STATUS DIGEST # 2: COOPERATIVE BEHAVIOR

BlockScience

September 28, 2020

OVERVIEW

This digest builds upon the foundational **STATUS Digest #1: Architecture** to describe the incentive mechanism underlying the p2p network. Applying tools from Computational Social Science, a brief overview of incentive theory is followed by a description of two incentive mechanisms: a local mechanism drawing upon reciprocal cooperation according to social norms, and a global mechanism leveraging the efficient usage of a DLT-based smart contract to conduct random audits of network behavior.

Contents

1 A Brief Tour of Incentive Theory

1.1 Historical Scope

Incentivization is the process by which a designer creates circumstances under which:

- **decision-making agents** take
- **actions**, whose
- **outcomes** reinforce
- **goals** the designer wishes to support.

The theory of incentives and incentive structures has a rich history in the social sciences, particularly in Economics. Following initial work connecting decision theory to operations research by Nobel Laureate Kenneth J. Arrow (Arrow 1957) and the theory of the firm as an implementation mechanism by (among others) Laureates Bengt Holmström, Oliver Hart, Jean Tirole, and Oliver Williamson (see e.g. Holmström and Tirole 1989, for a comprehensive survey), the foundation for the theory of incentives within Economics was largely laid within the field of Contract Theory, using the applied mathematical tools of Game Theory and Mechanism Design (see e.g. Bolton and Dewatripont 2004, for an accessible textbook approach).

Incentive theory was first articulated under the assumption of *agency*, where a decision-making individual takes an action that determines (partly or fully) an outcome affecting both themselves and another individual, often assumed to be the designer of the decision-maker's circumstances or environment. A prototypical example of this formulation is **principal-agent** theory, in which a *principal* designs the circumstances, such as a remuneration scheme, market mechanism, or other "rules-carrying" ecosystem, and an *agent* responds to the circumstances with actions the principal cannot (or cannot afford to) take. Based upon the combination of the circumstances and the actions, both the principal and the agent earn a *payoff* based upon the outcome(s) that obtain. Modeling interaction between a principal and an agent has been a mainstay of corporate finance (e.g. Tirole 2006), imperfect competition (e.g. Maskin and Riley 1984) and incentive theory (e.g. Laffont and Martimort 2002), while providing the canonical paradigm for exploring the two main problems of incentive theory, **moral hazard** and **adverse selection**.

1.2 Moral Hazard and Adverse Selection

In our daily experience we recognize that not every objective desired by one individual will be desired by another. Within the context of centralized institutions (such as a court of law), contractually agreed-upon decisions between one or more parties may be enforceable, even if in the absence of such institutions one party may have found it preferable to *renege* on their agreed-upon action and perform an action that benefited them to the detriment (or exclusion) of another. Such enforceability, however, relies upon *observability* of the agreed-upon action, which is not (in general) the case. More commonly, an action may be hidden from counterparties or from a centralized institution.

It is this lack of observability that creates two standard incentive problems: the first is that an agreed-upon level of effort may not be realized, as a party hides their actual level of effort behind the 'noise' associated with the day-to-day randomness of outcomes that depend upon their action. Much of the original work on incentive structures within a firm revolved around this hidden-action problem, known as **moral hazard**. As a simple example, consider the office employee who "kills time" by playing an online browser game when alone, but switches to a work-related browser tab when a fellow employee (or supervisor) is nearby. The employee believes that the 'signal' of their office productivity in work-related activities is difficult to disentangle from the output of the firm, and hence deviations from agreed-upon activities, such as online gaming, can be performed without an enforceable repercussion (such as a warning or, in the worst case, the loss of a job).

The second incentive problem concerns the observability of one or more *attributes* that a party wishes a counterparty to have, or that are depended upon in order to create an agreement—and, hence, agreed-upon actions—that achieve an objective. Such attributes, often summarized as an individual's *type*, may not be observable when an agreement is formed—thus, each party will wonder whether or not agreed-upon actions that are dependent upon type will actually be undertaken. The incentive problem of **adverse selection** occurs when there is a non-negligible chance that a party, wishing to form an agreement with a counterparty of one type, will instead find themselves in an agreement with another type that does not reinforce (or works actively against) the agreed-upon goals. Continuing the example of employment, an employer searching for an employee uses a résumé or *curriculum vitae* to screen job candidates, in order to ascertain their suitability for the employment position (i.e., the employee's type). The common problem of résumé "inflation", where skill sets are claimed that are not in possession of the job candidate, confronts the employer with an adverse selection problem because verifying such skills is too difficult (or prohibitively costly) during the hiring process. It is then possible that a candidate, hired for the inflated skill sets, will underperform in the position due to the type mismatch that occurs. This mismatch is (in the absence of a way to prove the résumé's correctness) generally only discovered over time, resulting in lost productivity.

Addressing moral hazard and adverse selection problems involves the principal designing the circumstances of an agent so that:

1. conditional upon the 'proper' agent actions, the outcomes support the designer's goals, while
2. conditional upon the circumstances, the agent finds it in their own best interest to select the 'proper' actions.

In this context, the 'proper' actions selected by the agent are *incentive compatible*. When compared to other feasible actions that produce outcomes with (possibly different) benefits, incentive compatible actions are those which are (at least weakly) *preferred* by the agent. The problem of designing the circumstances so that an agent adopts incentive compatible actions is a **mechanism design** problem (see e.g. Jackson 2001; Börgers 2015).

1.3 From Centralized to Decentralized Incentivization

Incentive problems have traditionally been studied within the context of a fully centralized enforcement system, in which one or more *institutions* exist to act as overseers and enforcers of contracted actions. Designing mechanisms to alleviate moral hazard and adverse selection have typically assumed centralized enforcement, while investigating e.g. different information structures (e.g. how much information about action or type is hidden), the timing of penalties, or the cost of enforcement. Models of fully *decentralized* enforcement were introduced largely in the work of Nobel Laureate Elinor Ostrom (Ostrom 1990; Ostrom 1991), whose seminal research on collective decision-making in the absence of centralized institutional enforcement ushered into Economics the formal study of norms, cultural understanding and enforcement mechanisms such as social exclusion/ostracization from a tribe, clan or village.[1] By carefully examining empirical evidence via case study, survey and other field work, Ostrom was able to conclude that mechanism design could be achieved at a societal level as a *collective goal* decision-making framework, by leveraging commonly-shared values instead of a centralized enforcement system.

Unfortunately, apart from simple toy models of behavior it is difficult to demonstrate analytically that one or more collective goals are incentivized through a commonly-shared value system (hereafter characterized as *norms*). This is largely due to the complexity of the interactions between individual decision-making entities, such that a decision made between two counterparties may have an effect on other parties far removed (contractually) from the original contract. As a result, it has only been in recent decades that such demonstrations are possible, by utilizing the massive growth in computational power to conduct computer simulations. As a leading example, when individuals are modeled as participating in a network (such as the STATUS network), it is possible to construct computational models of decision-making and interaction using tools from Computational Social Science (CSS). CSS (see e.g. Alvarez 2016, for an introduction to its application areas) originated in the mid-to-late 1980s, as computer processing power expanded enough to accommodate populations of decision-making entities, generically referred to as *Agents*, in stylized, "sandbox" models of population dynamics. Predator-prey models, cellular automata, and models of evolutionary economics were cast as computer simulations, providing a laboratory in which different initial conditions and behavioral assumptions could be imposed, and simulation results compared with real-world quantitative or qualitative 'stylized facts' to assess their veracity. Over the past 20 years, the explosive growth in both in-memory storage and parallel processing has helped to create formal CSS-based simulation frameworks, helping to test ever-more-realistic models of collective decision-making and the incentive structures that support them. An example of such a simulation framework is **cadCAD**.

[1] Such work was already extant in Sociology and Cultural Anthropology, but often without the formal structure favored by post-war Economics.

2 Decentralized Cooperative Behavior in the P2P Network

Recall[2] that the peer-to-peer (p2p) STATUS network handles the computation and (most) communication between network **Entities**, providing information flow to and from **App** components using their **Client** representatives in the network (the computing **Nodes**). **Clients** communicate in the p2p network using a network *protocol*, which stipulates both how communication can be achieved and also how information can be transferred.

In the **cadCAD** representation of the STATUS network (discussed in more detail in the third digest), incentivization is a distillation of communication flows to and from the **Contract** component (or components), which acts as an 'arbiter' in the assessment of 1) whether **File** or other service requests have been fulfilled, and 2) whether actions taken by **Clients** support system goals, or are (as in the case of e.g. D/DoS or Sybil attacks) detrimental to them. This distillation has the appearance of a 'centralized' enforcement mechanism because the reward and penalty methods are implemented at the **Contract** level, implying that a smart contract call is executed for *each* settlement event.

While this approach has the advantage of simplicity for the **cadCAD** representation, it is not a necessary condition for the incentivisation of the STATUS network that the **Contract** component be used for settlement of every service request. Rather, the norm-based collective goal decision-making discussed in Section 1.3 can be used instead to provide a *local incentive mechanism* that supports a **cooperative** outcome, i.e. an outcome where **Clients** engage in behavior that supports the collective goal(s) of the network. Instead of the **Contract** component exercising enforcement for every settlement event, it does so only rarely, depending upon the number of settlement events but otherwise in an unpredictable fashion. This *random enforcement* approach is discussed in further detail in Section 3, and is used as the point of departure for the overview of the **cadCAD** implementation of the STATUS network provided in the third digest.

2.1 Contrasting Local vs. Global Information

The foundation upon which the local incentive mechanism can be built is the flow of information within the p2p network, at both the local and global levels. *Local information flows* are governed by the network protocol, indicating what can (and cannot) be considered as communication between **Clients** and between an **App** and a **Client**. By contrast, *global information flows* are governed by the ability of the **Contract** component to 'broadcast' information to network **Nodes**,[3] and by the transmission of information from particular **Nodes**, such as "prover" **Clients**, to the **Contract** component. Each of these flows is required to support the incentive structure that allows **Messages** to be routed properly, and together they induce **Nodes** to behave in such a way as to support the system goals (cf. Section 4.1, STATUS Digest 1: Architecture).

As an example of the contrast between local and global information, consider an event in which a prototypical **Client** *C* has a nefarious neighbor, **Client** *X*, who requests routing of a **Message** to a recipient **Client** *Y* (see Figure 1, p. 7 for an example in which a **File** is requested). **Client** *C*, not having 'Y' in their neighborhood, accepts the cost of broadcasting

[2] Cf. STATUS Digest #1: Architecture.
[3] Recall that both **Clients** and **Apps** are **Nodes** in the network.

the Message to its neighbors on behalf of 'X' with the expectation that it will be compensated for the broadcast cost when the Message arrives at its intended destination. Client *X*, however, has no intention of paying if the Message is delivered—and Client *C* has, in this example, no way of distinguishing between two possible adverse delivery outcomes that each result in non-payment to 'C':

1. the Message was delivered as intended, but Client *X* does not pay;
2. the Message was unable to find a route to its destination, in which case no payment from 'X' is required—such a situation would obtain if, say, the recipient Client *Y* is no longer a member of the network, or if a network disturbance created a communication error.

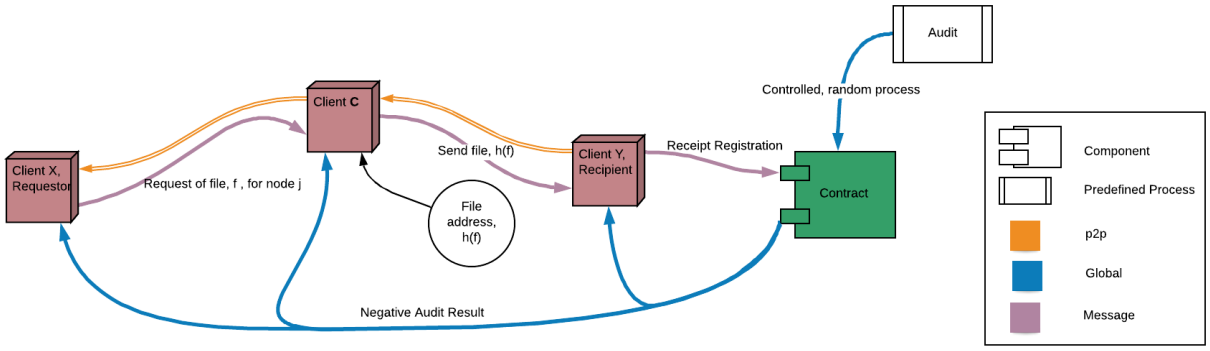

Figure 1: Request Routing Path and Information Flows

The network protocol, which responds to the *local* event information, does not encode Client *C*'s response to the event 'no compensation received'. The network protocol merely responds to local information as processed by each Client, and does not restrict what they can or cannot do with that information. Instead, the protocol allows Client *C* to decide whether to continue interacting with Client *X*, either by keeping 'X' in their neighborhood, or by dropping 'X' and ignoring further Messages from them.

Continuing the example, suppose next that the Message was in fact successfully delivered to its recipient. This information must be conveyed in a trustworthy fashion, so that all Clients participating in its delivery are notified (and thus expect compensation from the source Client *X*). This is the natural role that is played by a distributed ledger technology (DLT). Using cryptographic signatures to sign the Message as it propagates through the network, the recipient Client *Y* can register the receipt of the Message with the Contract component. This is publicly available, *global* information that can be checked in the event of a dispute, providing an auditable source of truth. Client *Y* can also relay its receipt registration to its immediate Client neighbor that passed on the Message, initiating an information flow from recipient to (ultimately) the source Client *X* via the intermediary Clients that handled the Message.

Thus, there are two information flows that proceed simultaneously: the registration of a completed delivery at the Contract (global) level, and the p2p (local) signaling to participants in the routing of the Message that delivery was performed. In the following section, an incentive mechanism at the local level will be introduced that supports the

decision space of a Client, i.e. that they can decide whether or not to interact with other Clients in their neighborhood. In Section 3, a supporting incentive mechanism at the global level will be introduced, that creates an additional global information flow when a randomly-audited route indicates bad behavior (such as non-payment) of one or more participating Clients.

2.2 A Local Cooperation Incentive Mechanism

In a decentralized, distributed network, parties have limited tools at their disposal to enforce agreed-upon behavior with counterparties. The simplest tool, perhaps, is the *exclusion* of a counterparty from interaction, following an observable signal that the counterparty has reneged upon promised behavior. This 'ostracization' or 'shunning' mechanism serves as one basis for collective action based upon norms, which Ostrom and others have argued are themselves the fundamental unit of accountability in societies where enforcement is local, i.e. between participants, rather than global, i.e. from an external (or centralized) institution. From a p2p networking perspective, the exclusion tool is attractive because it places little structure upon the network protocol, provided that the norm is shared by (or a requirement of) network participants. The burden rests upon the shared norm: each participant in the network must value interaction in the same, reciprocal way, although their individual assessments of counterparties will naturally be governed by their own experiences and not imposed by the norm or the structure of the network protocol.

One way to formalize the shared norm–exclusion mechanism is by utilizing a *trust score* approach.[4] First consider a network made up of N Nodes, indexed by $k = 1, \ldots, N$, with the kth node denoted by k or by n_k, as defined by context. Without loss of generality we shall assume in what follows that all Nodes are Clients, as they are the 'workhorse' computation and communication Nodes of the network and Apps are assumed only to communicate with those Clients that act as their 'gateway' to the network.

We know from the network's local structure (cf. Digest #1: Architecture, Section 3.4) that a Client k has a *neighborhood* η_k,[5] containing all Clients $i \neq k$ that Client k is aware of. Client k assigns every Client $n_i \in \eta_k$ a *trust score* $\tau_k(i)$, representing the **reputation** of Client i according to Client k:

Node Characteristic: Trust Score

The trust score is the probability that, upon receiving a message from Client i, Client k will interact with i. Thus, $\forall k, i,\ \tau_k(i) \in [0, 1]$.

Associated **cadCAD** variable: `control`

The trust score of a Client in Client k's neighborhood is a simple device that encodes the shared norm that every Client is assumed to hold:

[4] In what follows a mathematical formalization will be introduced, but examples in the same vein as Section 2.1 will be treated in Sections 2.2.1 and 2.2.3).

[5] Note that the notation presented here may differ from Digest #1 as comparisons between Clients will be made.

Node Characteristic: Shared Norm

The shared norm states that Clients keep their promises to other Clients.

If $\tau_k(i) = 0$ then Client k does not trust Client i enough to interact with—the shared norm has been seen (or inferred) to be violated by Client i, and we define this polar case to be the situation when Client k *does not trust* Client i. If, on the other hand, $\tau_k(i) = 1$ then Client k *fully trusts* that Client i adheres to the shared norm. Cases where $\tau_k(i)$ lies between 0 and 1 represent *degrees of trust* and specify Client k's uncertainty about whether or not Client i adheres to the shared norm.

This probabilistic interpretation of trust helps to automate a Client's decision problem. Suppose that a Message is received by Client k from Client i. Client k needs to know how to respond to the Message—in particular, it needs to **decide** whether or not to interact with Client i. To make this decision Client k appeals to the definition of the trust score as a probability of interaction: first, Client k draws a random realization $\hat{x}$ from a Uniform $[0, 1]$ distribution. Second, the random realization and the trust score are compared: if $\hat{x} < \tau_k(i)$, then Client i interacts with Client k's message. Otherwise, Client i is *excluded* from interaction, and the Message is discarded.

A trust score is updated according to an *external norm* (Crawford and Ostrom 1995), representing the coordinating mechanism that reinforces the (collective) system goals for all Nodes. This external norm, referred to by Crawford and Ostrom (1995) as a *delta parameter*, is denoted here by $\Delta_k(i)$ and represents Client k's **updating** of Client i's trust score:

1. When Client i is **first** discovered by Client k it is added to its neighborhood η_k with full trust, i.e. $\tau_k(i) = 1$ initially;
2. If Client i reneges on a promise to pay (denoted by setting a 'renege' parameter $r = 1$ in what follows) then $\tau_k(i)$ declines by an amount $\Delta_k(i)$;
3. If Client i upholds a promise to pay (denoted by setting $r = 0$) then $\tau_k(i)$ increases by an amount $\Delta_k(i)$.

The update value $\Delta_k(i)$ here represents a fraction of the overall trust that Client k attributes to its neighborhood η_k, so that it can reward a Client i in a manner that matches its overall trust (or, alternatively, penalize actions that go against the shared norm). More formally, it is a penalty or reward that is proportional to a function of the average trust score across η_k, where the constant of proportionality is the *distrust* that Client k attributes to a Client i.

Definition 2.1. A Client k updates the trust score of Client i (in response to either a broken or a kept promise of payment for delivery of a Message) using the update value $\Delta_k(i)$:

$$\Delta_k(i) := \mathbb{T}(r, \tau_k(i))(1 - \tau_k(i))f(\bar{\tau}(\eta_k)), \tag{2.1}$$

where

$$\bar{\tau}(\eta_k) := \frac{1}{N_k} \sum_{j=1}^{N_k} \tau_k(j) \tag{2.2}$$

is the mean trust score for neighborhood η_k, N_k is the size of node k's neighborhood η_k, and $\mathbb{T}(r, \tau_k(i))$ is the entry corresponding to the following *lookup table* for combinations of renege parameter r and trust score $\tau_k(i)$:

Table 1: **Delta Coefficient Lookup Table** $\mathbb{T}$

	$\tau_k(i) = 0$	$0 < \tau_k(i) < 1$	$\tau_k(i) = 1$
r = 0	1	1	0
r = 1	0	1	1

Finally, the function $f(x)$ has the following properties over $x \in \mathbb{R}$:

1. $f(0) \geq 0$, $\lim_{x \to -\infty} f(x) = 1$, $\lim_{x \to -\infty} f'(x) = 0$;
2. $f(1) \leq 1, \lim_{x \to +\infty} f(x) = 0$, $\lim_{x \to +\infty} f'(x) = 0$;
3. $f(\cdot)$ is at least C^1, i.e. is continuously differentiable with continuous derivatives.

2.2.1 Interpreting Definition 2.1

Before delving into the specifics of the functional form of f it is worth understanding this definition by examining its polar cases, and confirming that they agree with an intuitive interpretation of the shared norm–exclusion mechanism. First, suppose that **Client** k's trust score for **Client** i is exactly zero, i.e. $\tau_k(i) = 0$. If **Client** i then reneges on its promise, i.e. $r = 1$, Table 1 indicates that $\mathbb{T}(1, 0) = 0$. From equation (2.1) this means that $\Delta_k(i) = 0$, and so no update occurs—a **Client** that is already untrusted cannot "fall further" by continuing to break promises. Conversely, if **Client** i is fully trusted so that $\tau_k(i) = 1$, then keeping a promise, $r = 0$, does not make them "even more trusted": $\mathbb{T}(0, 1) = 0$ once again, and so too is $\Delta_k(i) = 0$.

Outside of these extremes there is room for improvement or for deterioration, when

1. $r = \tau_k(i) = 1$,
2. $r = \tau_k(i) = 0$, or
3. $0 < \tau_k(i) < 1$.

The first two cases cover when **Client** i "goes against type": if $r = \tau_k(i) = 1$, for example, then a trusted neighbor has broken its promise, while conversely if $r = \tau_k(i) = 0$ then an untrusted **Client** has fulfilled its obligation. The last case covers the situation where the trust score of **Client** i lies somewhere between zero and one, and thus has "room to maneuver" in response to actions (i.e. in response to a decision by **Client** i to renege or to keep its promise). In each of these cases, $\mathbb{T}(r, \tau_k(i)) = 1$ and so from (2.1) the trust score of **Client** i is updated by the value $(1 - \tau_k(i)) f(\bar{\tau}(\eta_k))$.

Continuing this interpretation further, we see that the first part of this term, $(1 - \tau_k(i))$, normalizes the update so that the trust score stays between zero and one (as it must for its interpretation as a probability of interaction). The second term, $f(\bar{\tau}(\eta_k))$, is a function of the average or *mean value* of trust scores $\bar{\tau}(\eta_k)$, across all **Nodes** in the neighborhood η_k of **Client** k. This value represents the environment's representation of the shared norm: if the mean neighborhood trust score is close to one, then **Client** k's experience[6] reflects the shared norm. If, by contrast, the mean neighborhood trust score is close to zero, then **Client** k's experience reflects a deviation from the shared norm.

Updating **Client** i's trust score must take this experience into consideration, generating an asymmetric response that both reinforces the shared norm and 'responds in kind' to observed deviations. For example, if **Client** i reneges on its promise (but is still at least partially trusted) then the 'penalty' to its trust score (its delta parameter update) should be larger, the more **Client** k's experience reflects similar untrustworthy behavior. In other words, **Client** i is penalized more harshly when **Client** k has faced (on average) behavior that does not support the shared norm—there is **reciprocity** in **Client** k's response. If, on the other hand, **Client** i keeps its promise (and is still at least partially untrusted) then the 'reward' to its trust score should also be larger when **Client** k has experienced largely *un*trusted behavior: **Client** k then 'does its part' to **proactively** reward the demonstrated adherence to the shared norm more strongly, the more strongly it has experienced deviations from the shared norm by members of its neighborhood.

In this way the function f, applied to the mean trust score $\bar{\tau}(\eta_k)$ of **Client** k's neighborhood, should *amplify* rewards and penalties when $\bar{\tau}(\eta_k)$ is low, and should *attenuate* rewards and penalties when $\bar{\tau}(\eta_k)$ is high. The properties of f as given in Definition 2.1 provide this general structure, while also ensuring (once again) that trust scores remain as probabilities (i.e. between zero and one).

2.2.2 Selecting the functional form of f

There are many functional forms for f that fit the properties described in Definition 2.1—for example, the statistical tail distribution (sometimes referred to as the *survival function*) that is at least C^1 is a candidate. The decreasing **logistic** function (see Appendix A) is advantageous because it is easy to implement on compute nodes that may have limited computational resources. In either case, the selected function contains a useful *threshold* property, namely $\exists M > 0,\ \underline{x}, \overline{x} \in X,\ \overline{x} \geq \underline{x}\ \ni$

- $|f'(x)| \geq M\ \forall x \in [\underline{x}, \overline{x}] \subset [0, 1]$,
- $|f'(x)| < M\ \forall x \in [0, \underline{x}) \cup (\overline{x}, 1]$.

Notice that the properties of f (above) ensure that $f'(x) << M$ as x approaches zero or one, and that the change from a large value of the slope to a smaller value (and eventually zero at the endpoints) is continuous.

[6] Note that local experience may also be interpreted as a local *estimate* of the network's global adherence to (or deviation from) the shared norm.

The threshold property is important because by 'tuning' M it is possible to generate more graduated or (if desired) more severe responses to reneging on a promise to pay, which allows the reinforcement mechanism to be configured to the network topology that is expected (and implemented in the **cadCAD** representation).

2.2.3 Supporting Local Cooperation at the Global Level

Putting the trust score and delta framework together yields the *law of motion* for trust scores in a neighborhood, and this law of motion is considered by a **Client** when deciding whether or not it is worth reneging upon a promise to pay. Notice that this is *not* a restriction on the network protocol, nor is it an *a priori* assumption regarding the preferences or decisions of such **Clients**. Rather, the framework "sets the stage" for rewards and penalties that local **Clients** can apply to one another, and this framework is conditioned upon by **Clients** when they make their decisions.

More specifically, a **Client** will decide whether or not to renege on a promise to pay for the fulfillment of a service request by comparing the relative values of the following two benefits:

1. the net benefit of **reneging** today and lowering their trust score, i.e. lowering (or maintaining, if they are untrusted) the probability of interaction tomorrow; and
2. the net benefit of **paying** today and improving their trust score, i.e. increasing (or maintaining, if they are trusted) the probability of interaction tomorrow.

These net benefits are most readily compared using the value function approach from dynamic programming[7]. It can be shown that incentive compatible behavior is reinforced, i.e. behavior is supported that reinforces the shared norm, at a frequency which is determined by how often a **Client** that reneges on a payment is "shunned" *at the global level*. That is, this framework works in conjunction with a *random enforcement mechanism* from the **Contract** component, in which routes are occasionally audited and checked to see if a **Client** has kept or reneged on its promise. The degree to which incentive compatibility is enforced, i.e. how often one might expect promises to be kept, depends upon how often the **Contract** component elects to audit **Message** routes. Finding the proper audit frequency (conducted analytically and in conjunction with **cadCAD**'s network implementation) allows the STATUS network to decide upon its desired trade-off between incentive compatible behavior on the one hand, and the involvement (and concomitant complexity, speed delay etc.) of the **Contract** component on the other. An overview of the **Contract** component and its role in the incentive structure is the focus of the following section.

3 Contract Random Auditing and Enforcement

The decentralized mechanism for incentivizing local behavior between **Clients** in the p2p network does not use direct value transfers. Instead, it uses local ostracization of a misbehaving **Client** as its method of enforcement. The result is

[7] See e.g. Bakos and Dellarocas (2011) for an example of the value function approach using a different 'trust score' updating mechanism.

that a **Client** who reneges on its promise to pay for a fulfilled service (e.g. **Message** or **File** transfer) loses some portion of its connectivity to the rest of the network.

Because this type of exclusion occurs only at the local, p2p level, it is conceivable that a **Client** could strategically decide to 'game the system' by keeping some promises and breaking others. That is, one possible outcome is that the resulting network is *sparsely connected* by a few **Clients** that:

1. exercise a form of 'market power' by acting as bottlenecks within the sparse network, while
2. breaking connections within the network by reneging on just those promises that are required to keep the network sparse, ensuring their market power.

As a stylized example, suppose that a particular part of the overall p2p **Client** network, called here the "subnetwork", is initially fully connected between its members: every **Client** is connected to every other **Client** in the subnetwork. Suppose further that a **File** request is made from a **Client** *X* outside of the subnetwork, and that only one **Client** in the subnetwork, **Client** *Y*, has a connection to that **Client**.[8] **Client** *Y* would like to ensure that as few **Clients** within the subnetwork are aware of **Client** *X*'s request as possble, so that **Client** *Y* maximizes the chance of its participation in any fulfilled route, receiving its participation reward.

To accomplish this **Client** *Y* embarks on a 'bridge burning' exercise—it initiates requests with **Clients** within the subnetwork, and reneges on promises until the trust score of all but one such **Client**, say **Client** *Z*, is dropped to (or sufficiently near) zero. This ensures that requests coming from **Client** *X*, outside of the network, are routed only via **Client** *Y* through **Client** *Z*. The **Clients** in the subnetwork (apart from **Client** *Z*, who may retain information regarding the initiating **Client** *X* from the receipt of request fulfillment it receives, cf. Section 2.1, p. 6 and Figure 1, p. 7) remain ignorant of **Client** *X*.

The key to this manipulation is that **Client** *X* does not know (or care about) the poor trust between **Client** *Y* and nearly everyone in the subnetwork—as far as it is concerned, **Client** *Y* "does right" by **Client** *X*, and there is no "stigma" associated with interacting with **Client** *X*. If such a "stigma" were to exist, then **Client** *X* would lower its trust score of **Client** *Y*, participating in the support of the external norm not because it had first-hand knowledge of **Client** *X*'s bad behavior, but because stigma is (by definition) a globally-observed property that **Client** *X* can condition upon. And conditional upon this behavior, **Client** *X* may (depending upon such factors as the probability of being stigmatized, see below) find it optimal to keep its promises to the subnetwork and relinquish its market power on routes from **Client** *X* to the rest of the subnetwork.

3.1 Costly state verification and auditing

Implementing a form of 'stigma' in response to bad behavior on the part of a **Client**, such that this stigma is 1) a trusted signal of actual behavior (rather than a strategic tool used by one **Client** against another **Client**) and 2) *globally* known

[8] This simple example represents the more general case where a **Client** may have high *betweenness centrality* between that part of the overall network external to the subnetwork, and the subnetwork itself. See e.g. Jackson (2008) for further information on Social Network Analysis statistics in this context.

by all members of the network, can be performed by the **Contract** component. This component is *a priori* trusted by all members of the network as it is part of the DLT implementation, and has as part of its action set the capability to broadcast to all members of the network **Messages** that may include whether or not particular **Clients** have reneged on their promised behavior. If such *monitoring* were both:

- perfect, in the sense that every instance of bad behavior in the network was caught by the **Contract** component, and
- costless, in the sense that the **Contract** component could perform smart contract calls instantaneously and with zero processing fees, charges, etc.,

then full enforcement of good behavior would be possible. A **Client** reneging on its promises would be instantly stigmatized, the local cooperation mechanism would update its trust score accordingly, and the resulting outcome would be worse for the **Client** than if the **Client** had kept its promise and fulfilled its obligations by supporting the shared norm.

Unfortunately, the information and implementation requirements needed to create a perfect and costless monitoring mechanism are usually too severe: the number of **Clients** in the network is expected to far exceed the capability of the **Contract** component to monitor every route for bad behavior, while the number of smart contract calls to broadcast bad behavior in every case found may suffer from bottlenecks (and hence increased costs) if, statistically, there are periods where bad behavior 'clusters' and is over-represented in a batch of routes undergoing monitoring. Such an environment has been extensively studied in the Contract Theory and Accounting literatures, where an unknown state cannot be monitored without cost. These *costly state verification* (CSV) models (see e.g. Krasa and Villamil 2000; Bakos and Dellarocas 2011; Ben-Porath, Dekel, and Lipman 2019; Erlanson and Kleiner 2020) carefully examine the foundations of the underlying game-theoretic/mechanism design problem, and determine (if possible) the correct set of "rules of the game" that supports good behavior without infeasible, "Big Brother" monitoring and oversight.

One such ruleset, or mechanism, is for the **Contract** component to relinquish complete oversight of the network in favor of "occasional" oversight, in the form of **auditing**. An audit acts probabilistically on routes that are submitted, or registered, with the **Contract** component as having been fulfilled. For example, if a proving **Client** has submitted a batch of routes that have had their **Message** or **File** requests successfully fulfilled, then the **Contract** component may randomly select one such route and conduct an audit of its properties. Such an audit would address:

1. non-payment, i.e. reneging on a promise to pay following request fulfillment;
2. self-routing, i.e. creating vacuous self-loops in the route to garner a larger share of the promised payment;
3. inefficiency, i.e. creating longer-than-required routes, perhaps as part of a sybil attack to garner a larger share of the promised payment for an underlying (controlling) **Entity**.

If the outcome of the audit were positive, i.e. none of the aforementioned issues were detected in the selected route, then the **Contract** component would do nothing further. If, however, one or more of the audit outcomes were negative,

then the **Contract** component would broadcast the identity of the **Client** (or **Clients**) participating in bad behavior to the network.

3.2 Parameterization and cadCAD

Conditional upon the frequency of auditing (which is up to the **Contract** component designer and is well-suited to **cadCAD** simulation and analysis), a potentially bad-behaving **Client** will reduce the frequency of such behavior to a value that (from a dynamic programming assessment) just balances the behavior's marginal payoff with its marginal cost, again in the sense of opportunities foregone from a reduction of trust that occurs due to stigmatization at the *global* level. Faced with the possibility that an audit will uncover its bad behavior and update *every* **Client**'s neighborhood with the stigma of its action (i.e. its trust score reduction), a **Client** will find it best to match up its own desire for cooperative reciprocity with other **Clients**, supporting the shared norm, with its degree of conformity *to* the shared norm that the audit incentivizes.

As a result, there will be *mapping* between the audit frequency and the (relative) frequency of bad behavior. Depending upon how the incentive mechanism is parameterized, this mapping can then be computationally tuned to meet a particular threshold for "acceptable" bad behavior in the network (since it is infeasible, as discussed above, to create a perfect, costless mechanism that fully prevents such behavior). The parametrization and specification of the network is discussed in more detail in the third digest, which covers the **cadCAD** implementation of the STATUS network.

Appendices

A The Logistic Function

The general form of the decreasing logistic function $f : \mathbb{R} \to [0, a]$ for a constant $a > 0$ is:

$$f(x) := \frac{a}{1 + bc^{-x}},$$

where $b \neq 0$ and $0 < c < 1$ are constants.

With this general form, the absolute value of the slope $|f'|$ of the logistic function achieves its maximum at

$$x^\star = \frac{\ln b}{\ln c},$$

with a value of $f(x^\star) = a/2$.

The logistic function can be used to compute the delta parameter (cf. Section 2) with the following restrictions on its free parameters: $a = 1$, $b = e^{-k}$, $c = e^{-2k}$, with k selected so that

$$0 < f(1) = \frac{1}{1 + e^k} < e^{-k},$$

$$0 < 1 - f(0) = 1 - \frac{1}{1 + e^{-k}} = \frac{e^{-k}}{1 + e^{-k}} < e^{-k}.$$

For example, selecting a value $k = 20$ gives $0 < f(1) < 2.06 \times 10^{-9}$ and $0 < 1 - f(0) < 2.06 \times 10^{-9}$, respectively.

References

[1] R. Michael Alvarez, ed. Computational Social Science: Discovery and Prediction. Analytical Methods for Social Research. Cambridge University Press, 2016. DOI: 10.1017/CBO9781316257340.

[2] Kenneth J. Arrow. "Decision Theory and Operations Research". In: Operations Research 5.6 (1957), pp. 765–774. URL: http://www.jstor.org/stable/166866.

[3] Yannis Bakos and Chrysanthos Dellarocas. "Cooperation Without Enforcement? A Comparative Analysis of Litigation and Online Reputation as Quality Assurance Mechanisms". In: Management Science 57.11 (2011), pp. 1944–1962. DOI: 10.1287/mnsc.1110.1390.

[4] Elchanan Ben-Porath, Eddie Dekel, and Barton L. Lipman. "Mechanisms With Evidence: Commitment and Robustness". In: Econometrica 87.2 (2019), pp. 529–566. DOI: 10.3982/ECTA14991.

[5] Patrick Bolton and Mathias Dewatripont. Contract Theory. MIT Press, 2004.

[6] Tilman Börgers. An Introduction to the Theory of Mechanism Design. Oxford University Press, 2015.

[7] Sue E. S. Crawford and Elinor Ostrom. "A Grammar of Institutions". In: The American Political Science Review 89.3 (1995), pp. 582–600. URL: http://www.jstor.org/stable/2082975.

[8] Albin Erlanson and Andreas Kleiner. "Costly verification in collective decisions". In: Theoretical Economics 15.3 (2020), pp. 923–954. DOI: 10.3982/TE3101.

[9] Bengt R. Holmström and Jean Tirole. "Chapter 2 The theory of the firm". In: vol. 1. Handbook of Industrial Organization. Elsevier, 1989, pp. 61–133. DOI: 10.1016/S1573-448X(89)01005-8.

[10] Matthew O. Jackson. "A crash course in implementation theory". In: Social Choice and Welfare 18 (2001), pp. 655–708. URL: https://doi.org/10.1007/s003550100152.

[11] Matthew O. Jackson. Social and Economic Networks. Princeton University Press, 2008.

[12] Stefan Krasa and Anne P. Villamil. "Optimal Contracts When Enforcement is a Decision Variable". In: Econometrica 68.1 (2000), pp. 119–134. URL: http://www.jstor.org/stable/2999477.

[13] Jean-Jacques Laffont and David Martimort. The Theory of Incentives: The Principal-Agent Model. Princeton University Press, 2002. ISBN: 9780691091846. URL: http://www.jstor.org/stable/j.ctv7h0rwr.

[14] Eric Maskin and John Riley. "Monopoly with Incomplete Information". In: The RAND Journal of Economics 15.2 (1984), pp. 171–196. URL: http://www.jstor.org/stable/2555674.

[15] Elinor Ostrom. Governing the Commons: The Evolution of Institutions for Collective Action. Cambridge University Press, 1990.

[16] Elinor Ostrom. "Rational Choice Theory and Institutional Analysis: Toward Complementarity". In: The American Political Science Review 85.1 (1991), pp. 237–243. URL: http://www.jstor.org/stable/1962889.

[17] Jean Tirole. The Theory of Corporate Finance. Princeton University Press, 2006.