Repnet: An Alternative to Currency

Part 1: Motivation

1. Peer-to-Peer Systems Will Not Thrive on Generosity Alone

Peer-to-peer systems present empowering and resilient alternatives to centralized corporate systems across many domains, and have much room for further innovation. Hosting files, social media, communication, etc. However, in most existing p2p systems, things only get done as essentially acts of charity. Admirable as individual actions, but systemically untenable as a strategy to eat the world.

First, the market cap of this strategy is low. The ideal of liberatory p2p systems may gain traction with a subculture of hacker nerds, but we will only ever be a fraction of the population. Furthermore, the vast majority of peoples' willingness to host p2p systems will sharply drop when that entails noticeable personal sacrifice. Seeding your torrents or spinning up an IPFS node in the background is a "might as well" type situation, but investing even a moderate amount of money in infrastructure is much less likely. Some might do so anyways to prove me wrong, prove that people are good. Admirable, but systemically untenable as a strategy to eat the world.

Second, dead altruists can't help anybody. Not literally dead in this case, but out of money. Even if we're in a world where many people are willing to sacrifice for liberatory techno-infrastructure, they still need to pay the rent, buy food, and buy the actual computers. If liberatory techno-infrastructure relies on being funded as peoples' side projects, as the fruits of abundance, it will be the first to go in times of scarcity when these people are more focused on surviving.

Third, charity is notoriously ineffective at making intelligent trade-offs, especially when venturing beyond the domain of basic survival needs. People are often willing to seed torrents that they themselves are interested in, but less so to arbitrarily host data that any person or script anywhere posts. This is as rational as individuals currently can be. If you consciously chose to download some specific movie, that itself is evidence that making that movie more available embetters the world. However, if you made your computer a public first-come-first-serve file host, it would quickly get filled up by people saying "oh free storage, I'll fill it up with random huge stuff there's a tiny chance anyone will ever want because it's free (to me) so might as well." This is not to mention stochastic vandal scripts posting random bytes as fast as their internet connection allows. So people put their computers to work on tasks they are personally intimately familiar with, and this produces more value than near-zero, but it's still far from collectively rational because cross-individual tradeoffs aren't being made.

Many may value a service that only others can provide, such as a file they have not yet downloaded, and value it such that they would be willing to sacrifice for it to be provided more than it is. But with no trustworthy mechanism to carry this signal the system will not adjust. As such, these systems rely on abundance to insulate themselves from their own inefficiency, able to liberate "might as well" type

tasks in “might as well” type contexts, but unable to metastasize beyond these domains, or to press against the edge of possibility of what can be provided, or to survive truly hard times, or even just to efficiently enrich the world with what abundance they grow in.

I believe in generosity and I believe that there are truly altruistic people. However, to a large degree, incentive pressures produce aggregate trends in individual behavior, and there tends to be misalignment therein between the selfish and the altruistic thing to do. I believe one of the most effective and efficient ways us altruists can act is by leveraging asymmetric strategies to nudge the selfish and the altruistic closer into alignment. That is the meta-strategy guiding this project.

2. Blockchains Aren’t the Solution; They Are Epistemically Centralized

The trending approach to p2p reciprocity mechanisms is blockchain-based filecoin systems. The central theme is that there is some cryptocurrency, a filecoin, with which the requester of a service pays the provider of that service. The implicit idea is that owning filecoin serves as proof that one has provided an amount of value to others, and thus may cash it in in exchange for a proportional amount of value in return.

My critique of blockchains in general is that they are what I call *epistemically centralized*. Blockchains are a technology which allows one to create centralized information processors with decentralized management. They create a tension sculpture of incentives which forces total consensus regarding system state. I believe that it’s fundamentally possible for people to cooperate on tasks to the degree that they agree on the premises underlying those tasks. If we agree on everything in the world except the rightful owner and use of some tavern, we can work together on most everything in the world except most tasks directly pertaining to that tavern. The nature of blockchains, however, is that one must act as if 100% of the premises encoded in the chain state are true, or they’re left with 0% ability to use the system and cooperate with people through it.

When one is appending transactions to a blockchain, they cannot say “I accept almost all of this, but I disagree with this one transaction because that money was actually stolen.” If they attempted to add their transaction to a version of the chain with that previous transaction excluded, no matter how they did it, they would fall behind in terms of chain length and thus not get their transaction picked up by anyone else, by design. Thus, when someone disagrees with some transaction in the chain, some underlying premise locked into the consensus state, the vast majority of the time they’re forced to simply take the hit and capitulate—so much cryptocurrency lost, stolen, hacked, manipulated, etc. I’d avoid betting my future on that, to say the least!

Every once in a while the energy levels of disagreement hit a point where a substantive chunk of people *would* rather 0% agree with and use the system than 100% agree with and use it, in which case a hard fork happens. In 2016 Ethereum launched a huge smart contract thing, which had a security vulnerability, because *of course* it did, and was hacked for about $50,000,000, because *of course* it was, and most of the Ethereum community simply packed their bags and moved to a hard fork. And this happens all the time. Because *of course* blockchains can’t turn *cooperation itself* a from fuzzy

defeasible open-ended thing into something where everything is objective and unambiguous and every statement uttered must be something one can commit to absolutely until the end of time. It just rounds the fuzzyness down to zero or up to one and externalizes the difference as dissonance and pain.

3. Existing Currencies Are Fake Objectivity

Implicit in that analysis is the idea that local patches of currency-mediated cooperation correspond to local patches of underlying factual premises the involved parties agree on. If a set of people agree on who among them is the rightful owner of certain things, they can negotiate trades of these things among themselves to determine what they consider to be just and positive-sum inter-individual tradeoffs regarding their use.

However, merely trading objects which have already been made is too limiting. People want to be able to trade plans for what to produce in the future. If the baker and the candlestick maker can both make a thousand dollars of product over the next week, and would both like to do so then swap, they should be able to. But if they can only trade with what they already own, and they each only have $100 worth of spare stuff, they have no way to make that arrangement.

Currency solves this by acting as tokens of ownership over near-future production. Owning a dollar means getting to choose the nature of one dollar worth of near-future work. In a highly idealized version of currency, the baker and the candlestick maker would both find themselves with $1000, representing that they will soon produce $1000 worth of value in the economy and are naturally entitled to ownership of it. They could then both order each others' product with that money.

This makes sense. Yet, there is a problem. To mediate an entire economy with one universally fungible currency as such is to take an intractably rich interconnected tapestry of defeasible tacit predictions and commitments and flatten it into a field of artificial objectivity and illusory clarity.

By whose word do we know that the baker and the candlestick maker can make $1000 of product in the next week? God's? Do we assemble everyone everywhere into a daily dissertation committee to audit the business promises of everyone everywhere? They may have convinced each other, but sometimes people are tricked. They may believe it themselves, but sometimes people are foolish. They may have a good argument, but so may the other 7 billion people in the economy. One can't hear them all out.

Speaking of the economy, what is "the" economy? We have global trade, but not with total symmetry of friction. A dollar represents value that can be given to others—but to whom? If someone in Antarctica during winter bakes a loaf of bread that will have long gone bad by the next time any travel to or from the rest of the world is possible, that's value which can be given to some people, but not to any arbitrary person. To a lesser degree there is nonuniform shipping costs everywhere. Some IKEA purchase may be worth the effort it imposes on others if it can be picked up from the same city, but not so if it has to be shipped.

There also may exist transactional friction in the form of social sanction. Someone may be perfectly willing to sell a computer to a bakery, willing to sell it for even less to a charity organization, and not at all willing to sell it to Lockheed Martin.

Speaking of value, valuable to whom? There are no “value” particles, “value” is a verb, to say a thing “is valuable” means that people value it, and value is subjective in that different people may value something more or less than others. Objective value exists only as statistical aggregates of individuals valuing something. By whose word do we know that many people would value something a certain amount? God’s? Do we scan everyone’s brain into a matrix that runs various permutations of simulations to see what they’d sacrifice for it?

4. This Is a Problem With Both Crypto And Dollars

The government’s answer to these questions is pretty unsurprising: just have technocrats figure out what’s true, and then it will be as such. Specifically, banks. Banks don’t just store dollars, they hold the privilege of bringing them into existence in the first place. For normal people, to spend some number of dollars, one first has to own that number of dollars. This is not true of banks. Banks are given special permission by the government to spend a certain amount of dollars without actually having them in the first place. They use this power to give loans to people who have convinced them they can use that money to produce economic value, and then they demand that money back in the future, with interest, under threat of stealing their house. This is how all dollars are created.

As everyone is well aware, this system basically works fine. Banks never ruin peoples’ lives in embarrassing ways and there’s nothing to be concerned about regarding a physically enforced state-cartelized monopoly on truth regarding who can do valuable stuff, which demands rent for doing so.

Cryptocurrencies make an admirable attempt to liberate currency from centralized hierarchical command structures. However, they’re hilarious in a different way: currency entering the system has zero causal link with the existence of near-future productive capacity at all. In cryptocurrencies, currency may be created by mining, or through some other mechanism such as proof of stake or of service, or not at all (see Nano coin). There is generally not a causal flow from evidence of near-future productive capacity to the minting of coins to represent it. “Proof of service” systems may be an exception, but only in very limited domains, and still subject to all aforementioned issues of subjectivity. There may technically be a *correlation* between circulating currency and productive capacity, because circulating currency generally increases with time and the economy tends to grow over time. But this is like saying the water level in a pot catching a leaking pipe is a signal of how frustrated people are in some meeting because the pot fills up over time and people get more frustrated the longer they’re in a meeting.

Part 2: Repnet

5. Repnet

I present a peer-to-peer system, repnet, which serves as an alternative to currency. I’m taking a gradualist approach to trying to bootstrap it into reality. To start, we try to integrate repnet into other peer-to-peer systems, such as file hosting, as a reciprocity mechanism. In the long run, we can keep our eyes open for paths to people starting to use it as an alternative to dollars for IRL purchases in one context or another.

Repnet starts with a microphysics of cooperation at the miminum meaningful scale: cooperation between two people. Repnet then facilitates the bottom-up emergence of cooperation from each link in a social graph to cooperation at the scale of the entire graph, without ever flattening out the truly decentralized, rhizomatic, subjective reality of the situation.

6. Rep & Groups of 2 People

“Rep.” Short for reputation, but it’s really an IOU. If Alice “gives” Bob X amount of rep, she’s promising to Bob that he can redeem that rep for a proportional amount of valuable stuff from her in the future. If Bob trusts Alice to honor this promise, and Bob has some sense of the conversion rate from redeeming Alice’s rep to getting stuff he values, Bob may be willing to sell stuff to Alice in exchange for a certain amount of Alice’s rep.

There is no broad system of blockchain-like incentive traps to enforce Alice’s commitment. There is only Bob’s trust in Alice and Alice’s desire to give Bob reason to keep trusting her. If Alice refused to give Bob valuable stuff in exchange for Bob redeeming her rep, Bob would simply stop treating Alice’s rep as valuable and stop giving her stuff or otherwise making sacrifices in exchange for it[1]. As such, all rep really is is promising to someone that you’ll pay them back for something, in a way that’s quantified and accounted for.

Because of this, a blockchain is totally unnecessary. Bob owns Alice’s rep if Alice agrees that that is true. This doesn’t even have to directly be publicly legible information, it’s really just between the two of them, although I will show soon how this *can* be made to cascade into large-scale cooperation. Alice *can* just arbitrarily fudge her records and decide that Bob doesn’t own her rep, but she could also just refuse to give Bob stuff in exchange for redemption of her rep, the effect is the same. This is simply a reflection of the inexorable fact that sometimes people renege on their promises, and rigid rules cannot reliably determine when that is just or unjust. *Repnet is truly epistemically decentralized[2]—that is its strength, and a bullet worth biting.*

Basic Diagrams / Syntax / Concept Stuff

One can visualize a graph of the rep between “accounts.” Each account is identified by a cryptographic public key, and there may be any sort of M:N relationship between accounts and people / scripts / organizations / ventures / rogue AIs / tulpas / etc.

1 Bob could also warn his friends that Alice doesn’t honor her commitments—I leave that to future work because it’s both tricky to get right and not strictly necessary for a minimum viable operationalization of this.

2 This is again a phrase which I invented and have not yet figured out a precise definition for, however it implies, among other things, that it is I-Confluent in the sense defined by arXiv:2012.00472.

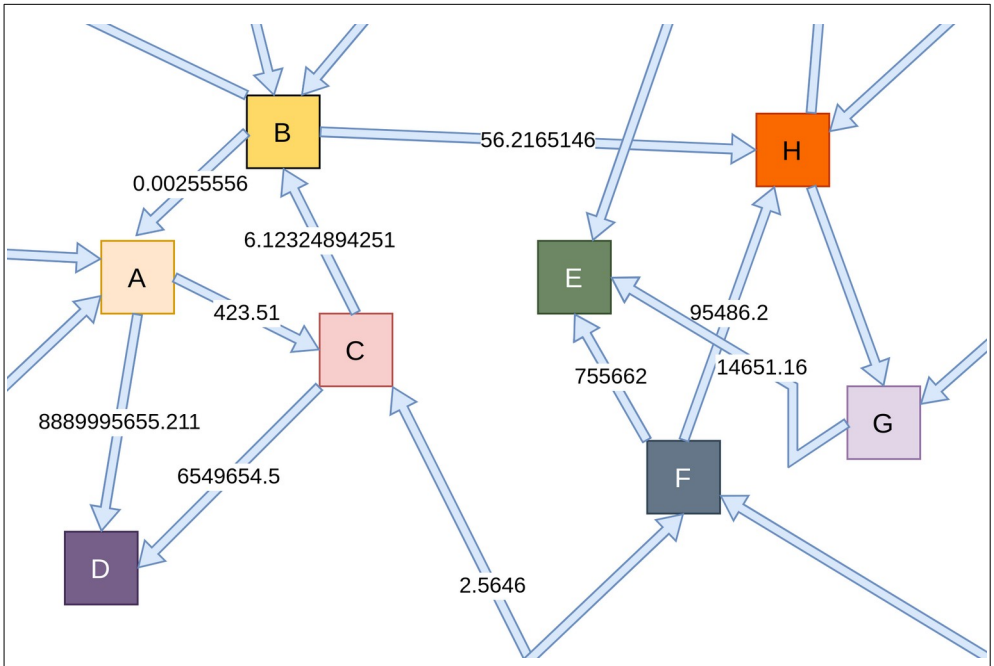

Figure 1: Example Graph

Each link in the graph represents the current tally of rep between them. We draw this as an arrow pointing one way or the other. The arrow's direction, which of the accounts it's pointing to, indicates who is currently owed by the other. So, the arrow points to the one that has the other's rep. The arrow's magnitude, some positive real number, represents how much is owed, how much rep is owned.

Consider this starting graph, with two accounts, Alice and Bob, who have never interacted:

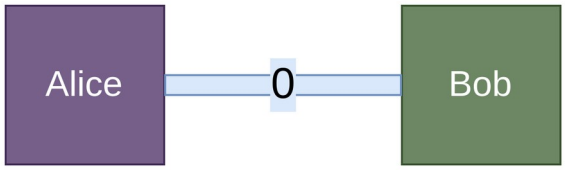

Figure 2

The tally between them is 0, the default. Now, let's say Alice buys a loaf of bread from Bob for 3 rep. Alice gives Bob 3 rep, so now there is an arrow pointing from Alice to Bob with a magnitude of 3:

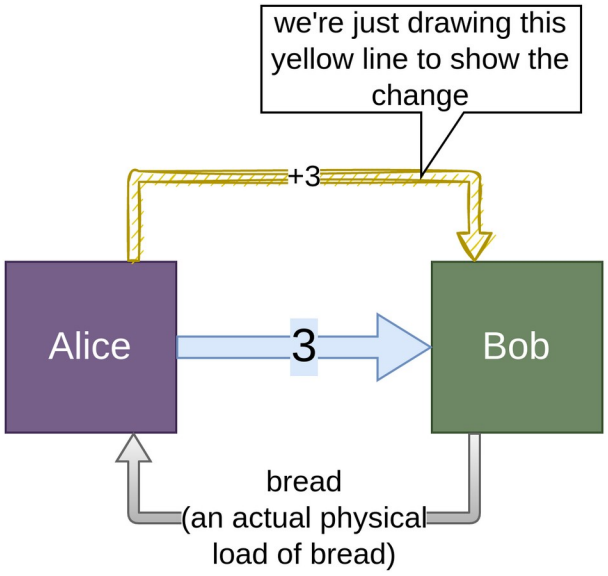

Figure 3

The tally from Alice to Bob, $t_{\text{Alice}\rightarrow\text{Bob}}$, is now 3. As such we say that the tally the other way around, from Bob to Alice, $t_{\text{Bob}\rightarrow\text{Alice}}$, is -3. Alice has agreed that she owes Bob 3 units of valuable stuff, of units specific to their particular pair of accounts and not directly commensurable to values between other accounts. Possibly, outside of repnet's protocols, Alice and Bob share some understanding of the category of stuff and the survival conditions of this commitment.

Let's say, then, that Bob buys a candlestick from Alice for 5 rep. Now, the arrow has a magnitude of 2 in the opposite direction:

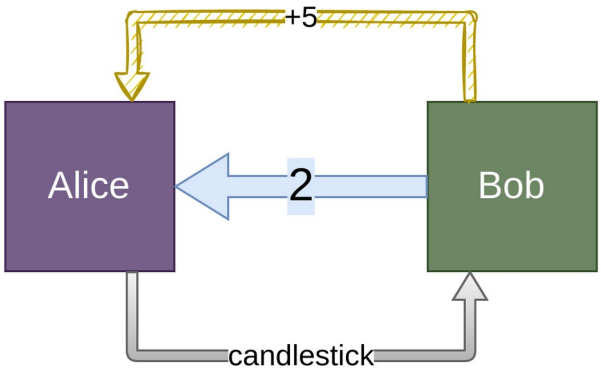

Figure 4

In agreeing to this exchange, they agree that the candlestick is worth enough to make up for the loaf of bread and also two thirds more, which Bob commits to pay back to Alice somehow in the future. The

tally from Alice to Bob, $t_{\text{Alice}\rightarrow\text{Bob}}$, is now -2. As before, the tally the other way around, from Bob to Alice, $t_{\text{Bob}\rightarrow\text{Alice}}$, is now 2.

7. Rep & Groups of 3 People

This system, as described so far, is basically just a description of direct reciprocity with quantified accounting. This alone is not enough to be an alternative to existing currencies, because trade is restricted to pairs of people who have built direct personal relationships with each other. An alternative to money needs to facilitate indirect reciprocity.

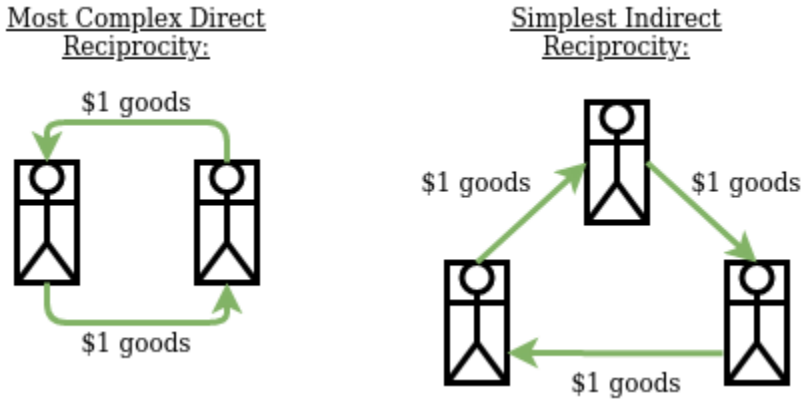

Figure 5

However, a way of supporting indirect transactions naturally falls out of the incentive structure of this system, so long as the protocol facilitates it and users are comfortable with the process.

First, consider a case where 3 people in a circle each trust the one counter-clockwise to them:

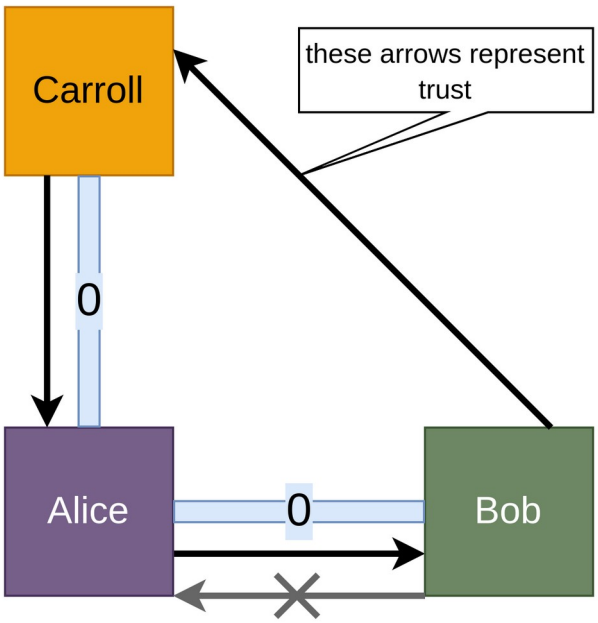

Figure 6

Bob trusts Carroll to let him redeem her rep in exchange for valuable stuff, and thus considers her rep valuable and is willing to sacrifice for it. The same relation holds from Carroll to Alice, and from Alice to Bob. However, Bob does not trust Alice, and Alice does not own any of Bob rep.

Now, say Alice would like to buy a loaf of bread from Bob. In the previous scenario, Bob trusted Alice, so Bob allowed Alice to essentially buy it "on credit" by giving him rep past the point where the arrow starts pointing at him. That of course won't work here.

Instead, Alice can first initiate a three-part circular exchange of rep as such:

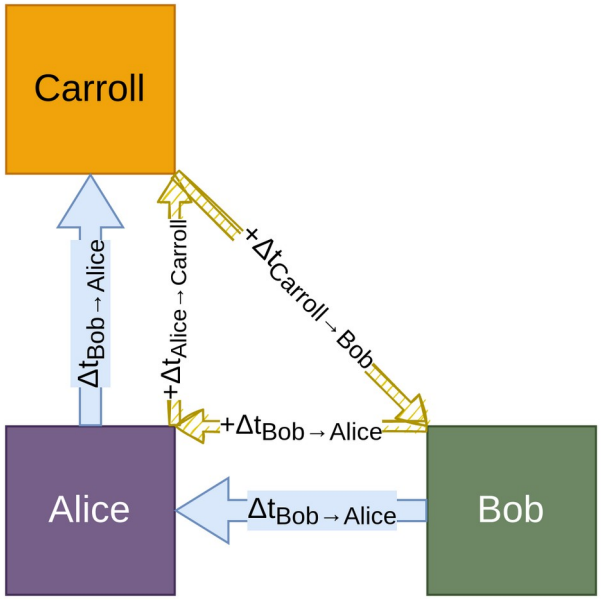

Figure 7

Carroll trusts Alice, and thus is willing to give Bob $\Delta t_{Carroll \rightarrow Bob}$ rep in exchange for receiving $\Delta t_{Alice \rightarrow Carroll}$ rep from Alice. Similarly, Bob trusts Carroll, and thus is willing to give Alice $\Delta t_{Bob \rightarrow Alice}$ rep in exchange for receiving $\Delta t_{Carroll \rightarrow Bob}$ rep from Carroll.

The most important effect here from the perspective of Alice is that, whereas initially $t_{Bob \rightarrow Alice}$ was 0, it now is some positive value. Carroll has arranged this for Alice in exchange for Alice committing to pay her back for it.

When Alice initiates this transaction, she fixes $\Delta t_{Bob \rightarrow Alice}$ to the amount of rep for which Bob is selling that loaf of bread. Based on that, the system is used to figure out the minimum $\Delta t_{Alice \rightarrow Carroll}$ and $\Delta t_{Carroll \rightarrow Bob}$ values sufficient to make Carroll and Bob willing to go through with the deal. Naturally, this means that economies with more trust are more efficient.

Anyways, Alice has now achieved this situation:

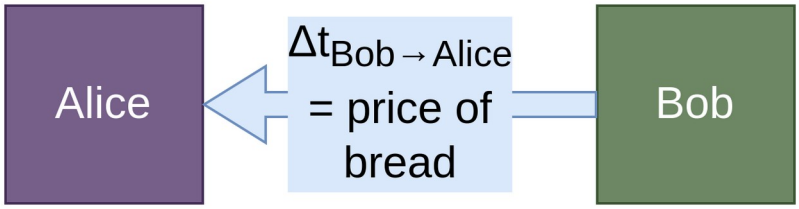

Figure 8

Which means that Alice can simply buy Bob's loaf of bread by redeeming his rep:

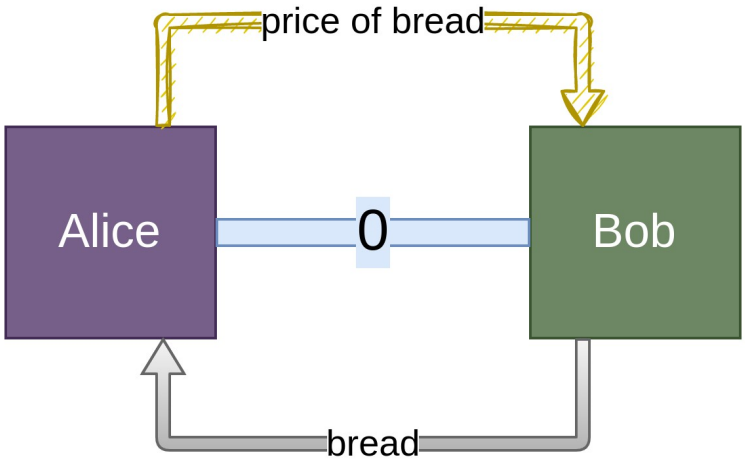

Figure 9

It doesn't matter that Bob doesn't trust Alice. Bob has committed to somehow repaying Alice, and now Alice is coming to collect on that promise, in the form of physically acquiring a loaf of bread. Alice doesn't need Bob's trust to spend rep that's pointing towards her, that represents promises he's made to her. Alice only needs Bob's trust to go into the negative and buy things on credit, which she's not doing here.

Of course, Bob could betray Alice and just not give her the loaf of bread. But we've presupposed that Alice trusts Bob, so she's betting that he won't do that.

A Note on Automated Trading

While the possibility of transactions do emerge from the incentives at play, it certainly wouldn't be practical to pull in one's acquaintances for trade negotiations every time one buys a loaf of bread. As such, most applications of this system would basically require individuals to run some sort of bot to automatically agree to agreeable 3-part transactions.

How to make this practical enough for widespread adoption is a tricky and open problem, and essentially a user-experience problem, with lots of room for variation between domains. Later on, this

article will show how repnet can be applied to peer-to-peer systems to mediate the trade of computer work-time such as a node's network bandwidth at some instant—an application of repnet where this problem is relatively simplified.

However, in one possible world I am imagining, using repnet to buy bread might look something like this. Alice and Bob use their phones to exchange public keys, possibly by one scanning a QR code on the other. Their phones use a peer-to-peer network to detect that the cheapest way for Alice's bot to purchase her the bread at the price Bob set in his bot is through Carroll's bot. Alice confirms the transaction and it is performed.

8. Rep & Groups of N People

Now that we have established this way to mediate transactions across length-2 pathways of indirect trust, we can simply chain this process to mediate transactions across length-n pathways.

Consider a situation like this:

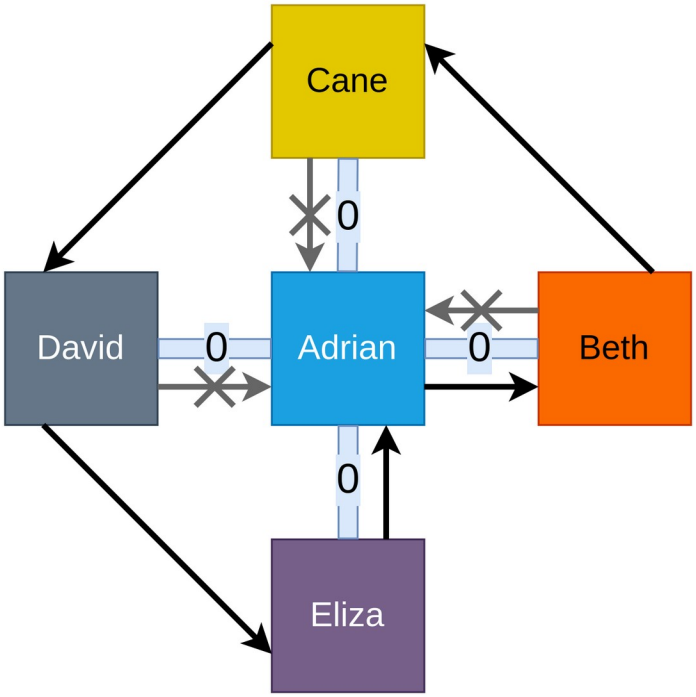

Figure 10

Adrian wants to buy a loaf of bread from Beth, but owns none of Beth's rep and cannot do so "on credit" because Beth does not trust Adrian. Nor does Beth trust someone who trusts Adrian or whose rep Adrian owns. However, Beth does trust Cane who trusts David who trusts Eliza who trusts Adrian.

What that allows Adrian to do is initiate 3 different 3-part transactions as such:

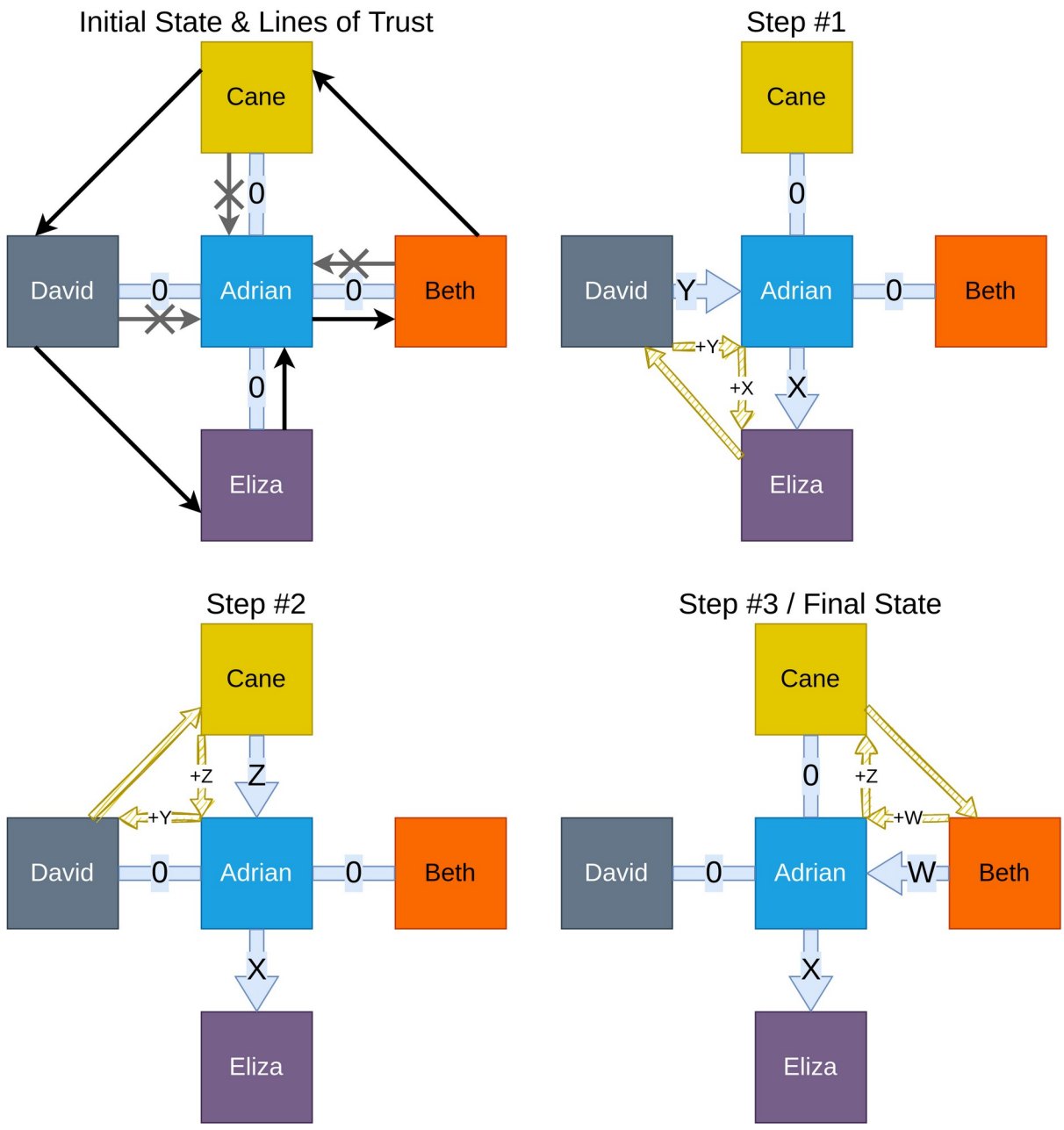

Figure 11

Adrian begins by fixing $\Delta t_{\text{Beth} \to \text{Adrian}}$ (W) to the amount of rep for which she's selling that loaf of bread, then uses the system to set the rest of the Δt values (X, Y, Z) to the minimum amount which makes all other parties willing to go through with the deal. The intermediate parts all cancel out from Adrian's perspective, as $\Delta t_{\text{Adrian} \to \text{David}}$ in step #2 equals $\Delta t_{\text{David} \to \text{Adrian}}$ in step #3 (Y), and $\Delta t_{\text{Cane} \to \text{Adrian}}$ in step #3 equals $\Delta t_{\text{Adrian} \to \text{Cane}}$ in step #4 (Z). The final result is that Adrian has now acquired Beth's rep in exchange for committing to pay back Eliza for it.

Like Alice in the previous example, Adrian has now achieved this situation:

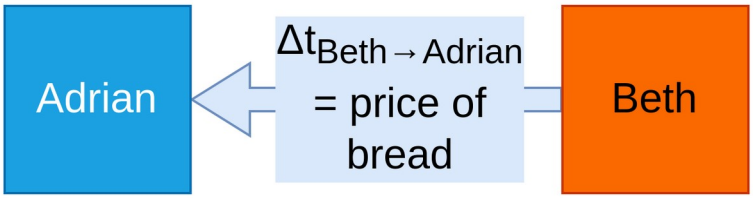

Figure 12

So, like in the previous example, he can simply buy Beth's loaf of bread by redeeming her rep:

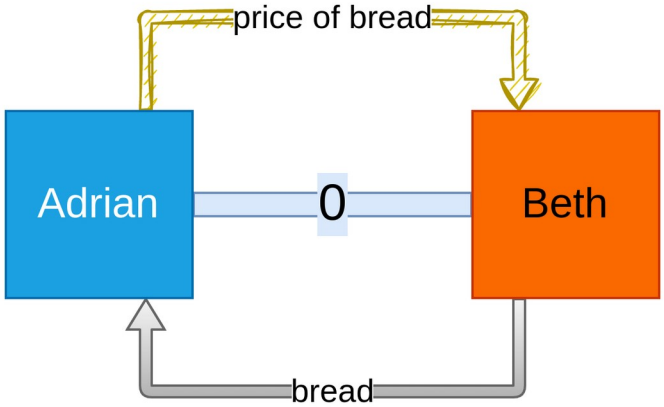

Figure 13

This process can be extended to arbitrarily long chains of indirect trust. So long as there exists some pathway of trust from person B to person A, person A may use that to effectively buy things from B by making future commitments.

Of course, assuming the amount of trust at each step is less than 100%, the incompleteness of trust at each step more or less multiplies together, creating an additional cost which person A internalizes by having to give more rep to the person who directly trusts them. This is similar to how higher risk loans have higher interest rates, and the consequence is naturally that economies with more trust are more efficient.

9. Transitivity of Trust Done Right

One might raise concerns about the possibility of betrayal in this process. For example, consider this alternative version of the previous example where it goes wrong:

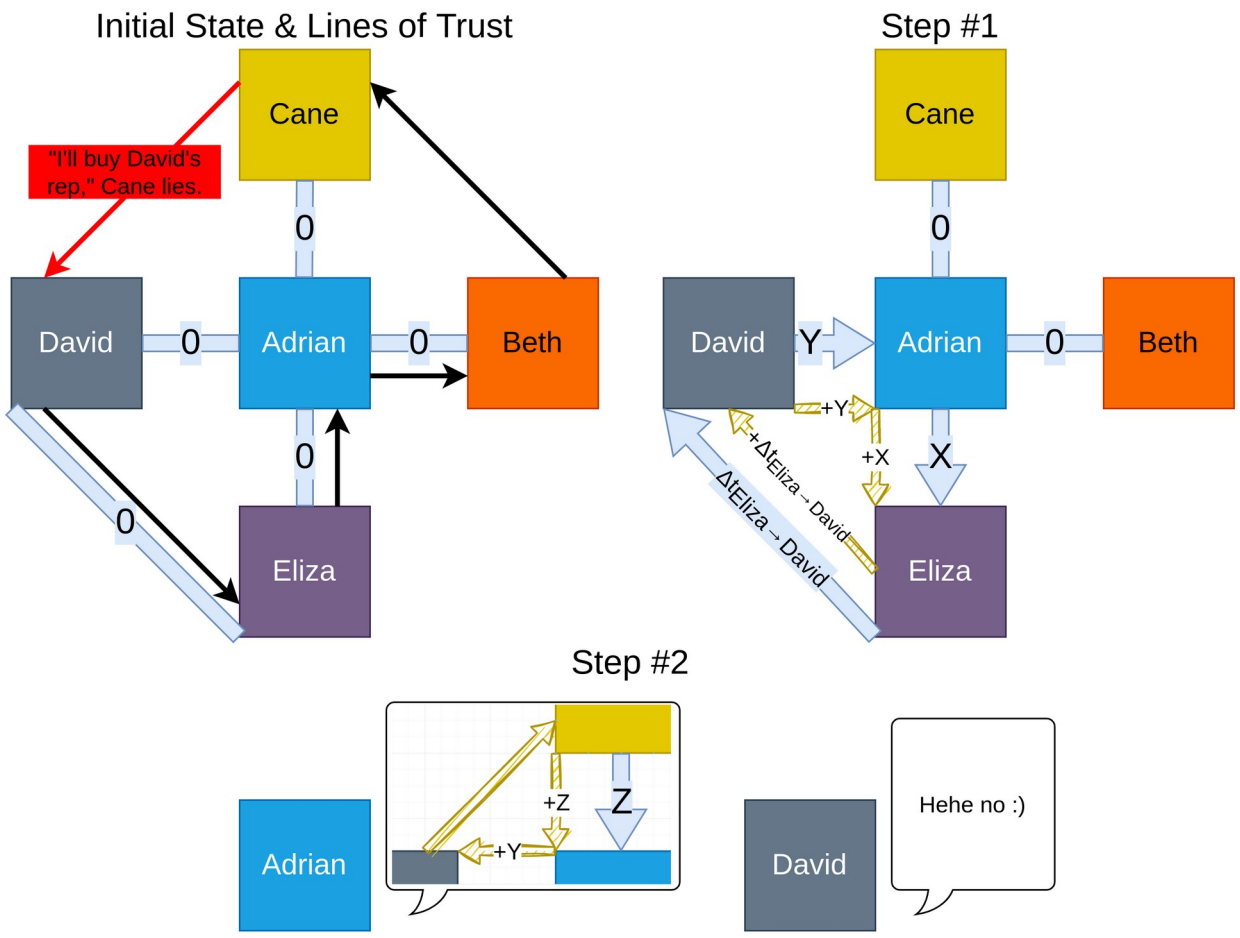

Figure 14: Betrayal (motivation for system to prevent betrayal)

Here, David and Cane are presumably either conspiring or the same person. As usual, David claims that in exchange for Y rep he'll pay Cane to give Adrian Z rep (what would normally happen in step #2). As usual, Adrian pays Eliza X rep to pay David to give Adrian Y rep (step #1). However, when Adrian tries to redeem David's rep (for step #2), David refuses to do what he said he would.

The result of this sequence of events is:

- David has promised to give Adrian valuable stuff, but he's not gonna follow through, so that's worthless.
- Adrian has committed to give Eliza valuable stuff, so Adrian experiences a loss from this.
- Eliza has committed to give David valuable stuff, so David/Cane (they're conspiring or the same person) profits from this.

For them to pull off this scam, they still have to legitimately earn Beth's trust towards Cane. That doesn't make the scam non-viable though, a scam very much could work by earning the trust of highly trusted people in order to scam others.

One could argue that this isn't a problem that needs solving. One could argue that Adrian trusting Beth necessarily entails Adrian trusting Beth's trust in further parties. They could then conclude that Adrian,

in response to David betraying him, should simply adjust his trust in Beth to a lower value, and that that sort of stigmergic recalibration would sufficiently address the problem. I'm attacking this strawman because I believe it's the exact same blind spot embodied in various peer-to-peer transitive trust network projects in the early 2000s that failed to take off. This blind spot, present in market non-understanders in general (that is, anyone who is not a left-wing market anarchist), is that it fails to mediate and incentivize the accurate transmission of information which is necessary even for totally reliable or even selfless people to act intelligently.

The fact that Cane could honor his commitments to Beth specifically, thus earning Beth's trust, but betray other people who trust him through Beth, thus profiting, and do so without Beth lowering her own trust in him, precisely demonstrates the weakness of that approach. Even if Beth is trying to act honorably, she simply doesn't know that Cane is scamming others through her. If the system's solution to this is simply for people to realize they can't reliably purchase things from Beth, that's bad for everyone who benefits from what Beth creates.

Adrian could complain to Beth that Cane betrayed him, but if it just comes down to a he-said she-said situation, that's not actionable without being abusable, especially between bots.

There's a relatively simple way to solve this problem. Instead of the intermediate accounts just telling the initiating account that they *would* go through with this deal if the necessary condition is met, they cryptographically sign a message saying that they *do* go through with this deal, *conditionally*, such that *it can be considered to have been done* if they are presented proof that that the necessary condition has been met.

As part of the computer protocol by which accounts trade with each other, we introduce a standardized type of cryptographically signed message, which we call a "give-if-receive" message.

It has these variables:

cause_account: A public key.
cause_amount: A number.
cause_purpose: Some sort of unique ID.
cause_expiration: A real-world time.

effect_account: A public key.
effect_amount: A number.
effect_purpose: Some sort of unique ID.
effect_expiration: A real-world time.

mediator_account (public key) + cryptographic signature of message from that key.

The semantic meaning of the message is as such:

> If this message is combined with a ticket
> that I can present to [**cause_account**] by [**cause_expiration**]
> to obtain [**cause_amount**] of their rep for [**cause_purpose**],
>
> then the combination of this message and that ticket itself serves as a ticket
> which [**effect_account**] can present to me by [**effect_expiration**]
> to obtain [**effect_amount**] of my rep for [**effect_purpose**].
>
> - [**mediator_account**]

Essentially, a “give-if-receive” message can be thought of as meaning “if [**cause_account**] gives me [**cause_amount**] rep then I give [**effect_account**] [**effect_amount**] rep,” with some additional complications about purpose IDs and expiration times.

The “ticket” aspect of this works recursively. As a base case, a ticket could just be type of message which we’ll call a “give” message, with this semantic meaning:

> This message serves as a ticket
> which [**receiver_account**] can present to me by [**expiration**]
> to obtain [**amount**] of my rep for [**purpose**].
>
> - [**giver_account**]

Essentially, a “give” message can be thought of as meaning “I give [**account**] [**amount**] rep,” with some additional complications about purpose IDs and expiration times.

As such, as an example recursive case, a ticket by which A gives B $\Delta t_{A \to B}$ rep for purpose X could be a combination of:

1. A “give-if-receive” message signed by A saying:
   “If C gives me $\Delta t_{C \to A}$ rep for purpose Y
   then I give B $\Delta t_{A \to B}$ rep for purpose X.”
2. A “give” message signed by C saying: “I give A $\Delta t_{C \to A}$ rep for purpose Y.”

Just these two types of messages are sufficient to essentially form a linked list structure of arbitrary length. You could, for example, have a “give” from A to B, and “give-if-receive”s from B to C, C to D, D to E, E to F, and F to G (with the amounts and purpose IDs all matching up and the expiration time not yet passed). When combined together, it would form a ticket by which F gives to G. Furthermore, recursively nested within that would be tickets by which E gives to F, and D to E, and C to D, and B to C, and A to B. However, if the first part, the proof that A gives to B, were not produced by A, then *none* of those tickets would be “activated” so to speak. As such it is, in a way, all or nothing.

Let’s make some diagrams for this.

First, let's look at the rep transfers in our previous example. Also, let's show it with the pairs of equal and opposite transfers removed. This shows only the transfers that are actually important to achieve in the end.

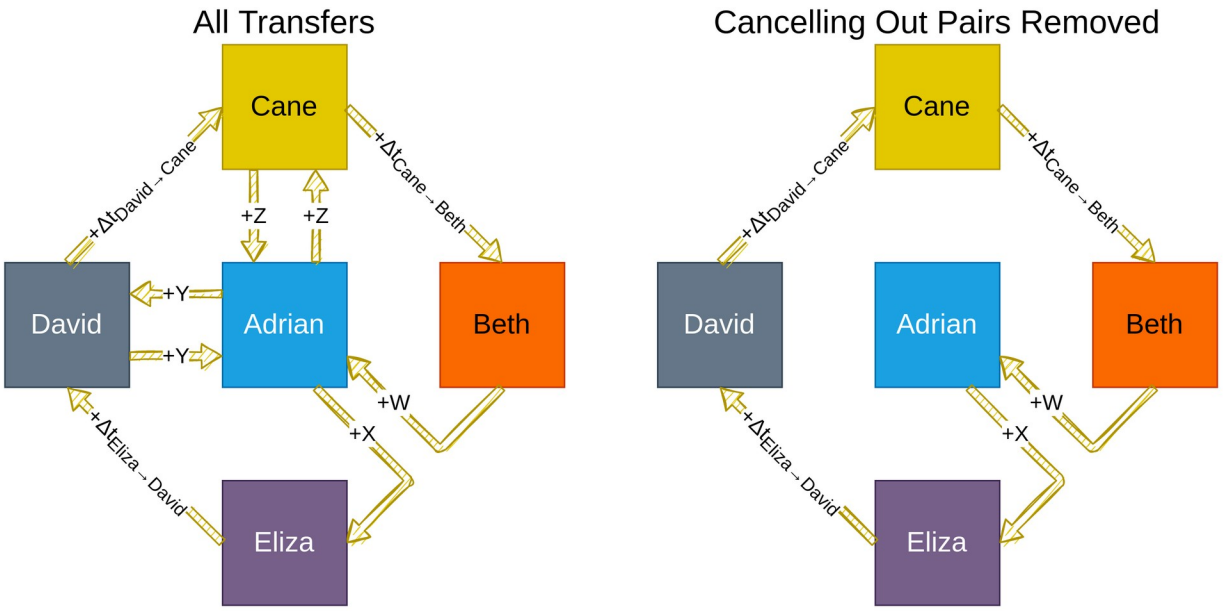

Figure 15

In our new, betrayal-handling way of doing this, instead of Adrian simply attempting to find lines of trust in some nebulous way, Adrian will actually collect from all the others (signed) "give-if-receive" messages as such:

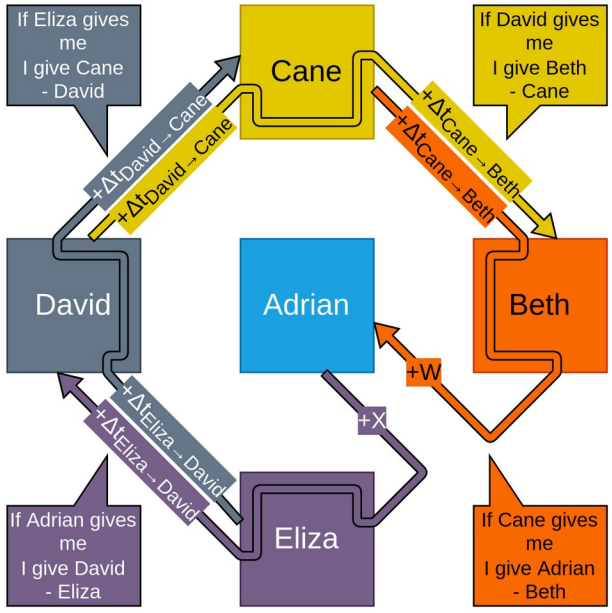

Figure 16

Eliza signs a “give-if-receive” message saying “if Adrian gives me X rep then I give David $\Delta t_{\text{Eliza}\to\text{David}}$ rep.” This is visualized by an arrow of the same color as Eliza’s account going from Adrian, through Eliza, and to David. Equivalent relationships hold with Eliza David and Cane, David Cane and Beth, and Cane Beth and Adrian.

We assume that the purpose IDs are made to match up and the timestamps are given unproblematic values.

Once Adrian has collected all 4 (signed) “give-if-get” messages, he produces his own (signed) “give” message, and combines it with the others:

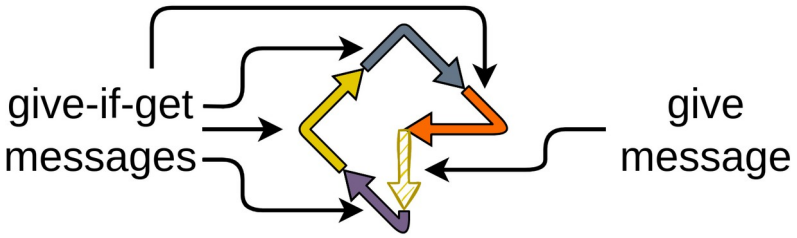

Figure 17

By doing so, this chain becomes “activated” in the sense that it now has the capacity to trigger transfers of rep by being presented to certain accounts by other accounts.

Adrian sends this ticket to Beth, resulting in Adrian and Beth both considering the rep to be transferred.

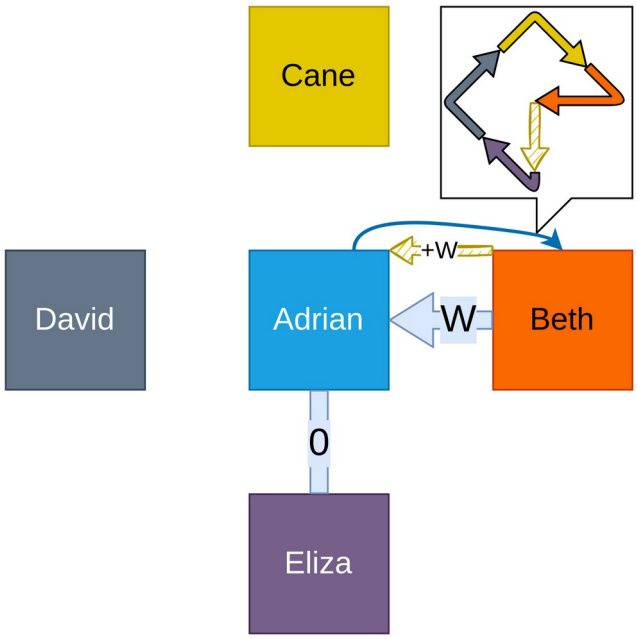

Figure 18

Beth, in turn, will remove the head from this ticket and send this nested ticket to Cane, causing Beth and Cane to consider the appropriate amount of rep between them to be transferred (part #1). She will do this because it benefits her, and if she neglects to do this, it is not a problem for anyone except her, she internalizes the full costs of it. For the same reason, Cane will then do the same to David (part #2), then David to Eliza (part #3), and finally, Eliza to Adrian (part #4).

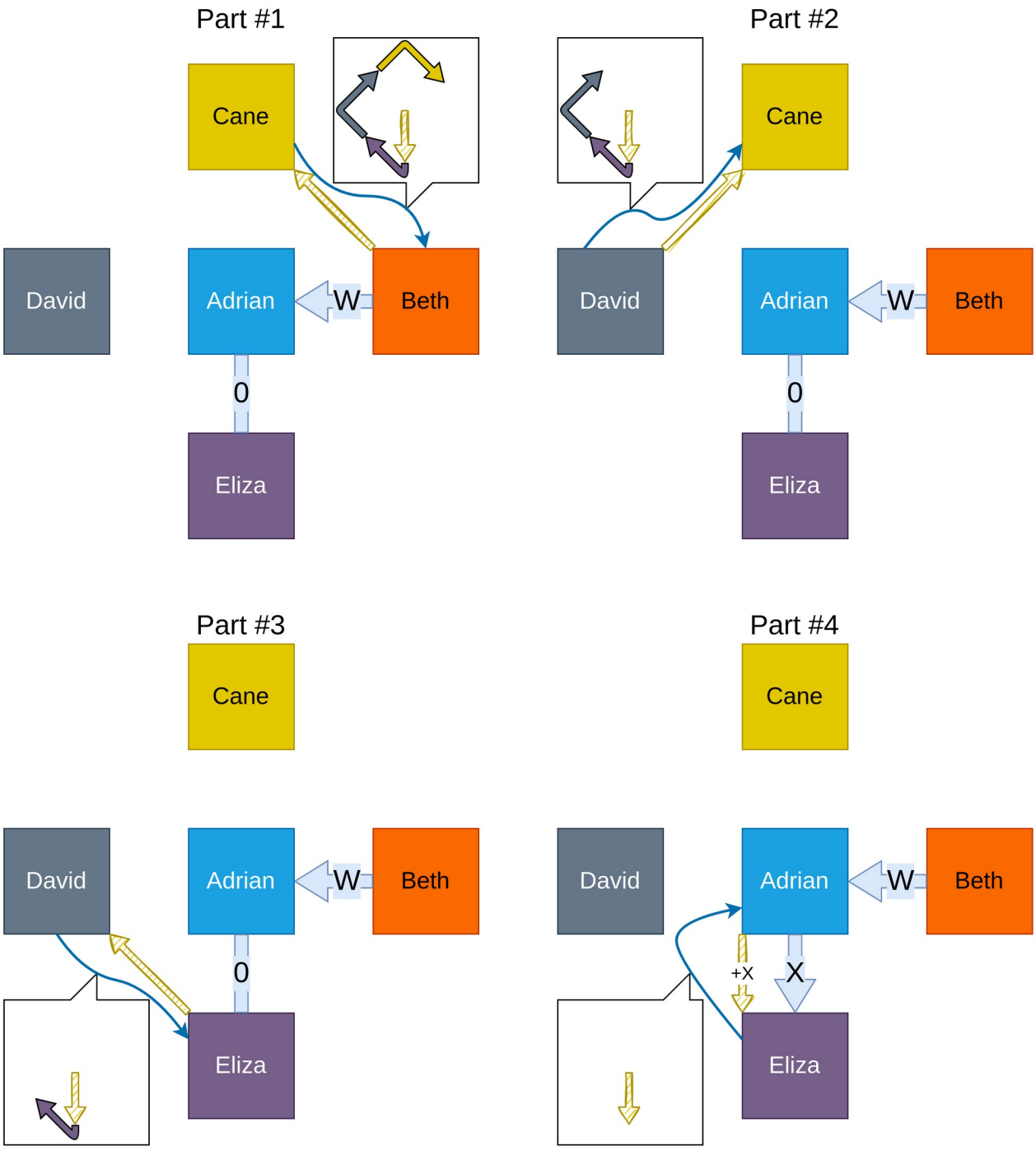

Figure 19

Part 3: Repnet In Other Peer-to-Peer Systems

10. Repnet Reverse Proxy

In this section I will describe an example of how a repnet-based system can be constructed as an enhancement to an existing peer-to-peer system, such as a file sharing system like IPFS. This is possible to do in a way that is backwards compatible with existing protocols, and can even be implemented as a network reverse-proxy that can be placed in front of an existing server without requiring modification to the server's code.

I'll speak for now about the underlying peer-to-peer system in a rather abstracted and simplified way, with a fair amount of hand-waving. We imagine there's nodes which can talk with each other over a network. We imagine each node is associated with a repnet account (which, I reiterate, is not an account in a centralized system, but rather the holder of some sort of cryptographic identity which is incentivized to behave in a self-coherent agent-like way). We don't think too much about the trading bots for this part—we assume some other subsystem is doing all that stuff simultaneously.

For this simple example, we'll say that the only finite resource we care about is network-time. We assume that our nodes ascribe the same cost to each byte they transmit to other nodes, and that each node ascribes the same value to each byte it requests and receives from other nodes. We'll hand-wave away concerns about nodes intentionally or negligently transmitting low-quality or fake responses; in some applications, like content-addressable storage downloads, cryptographic proofs can easily be used to verify the legitimacy of a response; in other applications, like distributed hash-table finger table traversal, more sophisticated mitigations are I think possible to develop.

11. Stream Prioritization

So, we have a local node, and remote nodes can request files from it. Our local node reads in this request, and then starts sending the file's content down a stream to the remote node until the file is completely transmitted. This is extremely standard.

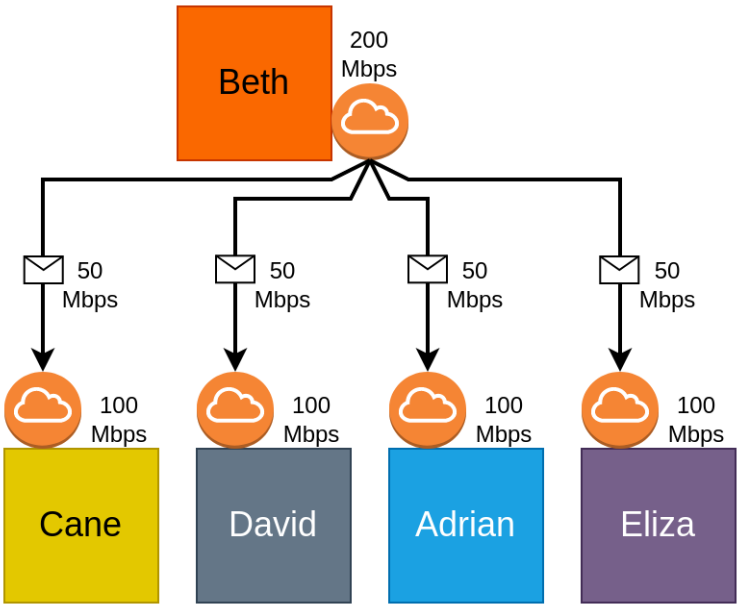

Figure 20

In the scenario seen above, there are 4 files being downloaded from our local node, Beth, simultaneously. The remote nodes are collectively capable of receiving 400 Mbps of data. However, Beth is only capable of transmitting 200 Mbps of data. The default way this would work with existing servers is that the network traffic would be split up more or less fairly. So, Beth would transmit 50 Mbps to each stream.

To integrate repnet into this system, we'll put a reverse proxy between our local node's application server and the internet. This proxy will be capable of *prioritizing* the network streams of remote nodes. It may still transmit bytes into the internet as fast as the network interface allows, however, if multiple requests are active simultaneously such that the local node has to make tradeoffs in terms of which stream to transmit bytes down in a given instant of network-time, the proxy will relay bytes on the streams which are higher priority first.

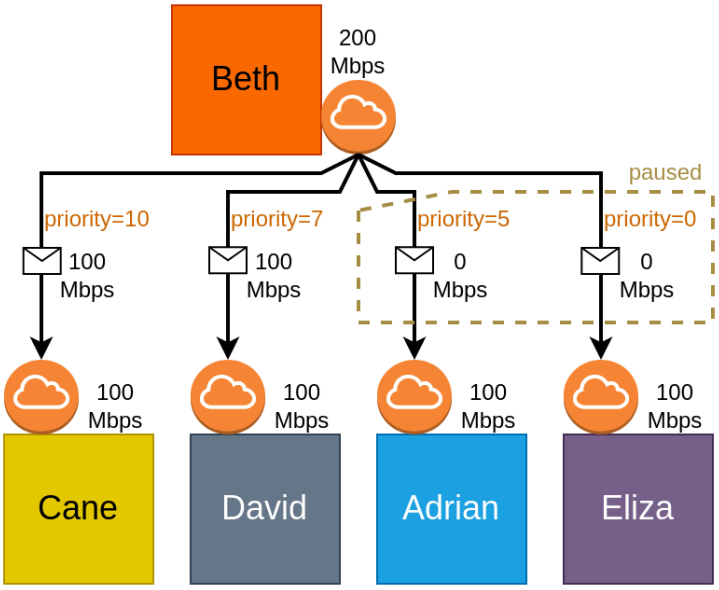

Figure 21

In this figure, the local node's repnet reverse proxy has assigned relative priorities to the streams. Since the local network interface is only capable of transmitting 200 Mbps but each remote network interface is capable of receiving 100 Mbps, the local server can only optimally serve 2 such clients at a time. As such, whereas before the server transmitted 50 Mbps to all clients, now, the server transmits 100 Mbps, the maximal possible rate, to the 2 clients with the highest priority, and effectively pauses the network streams for the other 2 clients. Only once Cane and David finish downloading their files will Beth move on to using her resources to transmit the files to Adrian and Eliza, assuming nothing else changes.[3]

12. A Utility Function

I'm getting to how the repnet reverse proxy assigns priorities to different streams.

Each account maintains a database which tracks, for all peers it's interacted with, the tally *from* the peer *to* itself—that is, how much the peer owes the node. Furthermore, the database tracks, for each peer, a variable known as the "total," which is the *total* amount of value that the peer has ever provided to the node. The total generally only goes up, not down, whereas the tally goes up and down as the peers trade back and forth. The tally is zero-sum, and is generally an arrow going towards one or the other, whereas the total is positive sum, both nodes' "total" assigned to the other can simultaneously get higher and higher over time.

3 That said, I recommend actually designing the proxy to spend a certain fraction of its network time, say 20% to 50%, servicing requests in a pure FIFO pattern. This serves as a sort of open-ended socialization to help nodes form new, potentially better mutualistic relationships with unfamiliar peers rather than just narrowly focusing on the peers they already know.

When a node requests content and a peer delivers it, the node adds the content size to its total for that peer and subtracts it from its tally for that peer. When a node receives a request for content from a peer, and delivers it, the node subtracts the content size from its tally for that peer, but does not alter that peer's total.

The node also configures a "trust ratio" variable for its system as a whole, a sort of "economic tuning parameter." It essentially refers to what fraction of a peer's total value delivered a node would bet on that peer delivering furthermore in the future. A conservative value may be as low as 10%, or even lower—however, in a low-risk, highly positive-sum context such as file sharing, it may be more beneficial to set it to some quite high value, such as 200%.

We define a "utility function" as a function of these variables. The utility function takes as input a peer's tally and total, as well as the node's trust ratio, and expresses a "utility" value, which is the node's expected utility for achieving those input values.

We show a slice of the utility function at a set "total" value in this illustration in red:

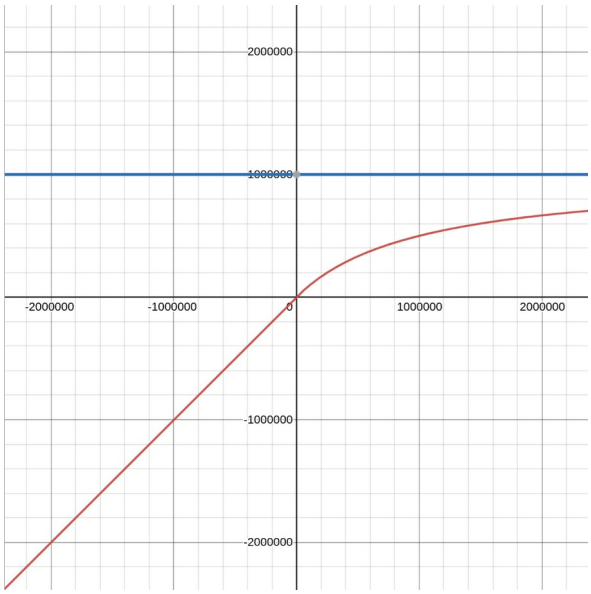

Figure 22

The X axis is tally (how much the peer owes the node), and the Y axis is utility. The blue line marks the current "ceiling" value, which is defined as the "total" value (how much value the peer has ever delivered to the node) times the trust ratio. We define utility piece-wise. For negative tally values (this node owes the peer), utility is simply the tally. This is because this node expects that it will fully repay its own debts, and as such, considers any debt incurred to have a cost exactly equal to the debt's magnitude. For positive tally values (the peer owes the node), utility asymptotically approaches the ceiling value (the amount of debt the node is willing to bet that the peer would repay). As such, the more value a peer provides to the node, the more the node expects to gain value from the peer going into debt to it, but also, the more the peer is already in debt to the node, the less the node expects to gain value from the peer going further into debt.

The fact that the utility function *asymptotically* approaches the ceiling without ever crossing it, rather than merely attenuating exponentially or something, provides resistance to a peer trying to perform

unscrupulous manipulation by going into an absurdly high amount of debt. A peer could give the node infinity rep, if the protocol supported such a thing, but the node would merely value it at the ceiling and no higher.

Furthermore, the ceiling being defined as a multiple of the total creates the nice property that, for peers the node has never interacted with before, the utility function looks like this:

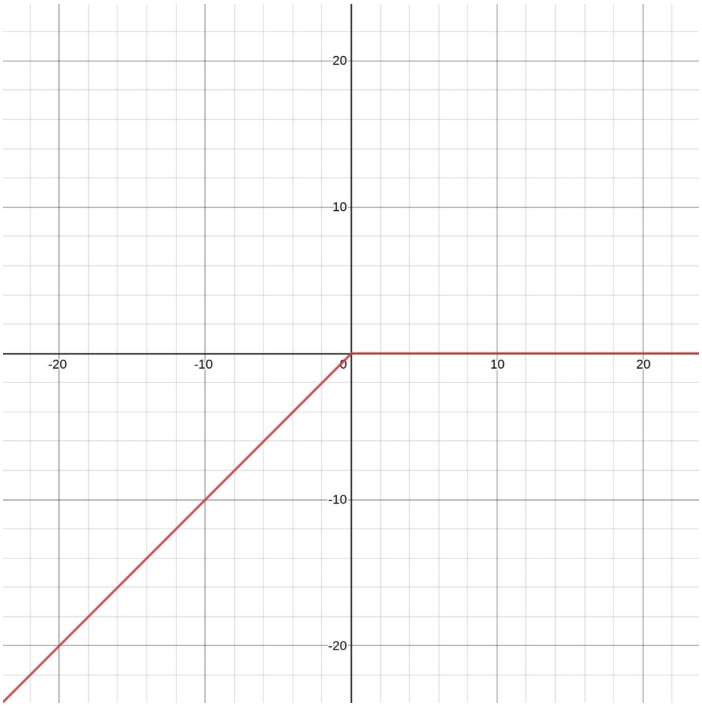

Figure 23

In this degenerate case, the total is 0, therefore the ceiling is 0, therefore the expected utility at positive tally values is 0. This means a node ascribes 0 utility to rep (promises, essentially) from peers it has never interacted with before. This provides resistance to whitewashing and sybil attacks.

13. Priority Function

The priority function is simply the slope of the utility function. That is, it is the rate of change of utility with respect to change in tally at the current total value.

As such, when the repnet reverse proxy is receiving requests, it:

- Checks whether a given request is being made “on account” for a particular repnet account. Requests not made on account are considered equivalent to requests made on account with a total of 0. How to associate this information with the data stream without interfering with the underlying peer-to-peer system’s normal operation varies from protocol-to-protocol. For example, for HTTP-based systems, this information may be placed into a HTTP header.[4]

4 See Gordon Brander’s “if headers did not exist it would be necessary to invent them” and general love for the concept of headers.

- Assigns that request's associated network streams a priority equal to the current slope of the peer's utility function. That is, naturally, it prioritizes sending that peer bytes, which increases the tally for that peer, in accordance to how much utility it expects to achieve per unit tally increase for that peer.

Here's a 3-dimensional graph which visualizes both tally and total:

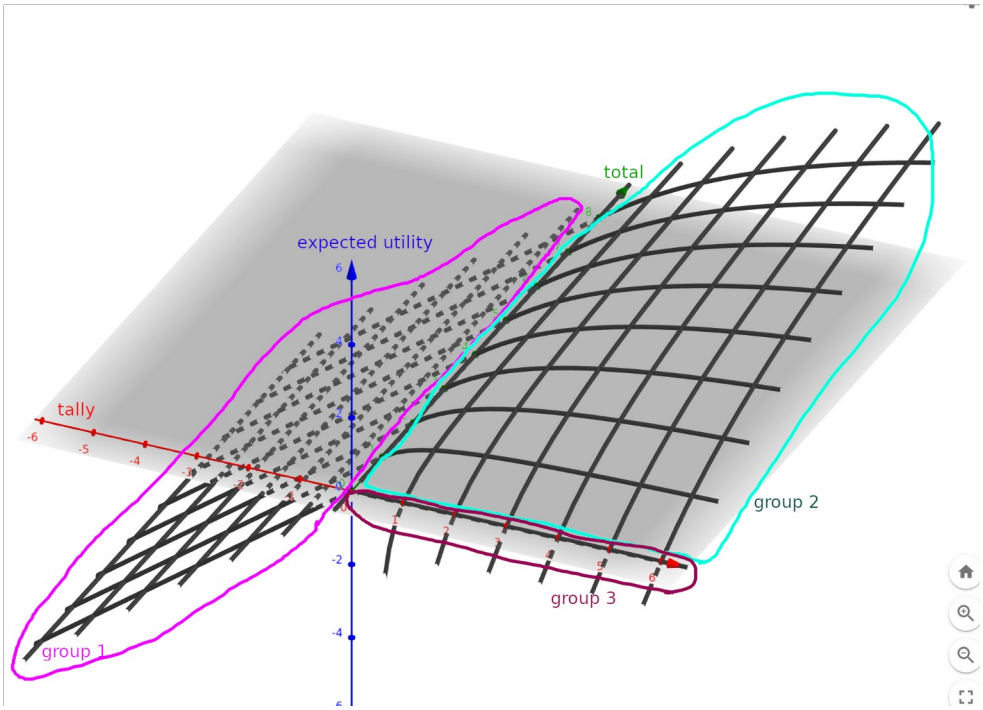

Figure 24

From this more complete view, one can see a very nice emergent property of the system. One can categorize parts of this space into 3 groups. Group 1 always has a higher slope than group 2, which always has a higher slope than group 3. These groups correspond to:

1. Group 1: This is where the tally is negative—that is, the node is in debt to the peer. This has a slope of 1. This is the highest because the node prioritizes repaying its debts.
2. Group 2: This is where the tally and the total are both positive. This has a slope between 0 and 1. This means that, the peer is in debt to the node, but the node nevertheless has received value from the peer in the past and thus thinks there's game-theoretic evidence that the peer will repay its debt, thus values the peer going further into debt.
3. Group 3: This is where the tally is not negative and the total is 0. This has a slope of 0. This means that the peer has never provided anything to the node, so the node ascribes no value to going in debt to the peer.

Another property of this utility function is that, as a peer's total approaches infinity, the utility function for the peer becomes more and more similar to the identify function. This reflects how, at the infinite limit of a node trusting a peer, the node has as much confidence in the peer repaying its debts as it does in itself repaying its own debts.

14. A Gross But Good-Enough and Straightforward Way to Prioritize TCP Streams

This part of the design has been unexpectedly problematic so far, but I think I found something that should work… fine.

Just have a static pool of threads or green threads that repeatedly drive the highest priority connection task. Make sure they occasionally get a chance to re-prioritize even if their connection wouldn't naturally yield. This is probably much easier to achieve with green threads built on an async runtime.

Or something similar to this.

If the number is set too low, you'll have local network resource under-utilization problems. If the number is set too high, you'll have problems with lower priority tasks competing with higher priority tasks for network resources.

A more sophisticated approach that doesn't rely on this gross tuning parameter would be something that's capable of distinguishing between "would block" due to limits of the local network connection versus the remote network connection. This ignores the terrifying possibility of there being all sorts of complicated semi-shared network-topological optimization problems in between the two. This might actually be more or less achievable if using some quiche-like QUIC implementation that lets you manually manage the UDP packets, whereas if using TCP, it becomes a lot more tricky, since TCP is built on raw IP packets and managed centrally by the OS. But maybe if I dug into linux's more exotic TCP system calls I could find a way to make it work.

Alternatively, it's possible one could try to take inspiration from TCP congestion control algorithms to dynamically modulate the tuning parameters towards a more optimal level.

15. This Client Side of All This

All that described above is about the server side of integrating repnet with a peer-to-peer application. The client-side part is relatively simple—if one takes for granted the rep trading system, which is not so simple.

Unmodified, repnet-oblivious clients will still work fine. The system as described is backwards compatible. However, a repnet-aware client may be able to negotiate for superior service if linked to a repnet account which is also providing services to others.

When a repnet-integrated service client, such as some sort of browser, makes a request to a server, it may:

- Somehow check whether the server has an associated repnet account. For example, if the protocol is HTTP, this may be done by making a GET request to a standard endpoint.
- Attempt to find the most efficient possible rep trading chain to make the tally from the client to the server positive.
- If such a chain exists and has an acceptable efficiency, make the trade.

If the server is behaves as described in previous sections, which it is incentivized to, this results in the user's request being prioritized above others' requests as a sort of reward for the user servicing others in the past. As such, indirect reciprocity has been achieved without any need for consensus state.

It's possible this part could similarly be implemented as a proxy on the client side, so as to avoid requiring modification of existing application software.

In an idealized environment where all nodes behave as such and a trading chain of 100% efficiency can always be found, this means that, for all the bytes of content a node serves, it can negotiate a prioritized download of an equal number of bytes. Of course, in real situations, trust is not perfect, and so nodes would have to seed some factor of bytes greater than they download. This is already considered a virtuous thing to do for torrents in the actually existing world.

16. Don't Add Latency Though

One thing I think worth noting the importance of. If this system requires one to do all this rep negotiation before the request can be made and the response can begin streaming back, it would add latency, which would make performance worse in low-load cases. That would probably drastically hinder adoption.

The solution to this is to still make the initial underlying request immediately, and start up the rep negotiation tasks which run concurrently. This requires the repnet reverse-proxy and similarly rep-integrated servers to be able to detect that the priority of a presently-happening request has increased, and re-prioritize it relative to other presently-happening requests on the server, possibly causing a change in which streams are actively streaming and which ones are effectively paused.

17. The Whole Rep Trading Thing

The main (non-boring) technical thing I haven't talked about yet is the whole, aforementioned, system for nodes to advertise which rep trades they would be willing to make and then negotiating these trades.

Here be dragons.

What I mean by that is, my biggest remaining fear for repnet is that I'll do something subtly suboptimal that bootstraps the system into social reality and wedges it into the world all sideways such that 1. some part of it is deeply wrong, some latency centralizing dynamic or otherwise problematic thing, and also 2. there is some sort of energy barrier around people escaping this situation. Because that sort of thing does often happen in tech.

Various decisions in the invention of the internet probably seemed inconsequential at the time, like making an IP address 4 bytes, which are now deeply techno-politically problematic and seemingly impossible to escape. The web started adopting the same-origin security model, and now this arguably is a similar situation. For one reason or another, evidence suggests it's no longer possible to create a new web browser or operating system.

Creating a protocol by which repnet nodes can advertise and negotiate rep trades is both a deeply complex problem with no clear "correct" solution, in many ways similar to problems of internet or

MANET routing, and an area where I can imagine these sort of subtle nightmare systemic failures creeping in over time.

I suspect the perfect immortal form of repnet would be a network of AGIs capable of open-ended ontological updating and learning and using language to transmit near-arbitrary ontological information and ideas between each other. However this is not within my capacity to create. As such I am forced to choose certain heuristics, protocols, etc. These heuristics and protocols always risk there being some sort of unfortunate emergent property including a centralizing one to their simplifications, and the protocols and approaches to integrating the world risk creating an energy barrier around people transitioning their systems away from this emergent unfortunateness.

For example, I'm imagining one scenario where everyone starts using some centralized service to manage their repnet accounts like Gmail but for repnet accounts because it's more convenient. Then it starts extending the protocol to provide integration with web frontends such that a large amount of content becomes coupled with that version of the protocol, and then this creates an energy barrier against independent third parties innovating different rep-related protocols with more advanced features, and it turns out the existing protocols have latent centralizing effects.

So, regarding this matter, I beg you to be thoughtful and perhaps a little bit cynical. And I might recommend perusing most things Gordon Brander ever wrote—such as:

- **Why did the web take over desktop and not mobile?**
- Redecentralization
- Aggregators aren't open-ended
- Composability with other tools
- If headers did not exist, it would be necessary to invent them