventional AC approach to interoperability [18] between parties operating under policies $P1$ and $P2$, has been to *compose* these policies into a single policy $P12$, which is, in some sense, consistent with both $P1$ and $P2$. The composition $P12$ is then to be fed into an appropriate reference monitor, which would mediate the interaction between the two parties. But composition of policies has several serious drawbacks: manual composition is laborious, and error prone; and automatic composition is computationally hard [18], and often impossible. Yet, composition is the natural, and perhaps necessary, approach to interoperability under AC—because AC employs a single reference monitor to mediate the interaction of any pair of agents.

Under the IC model, on the other hand, composition of laws is neither natural nor necessary. This is because IC employ duel mediation for every pairwise interaction. This enables an $\mathcal{L}1$-agent $x1$ to exchange messages with an $\mathcal{L}2$-agent $x2$, if such an exchange is permitted by both laws $\mathcal{L}1$ and $\mathcal{L}2$, and without the need to form a composition of $\mathcal{L}1$ and $\mathcal{L}2$. The basic mechanics for such interoperation is described in [28], where we have shown how it can be utilized for regulating B2B transactions.

Of course, for a law $\mathcal{L}1$ to permit the exchange of of some messages with agents operating under $\mathcal{L}2$, it—or, rather, its author—should examine the other law, or otherwise have some trust in it. This is not be necessary if the IC mechanism at hand supports the concept of *law hierarchy*, which is one of the possible extensions of the IC model, introduced by the

LGI mechanism [2]. In a nutshell, if laws $\mathcal{L}1$ and $\mathcal{L}2$ are both hierarchically *subordinate* to a common law $\mathcal{L}$, then they are guaranteed to conform to $\mathcal{L}$. This fact turns out to be sufficient for a very flexible and efficient composition-free regulation of dynamic coalitions of organizations (such as grids and virtual-organizations), as has been shown in [2].

3.6 Sensitivity to Context

The IC model seems to be inherently insensitive to anything but the history of the regulated events that occurred at each agent (which includes the history of interaction of a given agent with others). This is because the ruling of a law is a function of the pair $(event, state)$, and the state is itself some function of the history of local events. Nevertheless, we show here that IC laws can be written to be effectively sensitive to other things besides the interaction history. Specifically, we will discuss sensitivity to what we call: (a) physical context, and (b) inside information. (For a related approach to context sensitivity see [29].)

Sensitivity to Physical Context: By "physical context" we mean here attributes of the environment in which a given actors operates. These may, for example, be the temperature and pressure measured by the sensors associated with an actor; or the geographical location reported by its GPS. Of course, an IC mechanism *per se* is not aware of such attributes. But it can be made aware of them via the following simple means: (a) the actor in question sends a message, ostensibly to itself, but really meant to be picked up by its controller, reporting its location, say, as measured by its GPS; (b) the law at hand is written to pick up this message and to store the reported location in the control-state of this agent. Now, when the location is stored in the state of an agent it can be taken into account for computing the ruling of the law for various events occurring at that agent.

Of course, the resulting sensitivity to location, or to other physical attributes, is meaningful only if the self reporting of actors is trustworthy. But this is often the case in practice, either for specific actors, or even for entire community of them. As an example, consider a community of cars communicating electronically with each other, subject to some IC law. The GPS built into such cars, and the software that manages it, may have been produced by a single manufacturer, and their behavior may be known and trusted. Various sensor networks can be handled in a similar fashion.

Sensitivity to Inside Information: By "inside information" we mean information about some attributes of the *internal state* of an actor, such as the CPU utilization of its host, or its stage in whatever computing task it is involved in. Here, like in the case of physical context, we would rely on self reporting by the actor itself. And, again, such self reporting needs to be trustworthy to be relayed on by the law. Thus, a law of a certain community can incorporate information about the dynamically changing internal state of trusted actors, despite the fact that the IC model itself views actors as black boxes.

4 Conclusion

This paper is concerned about the lack of dependability, manageability and security of the large and open networked system, which increasingly serve critical institutions and infrastructure of our society. And it is our thesis that these problems can be addressed by subjecting the interaction between the components of such a system to governance via a mechanism that satisfies a set of what we have called IC principles.

We have introduced the an abstract reference model—called the IC model—of governance mechanisms that satisfy these principles. This model differs from conventional *access control* (AC)—the currently dominant approach to regulation of distributed systems—in its objectives, its underlying structure, and its resulting properties. The main objective of conventional AC is to permit access only to those that have the right for it, but without any concern about the dynamic behavior of those that do get access. The IC model, on the other hand, aims to regulate the dynamic behavior of interacting parties as well.

Structurally, IC differs from conventional AC, mainly in two ways: First, IC is inherently decentralized, although it lends itself, in practice, to a whole range of partially centralized implementations. Second, IC replaces the concept of AC *policy*—traditionally defined as an *access control matrix*—with the pair $\langle law, state \rangle$, where the fixed law holds sway over an entire community, while the state is a distributed collection of local states, each of which can change dynamically independently of the others, but subject to the global law. Among other things, this device unifies the concepts of DAC and MAC, conventionally viewed as a dichotomy under AC.

Among the resulting properties of IC, the following may be the most significant: (1) Scalability of the support provided for a wide range of constraints over interaction, including constraints that are sensitive to the history of interaction. (2) A significant realization of the age-old principle of *separation of policy from mechanism.* And (3) Flexible, composition-free, interoperability between agents operating under different laws.

The IC model is an abstraction of a mechanism called LGI. But it has been designed as a reference model that can be reified into a whole family of potential *IC mechanisms* that may support different types of communication, different performance requirements, and different application domains. (Indeed, an IC mechanism for ad hoc wireless communication has been recently defined and implemented, although not yet published.)

Also, the IC model admits a wide range of important extensions. The following are some of the extensions introduced to date into the LGI mechanism, and should be able to be introduced into other IC mechanism as well: (a) *proactive control*, provided via the concept of *enforced obligation* (first introduced in [21]), which ensures that a specified action would be carried out in the future, under specified circumstances; (b) the treatment of certain types of communication failures as regulated events, called *exceptions* [20], which can then be acted upon by the control mechanism, providing a significance fault-tolerance capability; (c) sensitivity to digital certificates held by the various agents, which provides this mechanisms an expressive power akin to that of *trust management* [10]; and (d) the organization of the laws of a given system into a *conformance hierarchy* [2], which facilitates the governance of very complex systems, like virtual organizations, grids, and dynamic coalitions.

Acknowledgment: I am indebted to Yaron Minsky for many rewarding discussions on this topic.

References

[1] X. Ao, N. Minsky, and V. Ungureanu. Formal treatment of certificate revocation under communal access control. In *Proc. of the 2001 IEEE Symposium on Security and Privacy, May 2001, Oakland California*, May 2001.

[2] X. Ao and N. H. Minsky. Flexible regulation of distributed coalitions. In *LNCS 2808: Proc. European Symp. on Research in Computer Security (ESORICS)*, Oct. 2003.

[3] X. Ao and Naftaly H. Minsky. On the role of roles: from role-based to role-sensitive access control. In *Proc. of the 9th ACM Symposium on Access Control Models and Technologies, Yorktown Hights, NY, USA*, June 2004.

[4] Xuhui Ao and Naftaly H. Minsky. Regulated delegation in distributed systems. In *Proc. of the IEEE 7th International Workshop on Policies for Distributed Systems and Networks, London, Ontario, Canada*, June 2006.

[5] W Arbaugh, D Farber, and J Smith. A secure and reliable bootstrap architecture. In *Proceedings of 1997 IEEE Symposium on Security and Privacy*, 1997.

[6] Tuomas Aura. Distributed access-rights management with delegation certificates. In J. Vitek and C. Jensen, editors, *Secure Internet Programming: Security Issues for Distributed and Mobile Objects*, volume 1603 of *LNCS*, pages 211–235. Springer, 1999.

[7] Olav Bandmann, Mads Dam, and Babak Sadighi Firozabadi. Constrained delegations. In *proceedings of 2002 IEEE Symposium on Security and Privacy*, May 2002.

[8] J. Barkley, K. Beznosov, and J. Uppal. Supporting relationships in access control using role based access control. In *Proceedings of the Fourth ACM Workshop on Role-Based Access Control*, pages 55–65, October 1999.

[9] E. Bertino, P.A. Bonatti, and E. Ferrari. Trbac: A temporal role-based access control model. *ACM Tran. on Information and System Security*, 4(3):191–233, 2001.

[10] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The role of trust management in distributed systems security. *Secure Internet Programming: Issues in Distributed and Mobile Object Systems*, 1603, 1999.

[11] M. Coetzee and J.H.P. Eloff. Virtual enterprise access control requirements. In *Proceedings of SAICSIT 2003 Conference*, pages 285–294. ACM, October 2003.

[12] L. Giuri and P. Iglio. Role templates for content-based access control. In *Proc. of the ACM Workshop on Role-Based Access Control (RBAC'97)*, pages 153–159, 1997.

[13] S. Godic and T. Moses. Oasis extensible access control. markup language (xacml), version 2. Technical report, Oasis, March 2005.

[14] Z. He, T. Phan, and T.D. Nguyen. Enforcing enterprise-wide policies over standard client-server interactions. In *Proc. of the Symp. on Reliable Distributed Systems (SRDS)*, 2005.

[15] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subramanian. Flexible support for multiple access control policies. *ACM Trans. on Database Systems*, 26(2), June 2001.

[16] James B. D. Joshi, Elisa Bertino, Basit Sahfiq, and Arif Ghafoor. Dependencies and separation of duty constraints in gtrbac. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT 2003)*, 2003.

[17] Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Transaction on Information and System Security (TISSEC)*, pages 128–171, Feb 2003.

[18] P. McDaniel and A. Prakash. Methods and limitations of security policy reconciliation. In *Proc. of the IEEE Symp on Security and Privacy*, pages 66–80, May 2002.

[19] Naftaly H. Minsky. The imposition of protocols over open distributed systems. *IEEE Transactions on Software Engineering*, February 1991.

[20] Naftaly H. Minsky. *Law Governed Interaction (LGI): A Distributed Coordination and Control Mechanism (An Introduction, and a Reference Manual)*, February 2006. (available at `http://www.moses.rutgers.edu/documentation/manual.pdf`).

[21] Naftaly H. Minsky and A. Lockman. Ensuring integrity by adding obligations to privileges. In *Proceedings of the 8th International Conference on Software Engineering*, pages 92–102, August 1985.

[22] Naftaly H. Minsky and V. Ungureanu. Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *TOSEM, ACM Transactions on Software Engineering and Methodology*, 9(3):273–305, July 2000.

[23] M.J. Moyer and M. Abamad. Generalized role-based access control. In *Proc. of the 21st Intern. Conf. on Distributed Computing Systems*, pages 391–398, 2001.

[24] S. Osborn, R. Sandhu, and Q. Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security*, 3(2):85–106, May 2000.

[25] Carlos Ribeiro and Paulo Ferreira. A policy-oriented language for expressing security specifications. *International Journal of Network Security*, 5(3):299–316, November 2007.

[26] R.S. Sandhu, D. Ferraiolo, and R. Kuhn. The nist model for role-based access control: Towards a unified standard. In *Proceedings of ACM Workshop on Role-Based Access Control*. ACM, July 2000.

[27] F.B. Schneider, editor. *Trust in Cyberspace*. National Academy Press, 1999.

[28] V. Ungureanu and Naftaly H. Minsky. Establishing business rules for inter-enterprise electronic commerce. In *Proc. of the 14th International Symposium on DIStributed Computing (DISC 2000); Toledo, Spain; LNCS 1914*, pages 179–193, October 2000.

[29] D. Weyns, A. Omicini, and J. Odell. Environment as a first class abstraction in multiagent systems. *Jou. on Autonomous Agents and Multiagent Systems*, 14(1), January 2007.

[30] W. Wulf, E. Cohen, W. Corwin, A. Jones, C. Levin, C. Pierson, and F. Pollack. Hydra: The kernel of a multiprocessor operating system. *CACM*, 17:337–345, 1974.