Synereo: The Decentralized and Distributed Social Network

Dor Konforty, Yuval Adam, Daniel Estrada, Lucius Gregory Meredith
{dor,yuval,daniel,greg}@synereo.com

March 15, 2015

Abstract

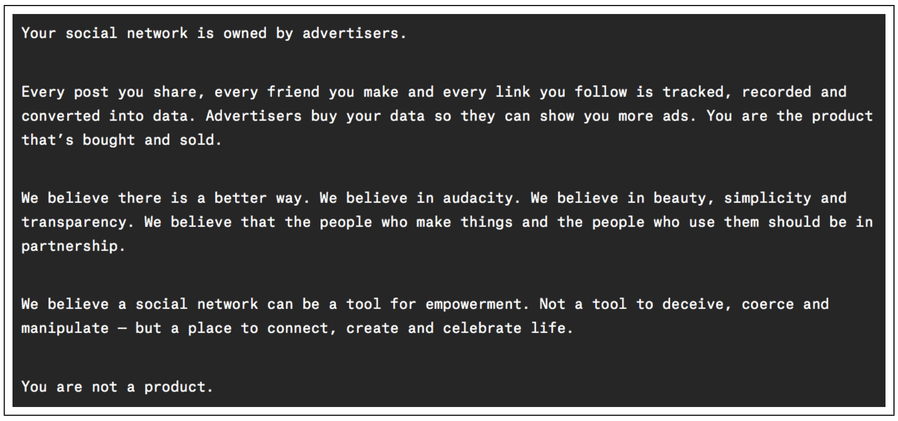
We present Synereo, a next-gen decentralized and distributed social network designed for an attention economy. Our presentation is given in two chapters.

Chapter 1 presents our design philosophy. Our goal is to make our users more effective agents by presenting social content that is relevant and actionable based on the user's own estimation of value. We discuss the relationship between attention, value, and social agency in order to motivate the central mechanisms for content flow on the network.

Chapter 2 defines a network model showing the mechanics of the network interactions, as well as the compensation model enabling users to promote content on the network and receive compensation for attention given to the network. We discuss the high-level technical implementation of these concepts based on the π-calculus the most well known of a family of computational formalisms known as the mobile process calculi.

0.1 Prologue: This is not a manifesto

The Internet is overflowing with social network manifestos. Ello has a manifesto. Tsu has a manifesto. SocialSwarm has a manifesto. Even Disaspora had a manifesto. Each one of them is written in earnest with clear intent (see figure 1).



Your social network is owned by advertisers.

Every post you share, every friend you make and every link you follow is tracked, recorded and converted into data. Advertisers buy your data so they can show you more ads. You are the product that's bought and sold.

We believe there is a better way. We believe in audacity. We believe in beauty, simplicity and transparency. We believe that the people who make things and the people who use them should be in partnership.

We believe a social network can be a tool for empowerment. Not a tool to deceive, coerce and manipulate – but a place to connect, create and celebrate life.

You are not a product.

Figure 1: Ello manifesto

The proliferation of these manifestos and the social networks they advertise represents an important market shift, one that needs to be understood in context. The shift from mainstream media to social media was all about "user generated content". In other words, people took control of the content by making it for and distributing it to each other. In some real sense it was a remarkable expansion of the shift from glamrock to punk and DIY; and like that movement, it was the sense of people having a say in what impressions they received that has been the underpinning of the success of Facebook and Twitter and YouTube and the other social media giants.

In the wake of that shift, though, we've seen that even when the people are producing the content, if the service is in somebody else's hands then things still go wonky: the service providers run psychology experiments via the social feeds [1]; they sell people's personally identifiable and other critical info [2]; and they give data to spooks [3]. Most importantly, they do this without any real consent of their users. With this new wave of services people are expressing a desire to take more control of the service, itself. When the service is distributed, as is the case with Splicious and Diaspora, it is truly cooperative. And, just as with the music industry, where the technology has reached the point that just about anybody can have a professional studio in their home, the same is true with media services. People are recognizing that we don't need big data centers with massive environmental impact, we need engagement at the level of the service, itself.

If this really is the underlying requirement the market is articulating, then there is something missing from a social network that primarily serves up a manifesto with their service. While each of the networks mentioned above constitutes an important step in the right direction, they lack any clear indication of exactly how to achieve the aims and goals they lay out. The manifestos represent an attitude, not a plan. They ask their audience to trust them, and that's not what the intended audience actually wants to hear. People want to hear how they can become engaged, how they can develop the service to support their own aims and goals.

That's why this document is not a manifesto. It's a how-to manual. True to the spirit of open source, this paper describes the Synereo attention model in enough detail that anyone with sufficient motivation can check it out or even build it for themselves. Likewise, it shows how to realize that attention model and the corresponding attention economy on a distributed network. It identifies best of breed tools, from mathematical frameworks to software stacks that make it possible to get the job done in a timely fashion with a level of quality and reliability that people have come to expect from their social media services. It also shows how these elements come together to provide new, unexpected opportunities not mentioned or even envisioned by the "manifesto networks".

We believe that this is what the market wants: not just the hackers and cryptonerds, but people who are concerned enough about the current situation that they are backing just about anything that looks like a viable alternative to the centralized incumbents. We believe that the major success of the Internet and then the world wide web was its distributed DIY approach to building and maintaining a communications infrastructure. The success of each of these developments was that ordinary people could add their computers and stand up their own websites and show the world what they had to offer. Synereo returns to that vision, and then refreshes, renews, and expands it. This paper describes how.

Chapter 1

Synereo: Design

1.1 Introduction

Social networking services are a major part of the Internet, and an important form of human interaction. The vast majority of the popular services run on centralized architecture, where the third party service provider grants privileges to the end users to use its service. In some cases the service is paid for, but in most cases the service is provided in exchange for viewing advertisements on the platform. Such services have various shortcomings which we attempt to address with Synereo.

Synereo is a decentralized and distributed social network service with a novel model defining network interactions. Managing the flow of information is critical for organizing an effective social network. Synereo implements an attention economy to address these concerns. Attention is the brain's natural method of information management, and it is a scarce and valuable commodity. Synereo treats user's attention, social feedback and activity on the network as having an inherent power to influence their reputation and the flow of content across their communities. This design is meant to give users a sense of social agency by recognizing and rewarding the value they create by engaging the network. We are building Synereo with a commitment to putting social networks back in their users' control.

The network model is based on physical principles of electrical current flow. We implement insight from the function of the neural networks of the brain to create a unique ecosystem where information is directed to where it will be appreciated and utilized well. This creates a stream of information that is most relevant to users and their communities, and that will promote the generation, curation, and engagement with content on the network.

Synereo uses AMPs as the currency for expanding the organic reach of content throughout the network. AMPs can be purchased and exchanged between users, invested in specific content to facilitate its flow through the network, or used to compensate users for their contribution to it. AMPs are also accrued for providing necessary infrastructure for the network to operate, such as storing content and providing bandwidth for sharing it. These mechanisms will be explained in further detail in the next sections.

From a technical point of view, Synereo is a distributed protocol and application. End users can run the service on their own computation devices, gaining full control over their data. Thus, Synereo is directly aligned with the shifting trends of the web from a centralized model, to a more open, user-centric, distributed architecture. As part of an effort to make the Synereo technology easily accessible to large audiences, we provide an architecture for deploying centralized Synereo web-based gateway services, such that technically-able users - or partner services - can host Synereo nodes as a service to their peers.

1.2 Background

1.2.1 The opportunity

Over the past decade, social media platforms have risen to become a major force on the Internet. As two-thirds of all Internet users are using these platforms, with 1 out of every 5 pageviews occurring on Facebook alone, and with many directly equating social media with Internet, the importance of these venues cannot be understated. [4] [5] [6] [7]

The amount of money social networks generate is staggering. Leading the pack is Facebook, earning 3.85 billion USD in the last quarter of 2014, followed by Twitter with 479m USD. [8] [9] When thinking about these numbers, its important to remember that the value created on these networks - what allows Facebook et al to generate these profits - comes directly and unequivocally from their users. [10] In fact, users and their worth is the primary parameter these connetworks are measured by. Facebook recently acquired WhatsApp for 19b USD, paying 42 dollars per user. [11] Similarly, the value of Facebook and Twitter users is often calculated through their market cap - currently at 141 and and 81.5 dollars, respectively. [12]

Faced with these numbers, many people are asking themselves, Does it make sense that the value we create simply by sharing our lives online is retained by the people who happened to be the first to provide the infrastructure allowing us to do so? That these social platforms stated aim is to increase the revenue they can extricate from us? From our basic need to communicate and share ourselves with others?

Indeed, this is how current social networking service providers see their users: as unpaid laborers. As free content creators whose behaviors can be recorded and measured, the data generated auctioned off to corporations. And for many, this may still be fine. The services given are now seen as basic necessities in our digital age, and so perhaps the balance struck between user and service provider is a fair one. However, there are other issues tipping the scale against the incumbents: theres been a breach of trust. The information going into user feeds is being manipulated, and the information going out - including details of our activity outside of Facebook - is being handed over to governmental authorities; privacy settings be damned. [13] [14] [15]

The entire foundation of established online identity is based in these axioms, of being surveilled, labeled and sold to the highest bidder. And with social network profiles serving more and more as the default identities on the web, everywhere – most services do not bother creating their own identity and authentication components – perhaps its time to stop and think if this how we want our digital identity to exist and evolve.

The more we wait, the more irreversible this becomes.

Its clear that many users are aware of these issues and are looking for other ways to network online. A simple look at different offerings in the space, aiming to subvert some of the aforementioned premises, have been met with hope and with praise before ever delivering anything substantial. Diaspora, in many ways ushering the concept of a decentralized service, was quickly backed financially by hundreds of people. [16] Ello, a recent attempt to create an environment where users arent monetized through ads, exploded in popularity within a few months of its launch, registering 1 million users and keeping 3 million more on its waiting list as it went on to scale its centralized technology. [17]

Synereo offers a new approach, one that is sustainable both technologically and financially. An approach that is based on the recognition that we are little more than money-making machines on current social networks, and turns this idea on its head. The networks we build amongst ourselves, the information we curate and spread, the social structures emerging, and the information flowing through them are all of great value that is completely unavailable to us or is cynically used to monetize us. Therefore, Synereo provides a venue where participating in the social network is participating in an economy – the attention economy, where participants benefit directly from the value they create. Further, in Synereo, our digital identity reverts to our control. We are free to construct it as we see fit and establish and utilize our reputation; its about our connections with others; about the communities we form; about the trust placed in us by our peers. Synereo, therefore, was not created only for reclaiming our financial value. Its about reclaiming our social capital.

Social agency and the attention economy

Synereo is designed for the economy of attention. But what is the value of attention? How should our networks recognize and appreciate our attention? What are we owed for the investment of our attention? We begin by noting a variety of new services that have begun compensating users for their online activity. For instance, Tsu[1] compensates users with a portion of earned advertising revenue on their content. Gems[2] allows users to compensate each other directly for messages and advertising using their own cryptocurrency. Gems developers have explicitly described the compensation model of their system in terms of the attention economy [3] Since users are paid for their time and attention, these networks are described by their developers as contributing to the economy of attention.

Compensating users for their attention is certainly better than expecting their labor for free. We welcome the move to recognize user contributions to the value of these networks, and to reward the love and effort that goes into all the content they produce. However, the issue is more complicated than simply paying people for their attention. Specifically, these networks fail to appreciate the role of agency in managing the flow of attention. The relationship between agency and attention has profound psychological, sociopolitical, and

[1]http://www.tsu.co

[2]http://getgems.org/

[3]See Bitcoin News Weekly episode 22, retrived Jan 2015 from http://shoutengine.com/BitcoinNewsWeekly/dollars-to-bitcoins-banks-hate-bitcoin-the-gold-5231.

ethical implications. It requires explicit critical reflection as we develop the next generation of social networks.

In this section, we justify the compensation model used in Synereo (social compensation - Reo, and monetary compensation - AMPs) in terms of how they assist the attention economy by making users more effective agents in the network.

Attention is selection for action

Any agent must deal with the challenge of information overload [18], [19]: there are typically more data available to the system than it can meaningfully process. Thus, agents must select among the available data for relevance in order to effectively use its processing resources. Additionally, there are typically many actions appropriate to a given data context. The agent must select among these possible outputs in light of its selected inputs. Wu [20] [21] describes attention as the solution to this "many-many problem". Attention meets "the challenge of sifting through many 'inputs' and many potential 'outputs' to generate coherent behavior."[4]

Wu claims that "attention is selection for action"[5]. An agent's attention to some stimulus demonstrates the agent's assessment that this stimulus is relevant to some activity to which the agent is engaged. For instance, if I'm playing frisbee in the park, I will usually respond to the airborne disc with behaviors appropriate for catching it. My attention is spent tracking the disc and ignoring distractions. This investment of attention serves to orient my body so that I can be in the right position to catch the frisbee as it flies through the air. If, in this process, someone yells "watch out!" behind me, I might divert my attention towards the source of the voice and lose track of the frisbee. My shifted attention represents a spontaneous revaluation of value: that this new stimulus might be more deserving of my cognitive resources than the frisbee. The warning comes to occupy my attention to the exclusion of the frisbee, and by doing so it reorients my processing resources to focus on a new input source and prepare me for a new set of actions.

In general, a user's attention is pulled to various attractors in the perceptual and behavioral space, depending on their spontaneous assessment of the relevance of these attractors to their various ongoing commitments. Since their projects can be quite diverse (ranging from catching the frisbee to staying alive), their attention system must weigh a very complex and dynamic set of criteria for resolving the many-many problem.

Put simply, attention is the system by which agents determine both their experience of the world and how they will engage it. The flow of an agent's attention indicates their continuous revaluation of value, their estimation of how their processing capacity is best spent. As Herbert Simon described [22], an economy of attention recognizes the value and scarcity of attention for the agent and the need to allocate attention efficiently. If attention is selection for action, then "allocating attention efficiently" means allocating attention in ways that help users act in the world, so that they might better realize their ends as agents.

[4]Wu (2011) p 3
[5]*ibid.*

In other words, the attention economy is about agency. Designing for an attention economy is essentially about making people more effective agents through the careful management of their attention. That means shaping both inputs and outputs in ways that reflect the user's own estimation of value. An efficient attention economy is one where users can quickly and easily resolve the many-many problem, and so are able to direct their attention in ways that effectively contribute to their projects, *whatever they might be.* An inefficient attention economy is one where these many-many problems are difficult to resolve and it becomes unclear for the agent how best to act in order to realize their goals. In either case, we can evaluate the success of an attention economy in terms of the difficulty users have in determining how to act.

Efficiency in the attention economy maximizes the user's agency; inefficiencies confound a user's agency. Note that the mere investment of time and attention is not itself evidence of inefficiency. The value of attention is goal relative and some goals are worth significant time and effort. The time I spend with an absorbing book or an awesome game is not wasted attention if it's what I want to do. The time I spend practicing an instrument has value even if I occasionally find it tedious. Value is measured not just in time and effort spent but in progress made.

Designing for the attention economy means designing systems that help their users be better agents. A social network designed for the attention economy would provide users with tools for making them effective agents in the world. This includes a felt sense of control so that users meaningfully understand the scope and limits of their agency. Mortier et al [23] describe agency as "giving people the capacity to act within these data systems, to opt-in or to opt-out, to control, inform and correct data and inferences, and so on." This sense of agency is central to the well-being of both the agent and her community [24]. Thus, from our perspective, designing an attention economy is a political and ethical imperative as much as it is a technical challenge. The importance of attention and agency in the construction of meaning is captured clearly in David Foster Wallaces 2009 commencement speech "This is Water" [25]

> ... learning how to think really means learning how to exercise some control over how and what you think. It means being conscious and aware enough to choose what you pay attention to and to choose how you construct meaning from experience. Because if you cannot exercise this kind of choice in adult life, you will be totally hosed.

Freemium attention

We can evaluate the compensation techniques of social networks like Tsu or Gems in light of their impact on the economy of attention as described above. We argue that compensation is not enough to simplify the many-many problem for the agent, and therefore does not itself address the attention economy. Instead, these networks have established a schedule of rewards that potentially complicates the decision making of their users. Reward schedules and positive reinforcement are critical to learning, but they also motivate addictive behavior and patterns of dependency that can compromise a users agency and control. The use of reward schedules and positive reinforcement to direct and manipulate attention manifests most obviously in so-called freemium games [26] [27].

Soroush et al. [28] finds empirical support for a correlation between in-app purchases in Candy Crush and lower levels of self-control.

The case of freemium gaming makes clear how schedules of rewards and reinforcement can be used to compromise or even undermine a users agency. Instead of assisting in the process of selecting for action, reinforcement learning trains the user to expect specific rewards when adopting the goals and behaviors imposed by the scheduling system. Effectively, these compensation networks redirect user attention and action to serve the purposes of the network. On these networks, users see the ads and content determined by the network administrators, regardless of how these inputs assist the user's autonomy. Compensating users for this attention doesnt enhance their agency; if anything, introducing new inputs and outputs may complicate the user's flow of attention, introducing inefficiencies in the network. Compensation orients users to the goals of the network by introducing a new set of input/output relations that the user must take into account in their evaluation of value. This may be good or bad for the well-being of the individual, but it is a distinct design goal from the issues of effective agency discussed in the previous section. Given the tendency of large, centralized social networks to compromise their user's privacy and agency, we believe the reorientation of design goals described in this whitepaper is in order.

Estrada and Lawhead [29] distinguish between systems that disrupt user behavior by introducing new computing tasks, from those that leverage existing behavior to perform useful computational work. They call the latter "natural human computation". We see Tsu, Gems, and other such compensation networks as a "disruptive" approach to social networking, and propose Synereo as a "natural" alternative. The distinction between "disruptive" or "natural" computing systems speaks to the heart of agency as it pertains to the attention economy. Although there are places for disruptive computing, we believe social networking should be as natural as possible, for the sake of the user's agency. A natural social network wouldn't complicate a user's attention by introducing irrelevant stimuli, or persuade users into performing meaningless tasks for meager compensation. Instead, present users with information relevant to their interests, and oriented for actions that are best aligned with the user's values. Of course, the activity of the users themselves define and determine those interests and values.

What would it mean to build a social network that recognizes the value of a user's activity, without simply imposing a schedule of rewards for menial tasks? How do we determine a user's evaluation of value, so we know how much their attention is worth? How do we recognize and reward (not just pay!) people for doing valuable and useful social work? What tools do users need to be better agents and build better communities?

Synereo and the emergence of a social consensus

Human beings are endlessly complex systems. Enhancing their agency isn't about dumbing things down and making them simple. It certainly isn't about telling people what to think or what to feel or how to act. Helping our users be better agents means filtering through the available information to direct their attention where it is needed most. It means presenting options for engaging content that have a clear impact on the content, the community, and the users themselves. Moreover, users should have the tools for evaluating the impact their engagement has on the network, and to use this information for deciding what actions would best realize their goals both individually and collectively. Put simply, users need to feel like they are in control of their presence on the network, from the level of data security to the level of social identity and everything in between. Existing social networks tally up likes and reshares, but they do nothing to describe the network you're helping to build or how to make it stronger. The point of these networks is to make you feel like a celebrity of your own minor fandom, not to make you feel like an agent in control of your social life. When analytics are available, they are typically offered as a premium service for advertisers, not a tool for the people to manage their web presence or organize themselves collectively.

As Bateson says, [30] "How will people learn that what they do "counts"? By counting." Synereo is designed to give users have a strong sense of their own capacities as social agents. We do this by counting. When users engage with content, they will do so understanding the impact their activity has on the network. User activity is taken to have inherent value that is represented explicitly in the network. The value of user attention is recognized by the communities they engage. As a community, our collective values emerge from this pattern of interactions. And our engagement makes a difference to the life of content as it spreads on the network.

Synereo represents value on the network in terms of Reo and AMPs. Reo is a measure of social standing, and accrues or expires naturally as users engage with the network. AMPs can be used to boost content and has a monetary value. These tools will be explained in detail in this whitepaper. Crucially, Reo can be exchanged for AMPs in a way that allows users to stake their reputation on content that matters to them. By establishing an explicit relationship between a user's social standing and the flow of content across the network, Synereo hopes to build a framework where users have a robust sense of social agency. A sense that what they do matters. A sense of a community that values their contributions and will continue to support them.

The exchange between Reo and AMPs establishes a natural mechanism by which financial reward can track social value. Users can earn AMPs by engaging the network, and can spend AMPs supporting the content they care about. But content is constrained by the shape of the network which always takes Reo into account. The result is that any AMPing of the network must play nice with the existing social values; the spreading of AMP'd content still requires community participation to propagate it.

This combination of Reo and AMPs makes the analysis of compensation on Synereo more difficult than the case of freemium games. Unlike freemium games, Synereo doesn't habituate users to expect trivial rewards for meaningless actions. Users have an active say in the value of their attention and how it gets distributed. Moreover, there is no central authority determining who sees what content. Since the flow of content depends on user activity, the network is imposing it's own reward system. Synereo moves toward a world where users are setting the terms and conditions of their own life, instead of merely adopting the reward schedule of a central network authority. We think this is critical for building self-supporting communities that can develop shared values and a sense of social agency to achieve our collective goals.

1.3 Synereo: Design goals

Synereo aims to be a social networking framework that focuses on providing users with an experience that is tailored to their preferences. As there is no third-party, central, authoritative entity acting as middleman and profiting from specific habits and types of user interaction on the network, the applications created on top of this framework are free to put their users in the center. Built on attention-economy principles, the Synereo network directs its users' attention in a way that accurately reflects their social goals. Different applications using the Synereo framework will also benefit from this user-centric prioritization of attention, providing them with the information and action types they want at just the right time and with the right context.

1.3.1 Core concepts: Reo, engagement, and AMPs, oh my!

Reo

Reo is a measure of your reputation as a publisher of attention-worthy content. Given two participants, say Troy and Abed, in an online community, we write $\mathsf{Reo}(\mathsf{Troy}, \mathsf{Abed})$ to indicate a quantitative measure of their relative standing in the community. Roughly speaking, $\mathsf{Reo}(\mathsf{Troy}, \mathsf{Abed})$ reflects how much the friends Troy and Abed have in common have paid attention to Troy versus Abed. As such, the measure is not symmetric; that is, we normally expect

$$\mathsf{Reo}(\mathsf{Troy}, \mathsf{Abed}) \neq \mathsf{Reo}(\mathsf{Abed}, \mathsf{Troy})$$

engagement

Reo is built on top of the notion of $\mathsf{engagement}(\mathsf{Troy}, \mathsf{Abed})$, which is a quantitative measure of how much Abed has engaged the content output of Troy; respectively, $\mathsf{engagement}(\mathsf{Abed}, \mathsf{Troy})$ is a measure of how much Troy has engaged the content output of Abed.

These two notions feature in the algorithm for prioritizing content in a user's stream. Essentially, the more Troy attends Abed's content, the more likely it is for Abed's content to show up in a place in Troy's stream where Troy will notice it. Likewise, the higher Abed's standing in the community relative to Troy, the more likely Abed's content will show up in Troy's stream in a place where he is likely to notice it.

The AMP Token

To complement these two forces, Synereo introduces the AMP. A user can publish content with a certain amount of AMPs attached. The algorithm for prioritizing content also takes into account how many AMPs have been invested in the content and can use this to bump it into a more desired location. In this way, AMPs provide a way to purchase a shot at a user's attention. Users receiving amplified content will also receive a portion of the attached AMPs. The more Reo they have relative to the poster, the more their attention is valued by their shared community, and the more AMPs they will receive proportionally.

Distributed ledger, genesis block and mining Notice that while Reo and engagement are calculated based on the ongoing behavior of users on the network, AMPs are a conserved quantity. They are effectively purchased and spent. The distribution of AMPs in a Synereo network is therefore kept in a distributed ledger, like the bitcoin blockchain. Similarly, when Synereo launches, an initial supply of AMPs will be available for purchase and distribution to early users and contributors. Synereo also offers a unique social approach to proof-of-work that will be connected to a kind of "mining" and AMP creation. However, the discussion of this is currently out of scope for this paper.

The attention economy

Out of these core concepts we build an economy of attention; a way to manage this highly limited resource of the human brain. In some sense, this economy is not unlike an economy in a functioning capital-based democracy. Typically, such economies come with two dials: on the one hand, citizens can express their voice through democratic processes, such as elections, referendums, and other mechanisms where they can cast their *vote*. On the other, citizens engage in the creation, distribution and consumption of goods and services and use capital as a means of facilitating the beneficial flow of this wealth.

Both processes ultimately result in the distribution of goods and services. And as long as the linkage between votes and capital exchange is sufficiently weak, citizens have *two* distinct ways of expressing their individual and collective will about the functioning of the economy in society. When working effectively, these two channels provide a mechanism for balancing the collective will (largely identified with democratic participation and self-determination at that level), with individual will (largely identified with the flow of currency and self-determination at that level).

The notion of Reo, as can be seen in its technical form in chapter 2, is essentially a kind of attention-based measure of the collective will: it looks remarkably like a kind of online voting. Meanwhile, AMPs are unabashedly an attention-based form of currency. As such, these two notions are balanced against each other and provide a means of balancing collective will with individual will. What distinguishes Synereo from the idealized capital-based democracy, however, is the reconciling force of engagement. In point of fact, a democratic society lives or dies according to the engagement of its citizenry; but there is no objective, reified measure linking this directly to the distribution and flow of value. In Synereo, there is - the attention model allows each and every individual to cast his "vote" and show his engagement without making any special effort beyond his normal participating in the network - and this additional sophistication creates more subtle network dynamics, hopefully leading to a more balanced social model.

1.3.2 User experience

Card-based thinking and designing has been an increasingly popular approach to UX in recent years, especially on handheld devices, with the shift on the web happening as well [31]. These cards are now evolving to contain not just bits of relevant information, attracting user attention to apps and sites, but provide immediate action items alongside that information. When receiving an

SMS using a modern smartphone's operating system, the "reply" and "read" buttons will appear right on the card itself, allowing the user to start and finish his engagement with the attention-requiring item in one go, through one interface. This move represents a way of presenting information to individuals that is in-line with what their brain expects, and indeed, with the way their mind operates.

When cognized information receives conscious attention, it is almost never composed solely of its sensory-perceived signals. Rather, it is subjectively experienced as information coupled with the most pertinent actions. A conscious agent becomes aware of the information after it has been subconsciously preprocessed to link it with inherent or learned actions[citation], allowing him to consciously choose and respond quickly without wasting time on the act of surveying the obvious available actions while the information inhabits the costly space of conscious attention.

This coupling of information and action, presented to conscious attention due to an urge or a need, represents a mechanism evolved to make the best use of the precious resource of attention. An urge – or a web app – presenting to a user's consciousness succinct information tied to possible actions is more likely to receive the attention and engagement it needs and strengthens the pathway between the originator of the request for attention and the user's attention-granting mechanisms. This logic forms the basis for the Synereo attention model: the more a user attends and engages with content, the easier it will be for the agent originating it to receive further attention from that user. That agent may be a friend on the network, a business advertising its services, a vote or discussion originating from a community, or a game or an application built on top of Synereo. Further developments of this model include more fine-grained distinctions between information types, social consensus, and other signals that shape the network and the flow of information and attention in it.

A central aim of Synereo is thus to bring its users units of information coupled with relevant actions so that they may attend in a way that is complete, being respectful of their time and attention. This means overwhelming them down with unnecessary input and options. We're not just talking minimalism, we're talking freedom. The Synereo architecture does not limit users to specific actions mandated by it: interactions on Synereo are open-ended. And while cards themselves play a major role in the Synereo user experience, this principle will accompany most interfaces available on our platform. Thus, we recognize the importance of the mechanism and thinking underlying this emerging trend, and give it its own name in the Synereo ontology.

Step forward and kneel young organizing principle, we dub thee Ceptron. Go forth and organize online experiences for the good of all peoples everywhere.

A $per-ceptron$ is an algorithm for supervised classification of an input into one of several possible outputs. It is based on logic gleaned from the operation of neurons and neuronal networks in brain. In a perceptron, there is a weighted connection between input and possible outputs. The decision process may be immediate and "automatic" or may involve higher-level information integration and functions, iterations on processing, and conscious attention. The perceptron algorithm dates back to the late 1950s, where perceptrons were implemented as the building blocks for the first neural networks. The new term we propose, the "Ceptron", symbolizes the informational space that contains both input information ("stimulus") and all possible output information ("response)". This space, presented to the user as an information/action coupling on Synereo, allows him to immediately engage with, and act on, the message received through the specific subset of actions suggested to him - or pause, invest mindful attention, and travel outside of the proposed pattern.

The types of ceptrons available and the actions within them will continually evolve and improve to represent a community effort to maximize the utility offered by their online social network, maximizing and aligning individual and social agency. This is how we become a community of autonomous, networked agents.

> "Between stimulus and response there is a space. In that space is our power to choose our response. In our response lies our growth and our freedom."

– Viktor E. Frankl

Queuing, priority, and attention

A key organizing feature of information on Synereo, directly shaped by the attention model, is the prioritized queue of items. Items transmitted to you on Synereo compete for your attention and are queued and aggregated based on your history of engagement with content and the people who originate it, your interests, preferences, time and location, your friends and communities, and so on. These are not machine-learned details about you; rather, these attributes manifest themselves into the shape of the network, affecting the way information flows to you and how it is prioritized for you.

The priority of items in the queue is determined by the amount of current these items reach you with. As information travels the network, current decreases the farther away it is from its point of origin. Each post you make starts with a capacitor full of electrical charge that is sent out across the wires of the network as electrical current. The amount of current you may send out to a peer on the network depends on:

- Your differences in Reo score
- The peer's engagement score with you (how "far away" he is from you on the network)
- How the other peers you've broadcast to have engaged with this specific post
- The amount of AMPs invested in the post
- How often you post

Current is only "consumed" if a user is exposed to the post in her stream. As long as the post is not seen, its charge is free to be offered to users, and that offering may change based on other users' engagement with it while it is waiting in queue or if the user is offline.

Let's describe a simple situation on Synereo and see how these different considerations are involved in shaping the experience of a user going through a

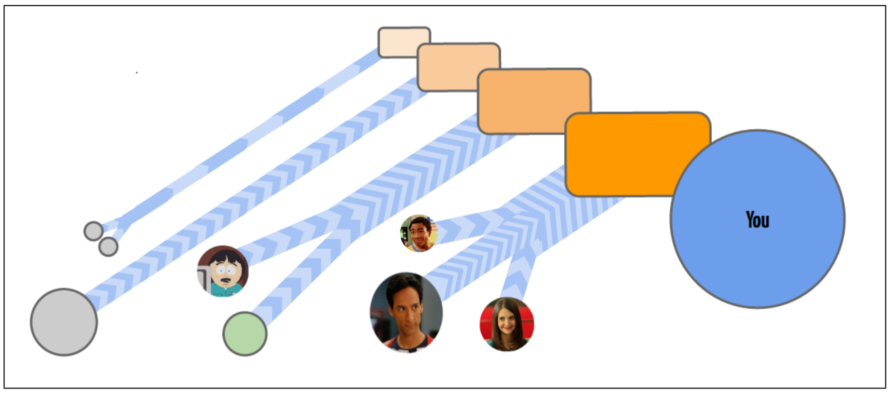


Figure 1.1: The strength of a post's Current will determine the order in which posts queue in your stream.

stream of posts. Suppose that Jeff is a member of a community of friends who are usually very attentive to his output. Jeff posts a new picture of himself, and offers a portion of his charge, sent out as current, to each friend. Annie is a close friend of Jeff and usually attends all of his posts. She receives enough current to make his post a priority, and sees it right away. Annie likes the picture and posts a flattering comment, an act that recharges Jeff's capacitor completely (no innuendo intended). Jeff's post has the same amount of charge left as when it started, meaning its potential to be viewed by the network remains the same, while Annie's engagement score with him increases, and so does Jeff's Reo in their shared community as a result. Next, Abed, who has been online the entire time, is slowly approaching Jeff's post's position in his stream queue. While Abed is relatively close to Jeff, Abed has many other interests, also being a member of other communities in which he has a high Reo score, and so his stream's queue is often populated with items with high current as users vie for his attention. Abed isn't as impressed as Annie with Jeff's picture, and he scrolls through it quickly. This wastes all of the current Jeff has sent his way, and decreases the amount of current Jeff may offer the friends who have not seen his post yet. When Pierce comes online, Jeff's post is the first to appear on his stream. Pierce stalks Jeff daily, engaging with every piece of content he publishes, and so even Jeff's diminished charge, and the relatively small portion of it he is willing to send to the low-Reo'd Pierce translates to a lot of current when sent to him. Unfortunately, Pierce is angry with Jeff for not responding to his instant messages earlier in the day, and so he spends some time looking at the picture as he mumbles obscenities to himself before scrolling past it furiously. As Pierce spent some time looking at the picture, Jeff's capacitor recharges a bit, but overall takes a hit from this interaction. Jeff's remaining friends either ignore or engage with his picture in various ways, until charge, which also slowly decays over time, runs out to a degree where it cannot reach anyone's attention on the network any longer.

... but wait! Pierce suddenly realizes that he had seen an incriminating detail in Jeff's picture. Jeff's watch shows that the picture was taken at a time where Jeff had claimed to be busy attending to his sick friend, an excuse he used to get out of the study group's most recent Diorama creating homework session! Pierce jumps at the opportunity to get back at Jeff for ignoring him. He amplifies Jeff's picture post, sharing it to all of Greendale, investing 2500 AMPs to make sure it reaches all of them accompanied by his snide commentary. Pierce's action refill's Jeff's post's capacitor completely, and because of the large investment, also increases its capacity beyond its previous maximum. As most of Pierce's plans turn to go, this one backfires as well, and Jeff's picture, now enjoying more attention than it would ever have received otherwise, spreads through the Greendale network and beyond. As each new fan engages with the picture, Jeff's capacitor recharges again, and the fan opens up new networks and communities for the post to flow to. Needless to say, Jeff's Reo increases network wide, and he also receives many new friend requests and phone numbers to boot.

The following week, Dean Pelton decides to advertise a new class added to the Greendale curriculum. Knowing that he isn't very popular on the network and that his Reo score is very low, he wisely chooses to amplify his post, investing AMPs so that the message reaches every student in Greendale and with high priority. Since Jeff's Reo increased dramatically after his picture had exploded in popularity network-wide, when he sees the Dean's amplified post, he receives a larger portion of the invested AMPs than everyone else who has seen it does.

Jeff likes the Dean's post, as the class added is The Ultimate Blowoff Class: beginner pottery. Jeff's rebroadcast of the Dean's post has a high impact on the Greendale network, making Dean Pelton's expensive investment in reaching Jeff's attention a success.

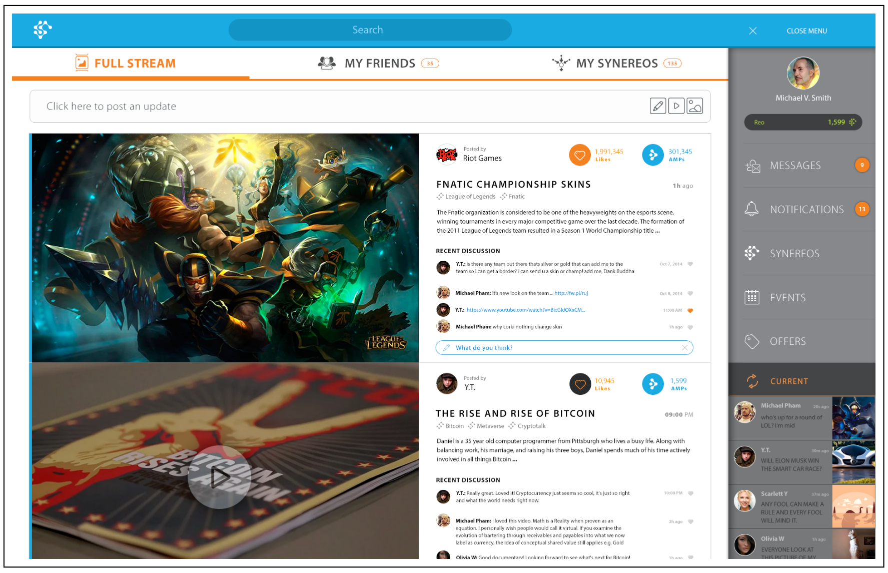
1.3.3 User interface

While the above scenario implicitly suggests a social network structure that is similar to what we're used to from Google Plus or Facebook, there are many different interfaces and ways to organize the prioritized queue and the information contained within it. Synereo aims to provide its users with multiple ways to experience the same stream of information. From a Reddit-type view in a current events community, to a tumblr-type view for an art community, to the single column stream view for your Home Synereo... which may look like this:

Users may configure each group they participate in to be displayed according to their preference or accept the default that may be set by its administrators. Users may also define streams that display items from specific sources, saving them for when they wish to browse through different parts of the Synereo network, its various communities, and different types of available interactions or items coming from different applications.

1.3.4 Other applications

Many different applications may be built on top of the Synereo framework, enjoying the benefits of its distributed technology and attention model. These applications may choose to use one of the interfaces provided, only changing the types of information and associated actions in each stream item, or innovate and offer completely different experiences. Applications that adhere to the basic information/action unit may offer users to send these items right to their standard Synereo stream, and if they succeed in engaging the user, find an enduring place in it. Applications that do this may send in a card with your next move ready to be played in a simple multiplayer game, incoming mails from senders or topics that the application has learned are important to you, or a mindfulness application sending you an inspiring message and prompting you to click it so that it sets a timer for your one-minute meditation. These items may appear as you browse through your “normal” stream, competing for your attention with posts from your friends and from other similar applications.



Posted by
Riot Games
1,991,345 Likes
301,345 AMPs

FNATIC CHAMPIONSHIP SKINS 1h ago
League of Legends Fnatic

The Fnatic organization is considered to be one of the heavyweights on the esports scene, winning tournaments in every major competitive game over the last decade. The formation of the 2011 League of Legends team resulted in a Season 1 World Championship title ...

RECENT DISCUSSION

Y.T.: is there any team out there thats silver or gold that can add me to the team so i can get a border? i can send u a skin or champ! add me, Dank Buddha — Oct 7, 2014

Michael Pham: it's new look on the team ... http://fw.pl/ruj — Oct 8, 2014

Y.T.: https://www.youtube.com/watch?v=BicGIdOXxCM... — 11:00 AM

Michael Pham: why corki nothing change skin — 1h ago

What do you think?

Posted by
Y.T.
10,945 Likes
1,599 AMPs

THE RISE AND RISE OF BITCOIN 09:00 PM
Bitcoin Metaverse Cryptotalk

Daniel is a 35 year old computer programmer from Pittsburgh who lives a busy life. Along with balancing work, his marriage, and raising his three boys, Daniel spends much of his time actively involved in all things Bitcoin ...

RECENT DISCUSSION

Y.T.: Really great. Loved it! Cryptocurrency just seems so cool, it's just so right and what the world needs right now. — 10:00 PM

Michael Pham: I loved this video. Math is a Reality when proven as an equation. I personally wish people would call it virtual. If you examine the evolution of bartering through receivables and payables into what we now label as currency, the idea of conceptual shared value still applies e.g. Gold — 2h ago

Olivia W: Good documentary! Looking forward to see what's next for Bitcoin! — 1h ago

Michael Pham 20s ago
who's up for a round of LOL? I'm mid

Y.T. 30m ago
WILL ELON MUSK WIN THE SMART CAR RACE?

Scarlett Y 37m ago
ANY FOOL CAN MAKE A RULE AND EVERY FOOL WILL MIND IT.

Olivia W 1h ago
EVERYONE LOOK AT THIS PICTURE OF MY

Figure 1.2: Synereo stream

Chapter 2

Synereo: network model and user interactions

2.1 Overview

2.1.1 Synereo high-level overview

Figure 1.2 shows an overview of the Synereo ecosystem. It is important to distinguish between Synereo in its manifestation as a fully-featured social network and the lower level Network Model and Attention Model. These models are designed to be fully extensible, and to be used to build other decentralized services on. The lower levels in the stack are thoroughly discussed later in this paper.

2.1.2 Outline of the chapter

In the remainder of the chapter, we introduce the network model from a global perspective and discuss the attention model and how it fits into it. Next, we map that onto a distributed model. Then, we map that model onto a distributed architecture and show how the model maps directly to working code in that

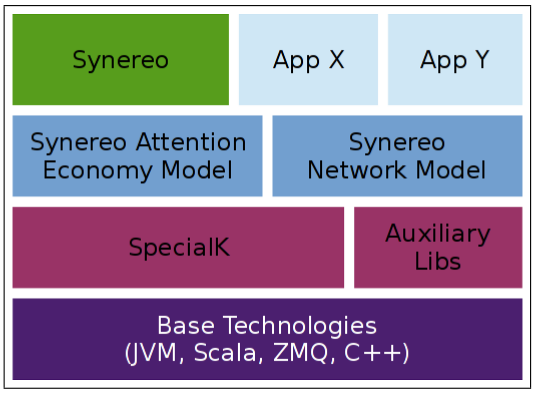

Synereo
App X
App Y
Synereo Attention Economy Model
Synereo Network Model
SpecialK
Auxiliary Libs
Base Technologies (JVM, Scala, ZMQ, C++)

Figure 2.1: Synereo stack overview

distributed architecture. Subsequently, we address both gateway and third party services. Then, we discuss deployment options. Finally, we make a few closing remarks about the nature of Synereo in the wider ecosystem of social networks.

Notation

Most of the notation adopted here is relatively standard for computer science. The presentation is primarily functional in the sense that the universe of discourse used is built from (structured) sets and functions. Additionally, we do avail ourselves of higher-order functions, making use of lambda terms $\lambda x.term$ to define anonymous functions.

Out of courtesy, whenever we are defining a function we first give its type. We assume a rich ambient type theory that includes basics like tupling (with fst and snd denoting the first and second projections of a pair, respectively) as well as fancier more modern machinery like parametric polymorphism and the usual suspects of collection types like List and Tree, and assume all the standard functions for collections (map, filter, etc) are well understood computationally and as mathematical objects. Occasionally we will use expressions like $\mathsf{node}(G)$ as a type; this should not be taken to mean that the model presented here fundamentally requires a dependent type theory. However, dependent type theory is becoming more and more mainstream, and so we don't feel that a more aggressive use of dependent types would actually present a serious obstacle. To the contrary, it would likely simplify the model in many places.

Essentially, all of these assumptions amount to making it quite straightforward to render the model as a simulation in a functional language. Likewise, a reader with some facility in functional programming should have no difficulty parsing the presentation of our model.

2.2 A dynamics for social agency

In this section, we describe the network and its dynamics. We show how engagement, quality of attention, and the flow of content are related to Reo, a measure of reputation in the network, and AMPs, Synereo's information flow currency. We'll present the model in stages of refinement. The first step regards the network as a whole. The next step shows how the network as a whole is mapped onto concurrently running distributed compute nodes relying solely on local interaction.

2.2.1 Network Model

The description presented here will provide a global or "god's eye" view of the network to illustrate the dynamics of the whole network. In subsequent sections we will show how to realize these global dynamics in terms of entirely local interactions in a distributed network of interacting processes.

We'll begin by modeling a Synereo network as a graph, G, with nodes, written $\mathsf{node}(G)$, representing users of Synereo (plus their system footprint) and edges, written $\mathsf{edge}(G)$, representing communication links through Synereo. When it's convenient we'll write $\mathsf{in}_G(N)$ to denote the set of edges in G with N as the target, and similarly $\mathsf{out}_G(N)$ to denote the set of edges in G with N as the source. More formally,

$$\mathsf{in}_G(N) = \{E \in \mathsf{edge}(G) \mid N = \mathsf{trgt}(E)\}$$
$$\mathsf{out}_G(N) = \{E \in \mathsf{edge}(G) \mid N = \mathsf{src}(E)\}$$

We calculate the neighbors of a node

$$\begin{aligned}\mathsf{neighbors}_G(N) = \{N' \in \mathsf{node}(G) \mid \exists E.N = \mathsf{src}(E) \ \&\& \ N' = \mathsf{trgt}(E) \\ \| \ N' = \mathsf{src}(E) \ \&\& \ N = \mathsf{trgt}(E)\}\end{aligned}$$

Events Synereo users take actions, such as posting content, liking posts, etc, that are represented in terms of events in the system. The type of such events is written $\mathsf{Event}[Content]$, parametric in the type of content. To each inhabitant of $\mathsf{Event}[Content]$ we associate a target $\mathsf{trgt} : \mathsf{Event}[Content] \to \mathsf{node}(G)$, a source $\mathsf{src} : \mathsf{Event}[Content] \to \mathsf{node}(G)$, as well as a way to retrieve the content of the event, $\mathsf{content} : \mathsf{Event}[Content] \to Content$. As we will see in the sequel we associate a notion of *charge* with the transmission of content through the network. Events play a role in keeping track of charge associated with content. As a result, we also associate a map $\mathsf{charge} : Event[Content] \Rightarrow \mathbb{R}^+$ to access the event's component of the charge associated with a particular transmission of content. Likewise, events will serve a purpose in the redistribution of AMPs, essentially serving as a carrier for the transmission of AMPs from one user to another. Hence, we also assume an accessor, $\mathsf{AMPs} : \mathsf{Event}[Content] \to \mathbb{R}$.

Closely associated with charge is the notion of automatic rebroadcasting. Some users will wish for some content to achieve as much reach in the network as possible. A basic element of this capacity is whether or not an event can be rebroadcast, given that there are not other information flow policies in play governing the content (see section 2.2.9). For our purposes it suffices to associate a boolean with events, $\mathsf{rebroadcast} : Event \Rightarrow Bool$, indicating whether it is permitted to rebroadcast the event to the recipient's connections. This may be linked to user control so that they can decide what content travels beyond its initial targets.

It will sometimes be convenient to assume a *constructor* for events, $\mathsf{event} : \mathsf{node}(G) \times \mathsf{node}(G) \times Content \times \mathbb{R} \times Bool \Rightarrow \mathsf{Event}$, recognizing that this is a minimal abstraction. A real implementation is likely to have a much more elaborate constructor.

A sequence of events, sometimes called a *history*, is treated as an element of the type $\mathsf{List}[\mathsf{Event}[Content]]$. It is convenient to be able to collect the events in a history, H, with the same content, c, $\mathsf{extent}(H, c) = \mathsf{filter}(H, \lambda e.\mathsf{content}(e) == c)$. Note that a single user action, such as posting content to a group of friends, may generate *many* events. So, other collection types, such as Tree may be used to provide histories where some events cannot be ordered, or interleaving makes less sense. When convenient and there is no risk of confusion, we elide the type parameter of the type of events, writing simply Event instead.

Reo

Definition 1 (rank) $\mathsf{rank} : \mathsf{Event} \to [0, 1]$ *is a primitive numeric measure of the attention* $\mathsf{trgt}(e)$*, has given the event,* e*, generated by* $\mathsf{src}(e)$*.*

For all intents and purposes, rank is a measurement of the amount and quality of attention given by a user on Synereo to a specific unit of information created by another user. Every piece of content appearing in your Synereo stream receives a rank score depending on your engagement with it. In a social network setting, engagement is measured by liking, commenting, sharing, and so on. However, other settings may have different engagement measurements. Advancements in technology, allowing more direct interfaces with correlates of attention, may further improve the fidelity of this measurement.

Setting

Example 1 *Assuming* $\mathsf{src}(\mathsf{contentPosting}) = \mathsf{Stilgar}$ *and* $\mathsf{trgt}(\mathsf{contentPosting}) = \mathsf{Usul}$

$$\mathsf{rank}(contentPosting) = 1$$

could be used to represent the fact that Stilgar *read, liked and reshared* Usul*'s posting, while setting*

Example 2

$$\mathsf{rank}(contentPosting) = 0$$

could be used to represent that Stilgar *ignored* Usul*'s posting.*

Definition 2 (engagement) $\mathsf{engagement} : \mathsf{node}(G) \times \mathsf{node}(G) \times \mathsf{List}[\mathsf{Event}] \to \mathbb{R}$ *is a measure of the attention Node B, the second parameter, has given to events in a history, say H, generated by Node A, the first parameter. Thus, without loss of generality, we may assume that* $\mathsf{foldl}(H, \lambda acc\ e.acc \cup \{\mathsf{src}(e)\}, \{\}) == \{A\}$ *and likewise* $\mathsf{foldl}(H, \lambda acc\ e.acc \cup \{\mathsf{trgt}(e)\}, \{\}) == \{B\}$*, for if not we simply filter the history, providing* $\mathsf{filter}(H, \lambda e.\mathsf{src}(e) == A\ \&\&\ \mathsf{trgt}(e) == B)$ *as the parameter instead.*

$$\mathsf{engagement}(A, B, H) = \begin{cases} \mathsf{engagement}(A, B, H') + \frac{\mathsf{rank}(e)}{\#(\mathsf{rank}(e), H')}, & if H' = e :: H \\ K(A, B), & otherwise \end{cases} \tag{2.1}$$

where $\#(\mathsf{rank}(e), H')$ *denotes the frequency of the rank(e) when applied to all of* H' *and* $K(A, B)$ *is some base level measure.*

Engagement is a measure of how attentionate a user is to another user's content and actions on Synereo. Engagement determines how difficult it is for users to broadcast information to their peers. The higher the engagement score, the easier it is to broadcast to the "engaged" user. Notice that engagement is not a simple accumulation of *rank*. Instead, events with the same *rank* contribute less and less as their frequency increases. This is intended to lower the impact of repeated responses such as "liking" every post, normalizing the effect this indication of provided attention has on the measure of engagement. i.e. if someone is statistically likely to engage a given post, the effect on engagement score is reduced as it is less meaningful as an indicator of provided attention.

The engagement score is more difficult to change the farther away it is from its neutral, starting point. The more a user is engaged with you, the less impact on engagement each further interaction with you will have.

Engagement score will decay to its neutral point over time.

Definition 3 (bundle) $\mathsf{bundle} : \mathsf{node}(G) \times \mathsf{node}(G) \to \wp(\mathsf{node}(G))$ *is the set of nodes in G which to which both parameters are connected.*

$$\begin{aligned}
&\mathsf{bundle}(A,B) = \\
&\qquad \{N \in \mathsf{node}(G) | \exists E1, E2 \in \mathsf{edge}(G). \\
&\qquad (\mathsf{src}(E1) = A, \mathsf{trgt}(E1) = N, \mathsf{src}(E2) = B, \mathsf{trgt}(E2) = N) \textit{ or} \\
&\qquad (\mathsf{src}(E1) = N, \mathsf{trgt}(E1) = A, \mathsf{src}(E2) = N, \mathsf{trgt}(E2) = B) \textit{ or} \ldots\}
\end{aligned} \tag{2.2}$$

The bundle between each two nodes on the network represents the area of the network that is shared between these nodes and that they are affected by.

Definition 4 (Reo) $\mathsf{Reo} : Node \times Node \to \mathbb{R}$ *is a measure of how the common community, written* $\mathsf{bundle}(A,B)$*, of the two node parameters, say A, and B values the contribution of each relative to the other.*

Let $S = (\bigcup_{N \in \mathsf{bundle}(A,B)} \mathsf{bundle}(A,N)) \setminus \mathsf{bundle}(A,B)$ *in*

$$Common(A,B,H) = \sum_{N \in \mathsf{bundle}(A,B)} \mathsf{engagement}(A,N,H) * \mathsf{engagement}(B,N,H) \tag{2.3}$$

and

$$\mathsf{Reo}(A,B,H) = \frac{Common(A,B,H)}{|\mathsf{bundle}(A,B)|} + \sum_{N \in S} \mathsf{Reo}(A,N,H) \tag{2.4}$$

Notice that, as currently formulated, Reo is asymmetric, i.e.

$$\mathsf{Reo}(A,B,H) \neq \mathsf{Reo}(B,A,H)$$

Using these definitions it's easy to see (figure 2.2) that for two Synereo users, Usul and Stilgar [1], no matter how well known Usul is to his community or Stilgar to his, they have equal standing if their respective communities do not overlap.

If their communities do overlap, however, then Usul's standing with respect to Stilgar will be higher than Stilgar's standing with respect to Usul just when their common community gives more attention to Usul's output (figure 2.3). This discussion should indicate precisely how the construction of the Reo measure defeats Sybil-like attacks. If either Stilgar or Usul create an army of bots, all ready to attest to the power of their creator's word, this attestation is highly unlikely to contribute to the standing of either in the eyes of the other because they will not share the bots in their common community.

AMPs

AMPs constitute a form of attention-based currency. They are primarily a conserved quantity. The network begins with a certain fixed amount. There is a distributed ledger (like blockchain) that maintains the distribution of AMPs in the network. Intuitively, AMPs may be spent to amplify the visibility and reach of content.

[1] and our sincerest apologies to Frank Herbert

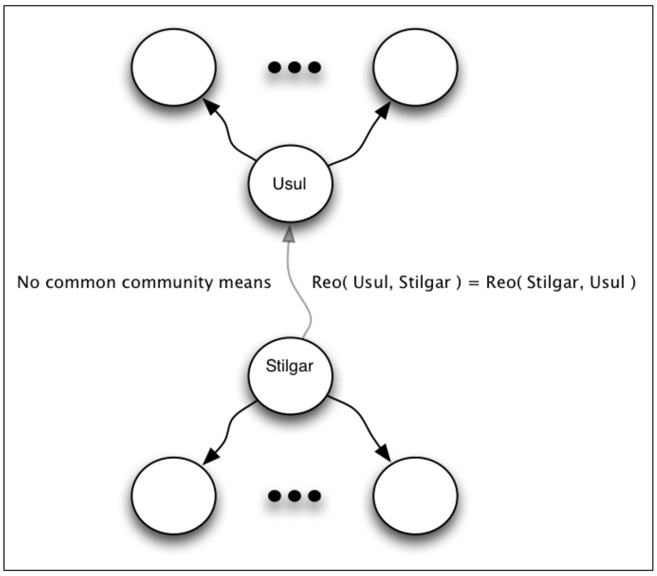

Usul
No common community means
Reo(Usul, Stilgar) = Reo(Stilgar, Usul)
Stilgar

Figure 2.2: Reo calculation - empty bundle

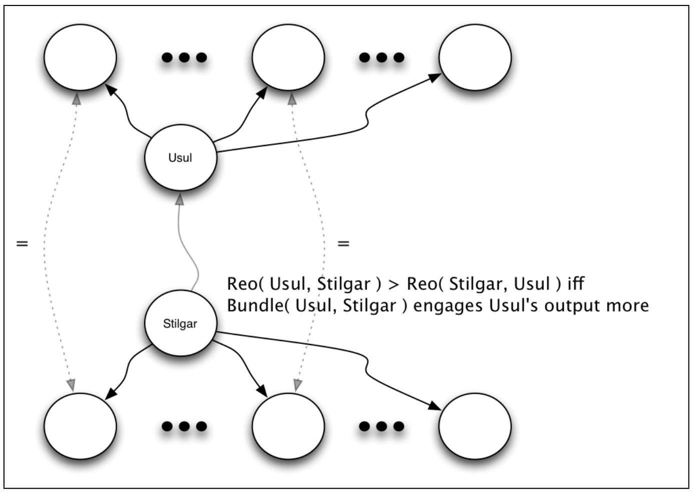

Usul
=
=
Reo(Usul, Stilgar) > Reo(Stilgar, Usul) iff
Bundle(Usul, Stilgar) engages Usul's output more
Stilgar

Figure 2.3: Reo calculation - full bundle

Definition 5 (AMP) *An* AMP *is a unit of currency available for a user, represented by a node, N to spend amplifying visibility and reach of content, through sharing and republishing.*

2.2.2 How Reo and AMPs get used in Synereo

Next, we look at the events Synereo users generate and how they are influenced by Reo and AMPs. With respect to content sharing, there are essentially two kinds of events, those associated with the production of content and those associated with the consumption of content.

We associate production of content with volition, whether it is production of original content or resharing content received. This is in contrast to content becoming more likely to show up in a user's stream because it has received some form of attention. Conversely, we associate consumption with the deployment of attention. We recognize that there is some overlap in these categories. For example, "liking" a post is an act of volition, but the aim of such an act is to acknowledge deployment of attention.

How does Reo figure into the publication of content in the network? Likewise how do AMPs influence the publication process? Let's consider a user, such as Abed, who wants to share a video and so publishes it to his community of friends. This means that the message with the link to the video is offered to each Synereo node associated with one of Abed's friends in Synereo. The node on the receiving end will initiate a process to determine the ordering of that message in Abed's friend's stream, and that process makes essential use of Reo and AMPs.

Network dynamics and information flow dynamics

In order to address this we have to introduce the dynamics of information flow in the network. The most basic and natural intuition is that events flow from node to node over the edges connecting them. As a first approximation we will treat the network as fixed and discuss how content flows in that fixed network. The reality, however, is that the network is dynamic, both at the level of nodes supporting the Synereo protocols and at the level of the social network, who is connected to whom, who is friends with whom. Moreover, the network topology, especially the topology of the social network, changes as a direct consequence of various content flows. Abed posts an interesting video. Troy shares it with a friend, acknowledging Abed's authorship. The friend is impressed and asks Troy for an introduction to Abed.

In today's world, network dynamics and their mutually recursive dependency on information flow dynamics are the bread and butter of social networks. Being able to model this phenomenon succinctly and precisely ultimately requires moving to a different approach, and is another key reason why we introduce a process calculus based model in subsequent sections. The concept of mobility in the π-calculus captures this notion perfectly in a beautiful algebraic theory, rather than in an adhoc manner. We recognize that while the theory of mobile processes is nearly 30 years old, it's still new to many. So, we get there in stages.

Part of the advantage of having a "god's eye" view is that we can not only see the whole network, in its entirety, but we can see the whole of time, in its entirety. Fortunately, through the magic of recursion, we don't have to see it all at once! We can approximate the behavior at a single node in terms of a recursive type.

User response To see this, let's assume that we have a basic type *Response* that represents the possible ways a user might respond to content coming their way. At this point in the modeling, we don't really need to know all that goes into this type. We know, for example, that it will be correlated to rank, and we could even take the target of range as a proxy for *Response*, but it's actually better if we can make our model of flow dynamics generic in this type. It allows us to change our minds later, if we refine our understanding of user response.

There is one aspect of user response that we can commit to, though: they generate new events in the system in response to events coming their way. Thus, a more full accounting of user behavior is a pair, their local response to the content, and the collection of events they generate in response to the content, i.e. the communications, reposting, comments, etc that happen as a result of being presented with an impression.

AMP distribution and other state As this is where the temporal dynamics are being modeled, it also makes sense that the AMP distribution figures into this type. That is, AMP distribution changes in response to events in the system. Likewise, it makes sense to model relevant features of a user's stream, and all other aspects of state associated to individual users here. For our purposes we assume a type State, with accessors $\mathsf{AMPs} : \mathsf{State} \Rightarrow \mathbb{R}^+$ and $\mathsf{stream} : \mathsf{State} \Rightarrow \mathsf{Stream}[\mathit{Content}]$. More generally, we assume that State supports a rich algebra. It's not unreasonable, for example, to assume that a projection of the type includes a vector space.

State predicates In the sequel it will also be convenient to have a language for predicates on states, alternatively a means of describing sets of states, recognizing that these are more or less interchangeable. We defer a specification of such a language until the last possible moment. For now, we simply adopt the notational convention that $\mathbb{S}$, $\mathbb{T}$, ... are used to range over such predicates or sets. Note that individual states can be trivially lifted to predicates.

Continuations There's a final aspect of user response that often goes unnoticed, *they continue their behavior as users.*[2] How a user behaves subsequently, as a function of the impressions they received, is part an parcel of a good model, one that takes into account an actual interaction between user and system.

Node agency

Equipped with these intuitions, we can model behavior at a node. A naive approach models behavior as a *function*, k, that takes an event, e, and determines a single 4-tuple recording the user's local response, the user's new supply of AMPs, the events the user generates, and the continuation of the user's behavior. This choice, however, implies that user behavior is completely deterministic.

[2]Or not! Sometimes, when presented with something a user chooses to cease being a user of the system, but even this behavior should be accounted for in our model.

A more realistic choice allows for some intrinsic non-determinism in user behavior. Thus, instead of returning a single 4-tuple, it returns a set of 4-tuples. Notationally speaking, it is traditional to denote continuations using a k, and since even the initial behavior, broadly speaking, is a kind of continuation, we denote the type of user behaviors with a K.

Definition 6 (node-agent)

$$\mathsf{K}_G[\mathit{Response}] : \mathsf{Event} \Rightarrow \mathsf{Set}[\mathsf{Option}[\mathit{Response}] \times \mathsf{State} \times \mathsf{List}[\mathsf{Event}] \times \mathsf{K}_G[\mathit{Response}]] \tag{2.5}$$

Given $k : \mathsf{K}_G$, and $e : \mathsf{Event}$, we have $k(e) = \{(r, a, [e_1, \ldots, e_N], k'), \ldots\}$ where r is the local response (enriched to $\mathsf{Option}[\mathit{Response}]$ to include no response at all); a is the user's updated state (including its updated supply of AMPs, etc); $[e_1, \ldots, e_N]$ is a set of generated events; and, k' is the continuation of the user's behavior. It will be useful to be able to pick out the components of an element of $k(e)$ symbolically, rather that just by index, which we denote rsp, state, ev, k. The entire set of 4-tuples can be seen as a menu of options, or range of possible behaviors of the user, in response to the incoming event.

Example 3 *For any state, say a, we can define the null behavior $\mathsf{0}(a)$ is the one which does nothing in response to any event, and keeps on being the null behavior.*

$$\mathsf{0}(a) = \lambda e.\{(\mathsf{None}, a, [], \mathsf{0}(a))\} \tag{2.6}$$

Example 4 *Similarly, we can define the echo behavior, $\mathsf{echo}(a)$, as the one which echoes back to its sender any event it may have sent (and keeps on being the echo behavior).*

$$\mathsf{echo}(a) = \lambda e.\{(\mathsf{None}, a, [\mathsf{echo}(e)], \mathsf{echo}(a))\} \tag{2.7}$$

where $\mathsf{echo}(e) = \mathsf{event}(\mathsf{trgt}(e), \mathsf{src}(e), \mathsf{content}(e), \mathsf{AMP}(e), \mathsf{rebroadcast}(e))$

Example 5 *Continuing in this vein, we can define the forwarding behavior, $\mathsf{forward}(a)$, as the one which forwards to all its node, N's connections any content it receives (and keeps on being the forwarding behavior).*

$$\mathsf{forward}(a) = \lambda e.\{(\mathsf{None}, a, w, \mathsf{forward}(a))\} \tag{2.8}$$

where $w = [\mathsf{event}(\mathsf{trgt}(e), f, \mathsf{content}(e), 0, \mathsf{true}) | f \leftarrow \mathsf{neighbors}(\mathsf{trgt}(e))]$

Example 6 *To illustrate the use of options we can define a non-deterministic forwarding behavior, $\mathsf{nforward}(a)$. Suppose that $\mathsf{npartition}$ provides a random partition of a set. That is if $(f_1, f_2) = \mathsf{npartition}(\mathsf{neighbors}(N))$, then $\mathsf{neighbors}(N) = f_1 \cup f_2$. To model a process that non-deterministically forwards content either to some random subset of $\mathsf{neighbors}(N)$ or its complement we could write:*

$$\begin{aligned} &\mathsf{nforward}(a) = \\ &\qquad \mathsf{let}\ (f_1, f_2) = \mathsf{partition}(\mathsf{neighbors}(N))\ \mathsf{in} \\ &\qquad \lambda e.\{(\mathsf{None}, a, w_1, \mathsf{nforward}(a)), (\mathsf{None}, a, w_2, \mathsf{nforward}(a))\} \end{aligned} \tag{2.9}$$

where w is defined as in the previous example and $w_1 = \mathsf{filter}(w, \lambda e.\mathsf{trgt}(e) \in f_1)$ and and $w_2 = \mathsf{filter}(w, \lambda e.\mathsf{trgt}(e) \in f_2)$.

Initiative The astute reader [3] will have noticed that inhabitants of K are reactive. They do not behave except in response to external events. In real networks users are not always so passive. The way we model this without stepping outside this framework is to treat events whose src and trgt are the same, i.e. events sent from a node to itself, as providing the basic means to model *initiative* in a node. That is, there are events of the form $\mathsf{initiate}$, with $\mathsf{src}(\mathsf{initiate}) = \mathsf{trgt}(\mathsf{initiate})$ such that for any $k : \mathsf{K}_G$, $k(\mathsf{initiate}) = \{(\mathsf{None}, a, [e_1, \dots, e_N], k'), \dots\}$. Given one such option from this menu, the events $[e_1, \dots, e_N]$ are sent to their respective targets without any observable external event causing the node to send them.

Accessors We can use a similar trick to provide *accessors* to state associated with node behavior over time, like AMPs. For our purposes, we will assume an event of the form get, such that for any k, $k(\mathsf{get}) = \{(\mathsf{None}, a, [], k)\}$ allowing us to access the node's supply of AMPs (or any other state, in an elaborated version of the model.) It is convenient, therefore to define a function $\mathsf{AMPs} : \mathsf{K}_G \Rightarrow \mathbb{R}^+$ that picks out the AMP component of the state, i.e. $\mathsf{AMPs}(k) = \mathsf{AMPs}(\mathsf{state}(\mathsf{nth}(k(\mathsf{get}), 0)))$ and similarly $\mathsf{stream}(k) = \mathsf{stream}(\mathsf{state}(\mathsf{nth}(k(\mathsf{get}), 0)))$. We assume polymorphic versions of these functions, that lift to the set of options. Thus, for example, $\mathsf{state}(k(e)) = \{\mathsf{state}(opt) \mid opt \in k(e)\}$.

Using this approach a state predicate, say $\mathbb{S}$, can be smoothly extended to a node behavior, say k, with $k \in \mathbb{S}$ just when $\mathsf{state}(\mathsf{nth}(k(\mathsf{get}), 0)) \in \mathbb{S}$.

A more experienced reader may notice similarities between this model and Hewitt's actor model.[32] There are essential differences, and while they are too numerous to list exhaustively, one that is key is that this model makes no assumptions regarding the *fairness* of communication. Instead we rely on a notion of equivalence and substitutability to impose that kind of order from "below" so to speak.

Agent equivalence This presentation of node behavior is amenable to some fairly standard techniques. One of these provides an effective notion of the equivalence of node behaviors, or when one node behavior can be substituted for another. The technique is called *bisimulation.* The intuition is that two behaviors are equivalent if anything the one can do, the other can do and vice versa.

Definition 7 *More formally, k_1 is* bisimilar *to k_2, written $k_1 \approx k_2$, if for each e and $(r_1, a_1, es_1, k_1') \in k_1(e)$ there exists $(r_2, a_2, es_2, k_2') \in k_2(e)$ such that $r_1 = r_2$, $a_1 = a_2$, $es_1 = es_2$, and $k_1' \approx k_2$, and vice versa, with k_1 and k_2 swapping roles.*

Properties extended in time The shape of bisimulation reveals something about the recursive nature of continuations: they allow us to define properties of node behaviors that extend through time. Here's a simple example. Recall the $\mathsf{rebroadcast}$ predicate on events. We can ensure that every continuation in every option respects the predicate throughout its behavior following a simple recursive scheme.

Definition 8 (respectful) *we say $k : \mathsf{K}_G$ is* respectful*, if given a history, H, content c, and an event, $e \in \mathsf{extent}(H, c)$, then*

[3] That's *you*, my friend.

- *if for each* $(r, a, es, k') \in k(e)$ *for all* $e' \in es$ *such that* $\mathsf{src}(e') = \mathsf{trgt}(e)$ *then* $\mathsf{content}(e') = \mathsf{content}(e) \iff \mathsf{rebroadcast}(e) == \mathsf{true}$*;*
- k' *is* respectful*.*

The idea is to thread the basic predicate, which in this case ensures that content is not rebroadcast unless the predicate controlling that was true, throughout the rest of the behavior by insisting that every continuation in every option also enjoys the same property, recursively.

Definition 9 (preservation) *More generally, given a state predicate* $\mathbb{S}$*, we say a node behavior* k *is* $\mathbb{S}$-preserving *(alternatively,* conserving*) just when* $k \in \mathbb{S}$ *and for any* e *and* $(r, a, es, k') \in k(e)$ *we have* k' *is* $\mathbb{S}$-preserving*.*

This scheme is used throughout this paper (see definition 12) because it is so fundamental to reasoning about reactive decentralized, distributed systems. Moreover, there is a close connection between bisimulation and predicates extended in time arising from the fact that they both derive from this underlying scheme. Furthermore, the scheme also provides some insight into the much more powerful tools for governing node behavior we describe in section 2.2.9 on *social contracts.*

From this point on we assume behaviors are restricted to repectful behaviors.

Stepping back we can see unequivocally that this model is only an approximation. Why? Because there is no place in this type equation to model the change in the structure of the graph, G, that may result in responses to events, as discussed earlier. In fact, to provide such dynamics would considerably complicate the model. The graph G would have to become intertwined with node behavior and state. When we get to the π-calculus model we will see that this is exactly what happens, and the mutual recursion between graph dynamics and information flow dynamics will be clearly and cleanly accounted for.

Still, this formulation does provide an approximation to the behavior of the global network. This is given simply as a function,

Definition 10 (network-agent)

$$\mathsf{GK}_G[Response] : \mathsf{node}(G) \Rightarrow \mathsf{K}_G[Response] \qquad (2.10)$$

A function of this type assigns user behavior to each node in the graph. To provide a naive simulation of a Synereo network, G, is to specify an inhabitant of the type GK_G, i.e. some function, gk, taking each node N in G to its behavior, $gk(N)$. With this basic model of flow dynamics in a given network we can enrich the model to include the effects of charge and current.

Before we move on, though, it is important to point out that user behavior is not the sole behavior on the network. There are protocols running behind the scenes, like the calculation of Reo, that support the user's experience. The model above is presented at a level of abstraction that easily encompasses both user behavior and lower level node behavior. This provides yet another justification for having the model be parametric in the type *Response*, but also means that just by distinguishing between user events and lower level node events, we can provide a uniform account of the whole system. Therefore, in the sequel we will refer to inhabitants of K_G as node behaviors rather than user behaviors.

2.2.3 Charge and current

This definition of this process makes use of the principles of electric flow. To this end we introduce the idea of current and relate current to Reo and AMPs.

Definition 11 (charge) $\mathsf{charge}[Content] : \mathsf{List}[\mathsf{Event}[\mathsf{Content}]] \times Content \Rightarrow \mathbb{R}^{+}$ *In Synereo a user imparts a* charge *to the transmission of content. The network of Synereo nodes is required to keep track of the aggregate,* $\mathsf{charge}(H, c) = \sum_{e \in \mathsf{extent}(H,c)} \mathsf{charge}(e)$.

Intuitively, with each reproduction step in the path of content throughout the network, charge, the charge associated with the content dissipates a little. Unless it is replenished, ultimately, the charge is completely spent and the content can travel no further on it's own. Attention, i.e. user engagement is the only means for charge to be replenished. This places restrictions on legal node behaviors.

Definition 12 (charge-conservation) *More formally, we say* $k : \mathsf{K}_G$ *is* charge converving, *if given a history,* H*, content* c*, and an event,* $e \in \mathsf{extent}(H, c)$*, then for every* $(r, a, es, k') \in k(e)$*, we have that*

- *if the response,* r*, to the event, indicates engagement, i.e.* $\mathsf{rank}(e) > 0$*, then* $\mathsf{charge}((H \setminus e) + es, c) \geq \mathsf{charge}(H, c)$*;*
- *if the response,* r*, to the event, indicates no engagement, i.e.* $\mathsf{rank}(e) = 0$*, then* $\mathsf{charge}((H \setminus e) + es, c) < \mathsf{charge}(H, c)$*; and*
- k' *is* charge conserving.

From now onward we assume we are working with charge conserving node behaviors. It is worth taking a moment to understand the import of this choice. Rather than providing an operational account of charge at the level of node behavior, we insist on a property of all the node behaviors in the network. We don't care how this property is manifest, just that it is. This gives a great deal of freedom to explore a variety of implementations. The flip side is that this imposes a burden of proof on node behaviors that participate in the network. How this burden gets discharged is of considerable interest and one of the crucial features of Synereo's notion of contract.

Definition 13 (current) $\mathsf{current} : \mathsf{GK}_G \times \mathsf{Event} \times \mathsf{List}[\mathsf{Event}] \to \mathbb{R}$ *defines the force of flow of a content publication event* e *over a connection between* $s = \mathsf{src}(e)$ *and* $t = \mathsf{trgt}(e)$*. Given* $gk : \mathsf{GK}_G$*,* $\mathsf{AMPs}(e) \leq \mathsf{AMPs}(gk(s))$ current *is calculated*
$\mathsf{current}(gk, e, H) = \mathsf{charge}(H, \mathsf{content}(e)) * \mathsf{AMPs}(e) * \mathsf{engagement}(s, t, H) * \mathsf{Reo}(s, t, H)$

We can normalize current, $\mathsf{nc}(gk, e, H) = 1 - \frac{1}{\mathsf{current}(gk,e,H)}$.

We relate current to the competition for attention. Roughly, current is inversely proportional distance from the top of a user's stream. [4]

[4]This idea is more general than a linearly organized stream. If post display information was organized in 2 or even 3D space, then current would be inversely proportional to those points most likely to correspond to the center of the user's attention.

Definition 14 (priority) $\mathsf{priority} : \mathsf{GK}_G \times \mathsf{Event} \times \mathbb{R}^+ \times \mathsf{List}[\mathsf{Event}] \to \mathbb{N}$ *represents the distance from the top of the stream a piece of content associated with publication event e over a connection between $s = \mathsf{src}(e)$ and $t = \mathsf{trgt}(e)$, and is calculated*

$\mathsf{priority}_L(gk, e, H) = \mathsf{floor}(L/\mathsf{log}(\mathsf{current}(gk, e, H)))$

where L is the length of the view of the event target's $\mathsf{stream}(gk(t))$.

High current behind a given content publication event means it's more likely to be seen by the user. Thus, Usul's post is more likely to get Stilgar's attention when Usul has a high standing in their common community, i.e. a high Reo. Likewise, if Stilgar engages Usul's posts more frequently, then Usul's recent post is more likely to appear at the top of Stilgar's stream and thus get his attention. Notice that if Stilgar's standing in the common community isn't so high and/or Usul engages his output less frequently, then Stilgar can still gain Usul's attention if he is willing to spend AMPs to promote the posting.

Slightly more formally, in a network of *respectful*, *charge conserving* node behaviors, we can associate priority with a probability that a user will see content associated with a post. Roughly speaking, this probability is proportional to priority. This begets a probability, also proportional to priority, that the post receives engagement from same said user. Node behaviors that conserve (in the sense of paragraph 2.2.2) the probability assignment will improve signal given attention by the collective. Similarly, such node behaviors will dampen signal ignored by the collective.

2.2.4 Intention and impact

Given two state predicates $\mathbb{S}$ and $\mathbb{T}$ we write $\langle\mathbb{T}|\mathbb{S}\rangle$ for the probability (amplitude) of $\mathbb{T}$ given $\mathbb{S}$. Suppose now that $\mathbb{S}$ is intended to describe conditions on the state of a node, say N, just prior to receiving a publication event, say e, and $\mathbb{T}$ conditions on the state of the node after responding to the event and we want to measure the likelihood of observing $\mathbb{T}$ given $\mathbb{S}$ after N has processed e. The node N is in state $\mathbb{S}$ in any $gk \in \mathsf{GK}_G$ with $gk(N) \in \mathbb{S}$, thus we what to measure the likelihood that $\mathsf{state}(gk(N)(e)) \in \mathbb{T}$, which we would compute by $\langle\mathbb{T}|\mathsf{state}(gk(N)(e))\rangle$, but we weight this by current. Because current is dependent on a history, H, we either need to include it as a parameter or "quantify it out" by summing over histories.

$$\langle\mathbb{T}|\frac{e}{gk}|\mathbb{S}\rangle = \sum_H \mathsf{nc}(gk, e, H) * \langle\mathbb{T}|\mathsf{state}(gk(e))\rangle \tag{2.11}$$

Likewise, this is dependent on being provided a gk. We can "quantify out" this parameter by summing over all $gk \in \mathbb{S}$.

$$\langle\mathbb{T}|e|\mathbb{S}\rangle = \sum_{gk\in\mathbb{S}} \sum_H \mathsf{nc}(gk, e, H) * \langle\mathbb{T}|\mathsf{state}(gk(e))\rangle \tag{2.12}$$

Note that additional assumptions are required to guarantee that these sums converge. Accepting these requirements, however, gives a quantitative language for talking about how the network responds to content. In particular, if $\mathbb{S}$ represents a characterization of the current state of affairs, and $\mathbb{T}$ represents a characterization of intent, in the sense that it describes the set of acceptable outcomes, then $\langle\mathbb{T}|e|\mathbb{S}\rangle$ represents the impact of an event e, computing just how likely the event is to bring about intended outcomes.

Rebroadcasting

In many cases it's not necessary for a user to explicitly reshare the content. Instead we can define $\mathsf{rebroadcast} : \mathsf{K}_G \Rightarrow \mathsf{K}_G$.

Definition 15 (rebroadcast)

$$
\begin{aligned}
&\mathsf{rebroadcast}(k)(e) = \\
&\qquad \mathsf{for}((r, a, es, k') \leftarrow k(e))\{ \\
&\qquad \begin{cases} (r, a, es + w, \mathsf{rebroadcast}(k')) & if\ \mathsf{rank}(e) > 0, \mathsf{rebroadcast}(e) \\ (r, a, es, k'), & otherwise \end{cases} \\
&\qquad \}
\end{aligned}
\tag{2.13}
$$

where $w = [\mathsf{event}(\mathsf{trgt}(e), f, c, 0, \mathsf{true}) | f \leftarrow \mathsf{neighbors}(\mathsf{trgt}(e))]$

By actively engaging with content, a person gives his "stamp of approval" to it. This does not necessarily mean that he *endorses* the content, only that he found it noteworthy or worth discussing (or engaging with in other ways), and so the network will try to bring it to the attention of others who may feel the same way.

2.2.5 Network dynamics theorems

In this section we formalize the intuitions regarding collective attention and signal improvement. What does it mean to improve signal in a network? One simple answer is that a node receives content that the network determines it is likely to find engaging; moreover, there is a very good fit between content the network determines a node will find engaging and content a actually finds engaging.

It is important to note that engagement is distinct from like and dislike. A user may find some content apalling, and as a result feel compelled to engage. This is an oft told experience of the social activist, for example. Likewise, a user may find some content less compelling precisely because it fits their world-view and provides no challenge or surprise.

Thus, we want to demonstrate that Synereo network provides its users more and more engaging content rather than coccooning them in a mesh of comfortable and soothing impressions.

Likewise, we should be able to show that when individuals and subnetworks become more effective at providing engaging content, then they are rewarded: their standing in the community improves, and their ability to amplify signal also increases.

Socially meaningful proof-of-work and "mining" AMPs

If all value associated with cryptocurrency must ultimately trace back to value originating in fiat currency the transition to cryptocurrency will be a slow process, indeed. If, however, we provide a mechanism whereby a key aspect of the creative process is reflected in the AMP's relationship to value, that process can be dramatically accelerated. What everyone implicitly understands, yet very few explicitly acknowledge is that creativity has a distinctive mark: creative processes are *ex nihilo*; they generate something from nothing! Whether it's a new algorithm, a new song, or a new way of looking at the world, we recognize creativity in the freshness and newness of the offering – something is there that wasn't there before. Human life vitally depends on the font of creativity; without the renewal of creativity value slow receeds from our lives.

However, without a mechanism whereby that creativity results in the creation of currency, none of the value created by a genuinely creative offering is recognized. All of the existing currency traces value back to some other source. So, without such a mechanism the creative act is not recognized properly by the system. Note though, that if they find their audience, the consistently creative participant in the network will have high Reo. Their standing in the community will reflect the recognition of their creativity. If they had a means by which they could turn some of their accumulated Reo into AMPs, literally capitalizing on their reputation, then the system would actually have a means to recognize the creation of value in the creative act.

2.2.6 Simulating the network dynamics

To simulate the dynamics of the network we can model a user population in terms of distributions. Content preferences can be modeded in terms of a distribution over histories of events, while initial conditions are distributions over AMP allocation and initial engagement measures. Given this data we can watch how content flows in a simulated Synereo network. More technically, it is possible to model the dynamics of a network organized in this manner using Markov processes. Of the Markov process analytic machinery, the best suited for these purposes are the more recently developed labelled Markov processes [33]. We hope to publish our simulation efforts in a subsequent paper.

2.2.7 A decentralized version of the network model

While the model presented above is good for getting a view of the network dynamics, as a whole, it assumes a "god's eye" view of the social network. This is impractical for a distributed implementation. The question is whether there is a formalism sufficient to the task of reasoning about the global dynamics while restricting any formal specification to the local behavior of nodes in the network and their interactions with other nodes in the network. Without such an intermediate model it would be hard to be confident that any distributed implementation actually manifested the intended global network dynamics. In the next section we describe just such a model.

New tools for distributed applications

The mobile process calculi constitute a family of computational formalisms [34] designed to be for concurrent and distributed programming what the lambda calculus is for functional programming [35]. Among the mobile process calculi, Robin Milner's π-calculus is paradigmatic, and the variant known as applied π-calculus [36] has the richest toolset currently. (In the interest of keeping this paper somewhat self-contained we have provided a very brief introduction to the π-calculus in appendix 2.7.) The applied π-calculus has been used to describe, reason about, and prove correct a variety of protocols, such as electronic voting [37].[5]

In the next section we will develop a distributed model of the Synereo network dynamics using the π-calculus. There are three key ideas at play. The first is that the graph of the previous model will be represented as the parallel composition of processes representing the nodes of the graph, while the edges of the graph will be represented by channels these processes use to communicate. The second is that all the measures, like Reo and AMPs that are used to influence network dynamics will be internalized and used in protocols between the processes representing the nodes. We will see that the global dynamics arises as the dynamics of ensembles of these processes as they interact.

One final key idea is to remember that π-calculus expressions live a dual life. One the one hand π-calculus expressions are actually programs. They represent (source code for) processes that evolve by interacting and communicating with each other. Moreover, the parallel composition of a collection of π-calculus processes is itself a process, and reasoning about the behavior (aka dynamics) of that process is reasoning about the behavior of the whole network. In short, the π-calculus enables us to reason about the behavior of the whole by reasoning about the behavior of the parts. We only have to compare the behavior of the encompassing network process to the global dynamics to ensure we have met the global specification.

On the other hand, they are formal expressions in a formal calculus. As such they can be treated as *syntactic* entities; thus, the syntactic entities that represent the graph of the social network can be treated merely as an idiosyncratic data representation for the graph. As long as the graph structure can be recovered, then these syntactic representations can be the input of the graph required in the definitions in the previous section (2.2.1). This secures a connection between those definitions, the global dynamics, and the π-calculus representation of the network.

Spelling it out a bit more, verifying compliance proceeds by a structural induction on the shape of the process expression, which just happens to encode the shape of the graph.

The π-calculus in one easy lesson

Far from asking that we reduce the richness of experience to abstractions like 1's and 0's, the π-calculus suggests that computation, much like physics or biology, is built from interaction and exchange. At the heart of this model is a decomposition of interaction in terms of two kinds of processes, one waiting for input, and the other providing output, together with a means for them to come together and make the exchange. In symbols, the process $x?(y)P$ is waiting on the channel x for input, which when supplied will be bound to y in the body of P; meanwhile, $x!(z)$ is the symbolic expression of a process that supplies the data z on the channel x.

In this model, processes are considered autonomous, just like molecules in a solution, or devices on the Internet. Symbolically, we put processes P, Q, and R together in a mixture with expressions like $P|Q|R$. Because the $|$ represents a freely mixing solution, or a computational situation like the Internet where

[5]This turns out to be fortuitous, as the Reo calculation protocol is effectively a form of electronic voting.

devices may be mobile, the processes P, Q, and R are free to move around in $P|Q|R$ without changing the meaning of the expression. That is, $P|Q|R = Q|P|R = Q|R|P$, etc. In this sense, the | operator represents a very primitive form of distribution.[6]

Now, in a mixture of the form $x?(y)P|R|x!(z)$ the process, $x?(y)P$ which is waiting for input on the channel x can come together with the process $x!(z)$ which is ready to output z on the channel x and when they come together the data can be passed into P. Again, symbolically, $x?(y)P|R|x!(z) = x?(y)P|x!(z)|R$. When the data is exchanged the subexpression $x?(y)P|x!(z)$ rewrites to $P\{z/y\}$ indicating that everywhere y was mentioned in P it is replaced with the data z. Again, in symbols, $x?(y)P|x!(z)|R \to P\{z/y\}|R$.

For a more detailed account the reader can consult the appendix 2.7 or any of the wealth of papers on the model. Perhaps the most important take away is that it is consistent with the computational model the calculus entails to interpret π-calculus channels, like x in the expressions above, as persisted queues. Thus, $x?(y)P$ is interpreted as a program waiting for input from a queue named x, and $x!(z)$ is interpreted as a program that will output a message z on a queue named x. Then the π-calculus rewrite rule discussed above is simply programmatic communication over the queue. (See figure 2.12 for a complete translation to `Scala`.)

As mentioned above, the π-calculus provides a simple and clean model to express and reason about network dynamics as a result of information flow dynamics. In the π-calculus communication topology changes as a result of the information flow dynamics quite simply because channels are part of the content of messages. Thus,

$$\begin{aligned} &www?(useremail)useremail!(private)private?(request) \\ &\quad |\ www!(myemail)myemail?(private)private!(sell) \end{aligned} \tag{2.14}$$

is a simple process made up of a server (the top process), running concurrently with a client (the bottom process). The server waits at www for a client to approach, provide an email, and request a session with the session access to be sent in an email. The client approaches the server at www, providing an email where to send the session access, and then makes a *sell* request in the private session. The first exchange is at the well known channel www, but all the rest are on channels learned as part of interaction. The communication topology is changing with each exchange!

Social graphs as distributed processes

As mentioned above, the first difference between the network model as depicted above and a π-calculus based model is how the graph, G, is represented. In the latter, the graph is interpreted as a bunch of processes running concurrently, one for each node in $\mathsf{node}(G)$, and the edges in $\mathsf{edge}(G)$ are interpreted as communication channels, x, used between processes representing adjacent nodes to communicate. Thus, each process interpreting a node is actually parametric in the channels used to represent its incoming and outgoing edges.

[6]It doesn't capture a notion of subdomain or of fail-together regions, or other semantics commonly associated with distribution; but, those are more sophisticated notions that can be built in terms of these primitives.

More formally, if $A \in \mathsf{node}(G)$, we'll write $\pi(A)$ for the π-calculus process representing that node. Likewise, if $E \in \mathsf{edge}(G)$ with $src(E) = A$ and $trgt(E) = B$, we'll write $\pi(E)$ to represent the channel used by $\pi(A)$ and $\pi(B)$ to communicate. The π-calculus representation of the entire graph, which we'll write $\pi(G)$, can be expressed as a parallel composition of processes, that is

$$\pi(G) = \Pi_{N \in \mathsf{node}(G)} \pi(N) \tag{2.15}$$

Thus, if $\mathsf{node}(G) = \{N_1, \dots, N_m\}$, then

$$\pi(G) = \pi(N_1) \mid \dots \mid \pi(N_m)$$

Note that $\pi(N)$ is actually parametric in the interpretations of the incoming edges, $\pi(\mathsf{in}_G(N))$ and the the outgoing edges $\pi(\mathsf{out}_G(N))$; that is, the expression $\pi(N)$ is more accurately written $\pi(N)(\pi(\mathsf{in}_G(N)), \pi(\mathsf{out}_G(N)))$.

One of the most important features of this encoding is that it's compositional. That means that if the graph changes, for example if it grows to include two Synereo networks, the encode handles this without needing to reinterpret the new network from the ground up.

$$\pi(G + H) = \pi(G) \mid \pi(H) \tag{2.16}$$

In terms of software deployment, this means just adding in the processes that interpret the nodes of the second Synereo network; that is, the maths match the practice exactly.

The next difference is that the functions which are defined globally on the graph can be defined in terms of protocols between the processes. These constitute the behaviors of a node in the Synereo network. Each $\pi(N)$ is really just a π-calculus representation of the behaviors of a Synereo node.

We can, in fact, give a rigorous translation of the model of user behaviors presented in the previous section into a π-calculus model. Given k, representing a user behavior the translation into a π-calculus process is given by the equation

Definition 16 (π-node-agent) $[\![-]\!] : \mathsf{K}_G \times Channel \Rightarrow Proc$

$$[\![k]\!](d) = \sum_{(r,a,es,k') \in k(e)} trgt(e)?(e)\{d!(r) \mid \Pi_{e' \in es} trgt(e')!(e') \mid [\![k']\!](d)\} \tag{2.17}$$

Here the channel d, the second parameter to the translation, represents the place to send or record the response determined by k. This same technique can be used for making the state available to the node process. Likewise, to translate the behavior of the global network we simply put the translations of each node behavior in parallel composition.

Definition 17 (π-network-agent)

$$[\![\mathsf{GK}_G]\!] = \Pi_{N \in \mathsf{node}(G)} [\![\mathsf{GK}_G(N)]\!]$$

Theorem 1 (full abstraction) *Let k_1 and k_2 be node behaviors $k_1 \approx k_2 \iff [\![k_1]\!] \approx [\![k_2]\!]$*

Proof sketch: The proof that this is a full and faithful translation is remarkably straightforward. The intuition behind the translation and the proof of its correctness is that each event in the system is modeled as a communication step in the π-calculus process modeling the (aggregate) node behavior(s).

As noted above, the users's behavior, however, is actually only a part of the behavior of a node operating on the user's behalf. There are a number of protocols working to support the user's experience within Synereo. Let's look at a more detailed example, illustrating how Reo is calculated as a protocol between two nodes.

Figure 2.4 shows a pingpong diagram of how such a protocol might work.

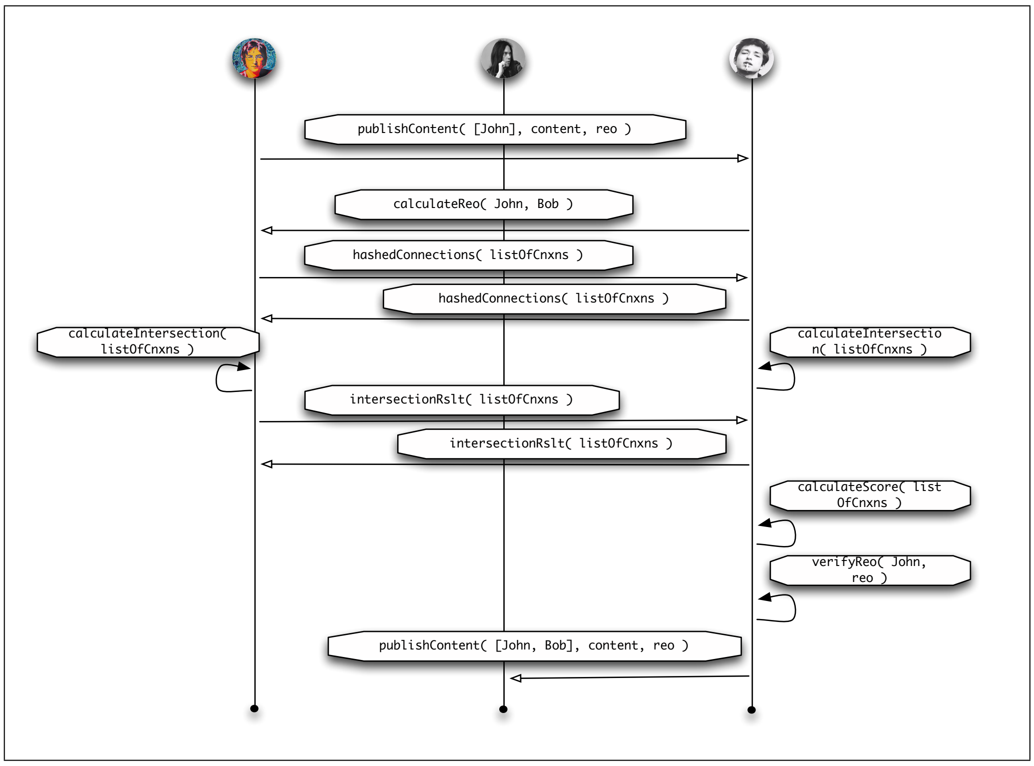

publishContent([John], content, reo)
calculateReo(John, Bob)
hashedConnections(listOfCnxns)
hashedConnections(listOfCnxns)
calculateIntersection(listOfCnxns)
calculateIntersectio n(listOfCnxns)
intersectionRslt(listOfCnxns)
intersectionRslt(listOfCnxns)
calculateScore(list OfCnxns)
verifyReo(John, reo)
publishContent([John, Bob], content, reo)

Figure 2.4: An example reo calculation protocol

Each vertical timeline, say T_i for $1 \leq i \leq 3$, in the protocol corresponds to a π-calculus process, $\pi(T_i)$. Here's the process encoding for the rightmost timeline, T_3.

$$\begin{aligned}
&\pi(T_3) \\
&\quad = \\
&\quad x_{trgt}!(\mathsf{calculateReo}(x_{trgt}, x_{src})) \mid x_{trgt}!(\mathsf{hashedConnections}(\mathsf{srcCnxns})) \\
&\quad \mid \mathsf{WaitForHashedCnxns}
\end{aligned} \tag{2.18}$$

$$\begin{array}{l} \mathsf{WaitForHashedCnxns} \\ \quad = \\ \quad x_{src}?(\mathsf{hashedConnections}(\mathsf{trgtCnxns}))\{ \\ \qquad | \; x_{trgt}!(\mathsf{intersection}(\mathsf{srcCommonCnxns})) \\ \qquad | \; x_{src}?(\mathsf{intersection}(\mathsf{trgtCommonCnxns}))\{ \\ \qquad \mathsf{Engagement} \\ \qquad \} \\ \quad \} \end{array} \tag{2.19}$$

Note that in the diagram the calculation of engagement is effectively elided. However, it is easily expanded as firing a bunch of queries to neighboring nodes in parallel.

$$\begin{array}{l} \mathsf{engagement} \\ \quad = \\ \quad \mathsf{if}(\mathsf{srcCommonCnxns} == \mathsf{trgtCommonCnxns})\{ \\ \qquad \Pi_{x \in \mathsf{bundle}(src,trgt)} x_{trgt}!(\mathsf{engagementReq}(\mathsf{trgt},\mathsf{x})) \\ \qquad | \; \Pi_{x \in \mathsf{bundle}(src,trgt)} x_{src}?(\mathsf{engagementRsp}(\mathsf{trgt},\mathsf{x})) \\ \quad \} \end{array} \tag{2.20}$$

Now, if D is the whole diagram, then π-calculus process interpreting D, or $\pi(D)$ is simply the parallel composition of the process interpreting each timeline. That is,

$$\pi(D) = \pi(T_1) \mid \pi(T_2) \mid \pi(T_3)$$

In section 2.3.1 we'll see how to render the term in equations 2.18, 2.19, and 2.20 as a Scala program using the SpecialK DSL.

With this formulation we can close a gap in the more abstract, global presentation. In the network dynamics model we didn't actually say how the nodes in the graph actually made use of the Reo formula. We simply showed how the Reo formula utilized the graph structure to reflect the attention and engagement of the community. We didn't show how that calculation then folded into the behavior of the nodes in the graph. With the model above we can not only see how the Reo formula can be calculated through only local communications, but also see how it can be folded in to influence the subsequent behavior of Synereo nodes.

Additionally, it is clear that repeatedly querying neighboring nodes for engagement scores will be inefficient. There are obvious optimizations involving caching such scores and invalidating the caches when connections get updated. We can use this model to reason about such optimizations and prove their behaviors equivalent to the original specification. If the π-calculus did nothing more than this, it would have paid its own freight. The fact is, however, that it is the technology enabling a step by step refinement from an idealized vision of how the Synereo network might work to functioning, correct code.

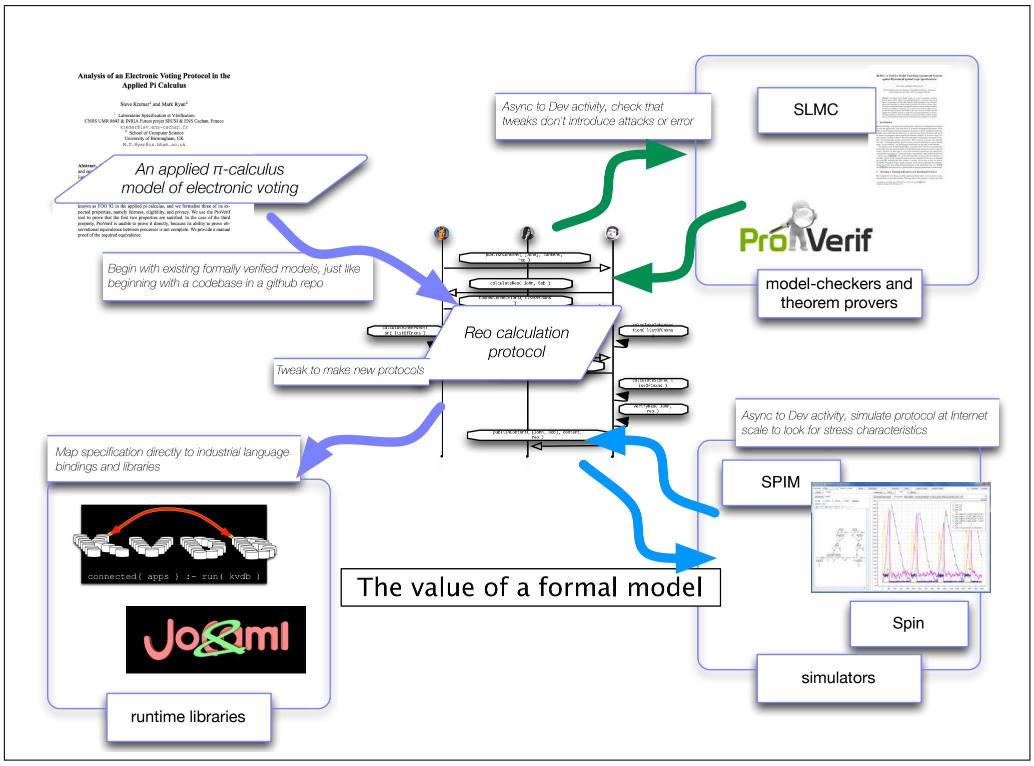

An applied π-calculus model of electronic voting
Begin with existing formally verified models, just like beginning with a codebase in a github repo
Async to Dev activity, check that tweaks don't introduce attacks or error
SLMC
ProVerif
model-checkers and theorem provers
Reo calculation protocol
Tweak to make new protocols
Map specification directly to industrial language bindings and libraries
connected(apps) :- run(kvdb)
JoCaml
runtime libraries
The value of a formal model
Async to Dev activity, simulate protocol at Internet scale to look for stress characteristics
SPIM
Spin
simulators

Figure 2.5: New tools for distributed applications

2.2.8 The *DendroNet*

Notice that using this approach it is not necessary to use blockchain technology for every aspect of distributed content sharing. In point of fact, that's rarely desirable. Blockchain is suited to distributed consensus. However, there need be no consensus amongst Alice, Bob, and Carol for what Alice shares privately with Carol. In the Synereo model, Reo, for example, is not a conserved quantity. It's calculated, not spent, and changes with the community's engagement with the behavior of its members. So, there is no need to use blockchain to manage and maintain Reo. On the other hand, it is a property of a distributed network and relies on communication amongst the nodes to be calculated as well as to influence how content propagates throughout the network.

By contrast, AMPs are a conserved quantity. The distribution of AMPs around the network is a matter of (distributed) consensus. Blockchain is a perfect tool for maintaining the distributed AMP ledger. Synereo, therefore uses a broader technology base, which we dub here the *DendroNet*, including a distributed network for content sharing and a separate network for maintaining the AMP ledger.

The *DendroNet* is the core component of Synereo which encompasses every node's awareness of its vicinity in the network, and where connected node metadata is stored. Through the *DendroNet* a node is able to make decisions about and prioritize content coming in from neighbouring nodes. The *DendroNet* is also used to store operational metadata of the network such as node attributes, preferences and encryption keys.

Content model and security

The *DendroNet*, as opposed to previously discussed concepts, is implemented on top of refinement of SpecialK/KVDB. SpecialK/KVDB enables pieces of content to live on the network regardless of whether the originating node is online or not. Public content is posted using keys directly to the KVDB; recipient processes querying at keys where content is not yet present are automatically suspended until such time that the content is available at that key. When content does land at the key, suspended recipient processes are awakened and provided the content. In some sense, SpecialK/KVDB can be regarded as just a fancy distributed hashtable (DHT) where the keys have internal structure allowing for efficient search over an infinite key space. Despite an infinite keyspace being large enough to enable strong security by obscurity, content saved to the KVDB should be regarded as public-knowledge. Permission models that reflect the Synereo-specific user model reside as a layer of permissions above this DHT, articulating how the content model addresses notion like *friends only*, *friends of friends*, etc.

When encrypting content on the *DendroNet* (as with any other similar encryption scheme) there is a tradeoff between strong per-recipient encryption, which naturally requires $O(n)$ storage and bandwidth, and sharing keys between recipients which is highly problematic in terms of authenticity and reliability. The Synereo content model relies on several permission classes, that in turn rely on a hybrid implementation of direct communication between nodes and using strong encryption on the *DendroNet* as a semi-persistent storage for offline nodes. We examine all the possible scenarios. Public content can be naturally be saved on the *DendroNet* without any encryption. Private content (i.e. direct messages between two nodes) will initially be delivered directly between the nodes, unless the recipient is offline. In this case, the content will be encrypted with the recipient node's public key, and saved on the *DendroNet* at a location where the node can later fetch it (give example). Thus the *DendroNet* is used as ephemeral storage until the node comes back online.

Content that has limited distribution, such as *friends only*, will be distributed using per-content keys. A single encrypted version is saved on the *DendroNet*, while peers are notified of the new content, and given the proper single-use decryption key. Given a content length calculation, short content can be directly delivered without the of using the *DendroNet*. In any case, short content or decryption keys are delivered via the process model, which provides proper assurances regarding offline delivery. As with any permission model, regardless of being distributed or centralised, nodes acting in bad faith (for example, sharing beyond intended audience) must be taken into account, and discussion as such is more of a social problem rather than a technical one.

Conserved quantities, double spending, and the DendroNet

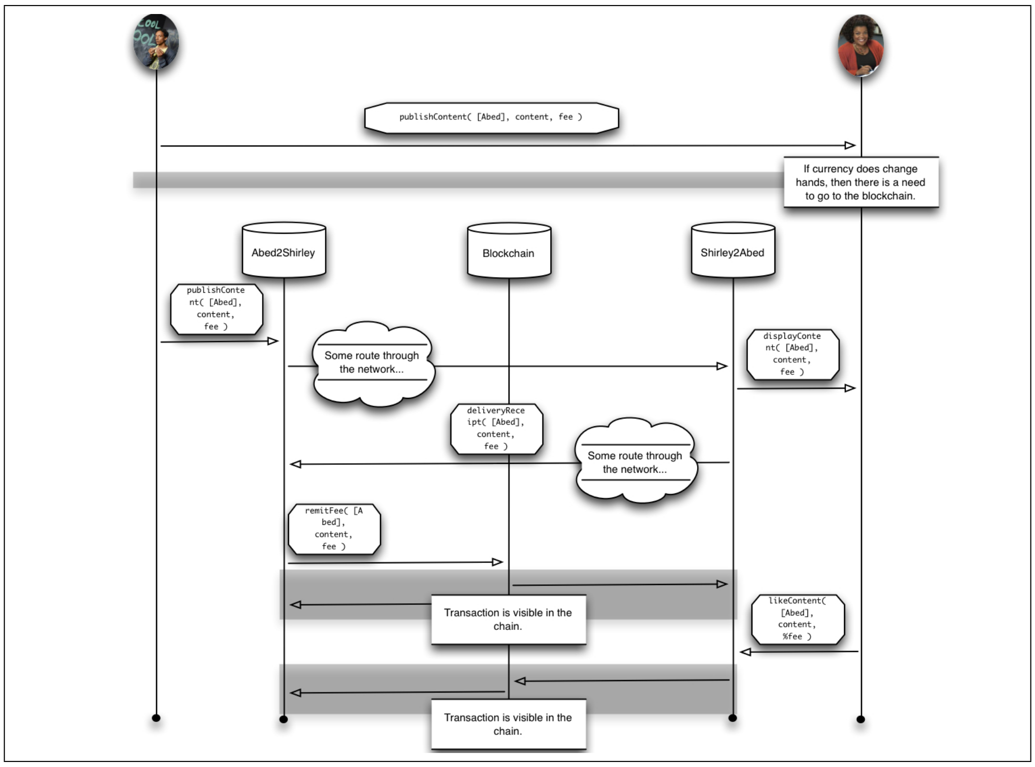

publishContent([Abed], content, fee)
If currency does change hands, then there is a need to go to the blockchain.
Abed2Shirley
Blockchain
Shirley2Abed
publishContent([Abed], content, fee)
Some route through the network...
displayContent([Abed], content, fee)
deliveryReceipt([Abed], content, fee)
Some route through the network...
remitFee([Abed], content, fee)
Transaction is visible in the chain.
likeContent([Abed], content, %fee)
Transaction is visible in the chain.

Figure 2.6: Interaction between content network and blockchain

Given that AMPs are a currency, i.e. a conserved quantity, with all that entails, they will naturally reside on a standard blockchain that will run as another tier of the Synereo network - or alternatively on an existing blockchain network, depending on implementation. Wallet data per each node, including public address and private keys, will be stored locally on the node, and not on the distributed KVDB. This is in line with practically all wallet clients that do not attempt to provide any secure means of online backup. It is the end user's responsibility to backup their wallet data, in case of node failure or theft.

Content exchanges where AMPs flow create transactional boundaries and atomicity constraints. Synereo is chooses optimistic algorithsm and protocols for content delivery where possible, but errs on the side of safety when it comes to all transactions involving the flow of currency. For example, in the flow above, content is delivered right away, but AMPs are not remitted until a read receipt is delivered to the poster.

2.2.9 New tools for security, privacy, and information flow policy

As noted above, blockchain technology only accounts for the maintenance and security of the AMP ledger, but what about the distributed content sharing mechanism? Being able to express who can see what and when is becoming a considerable concern in today's climate. [3] A quick look at the flow chart for who can see a user's post on Google+ [38] makes it clear both that people are concerned about this for a variety of reasons, from self promotion, or commercial advertising to a desire for privacy and and anonymity. It also makes it clear that it is far from easy to write down and reason about information flow policy in a distributed communication network.

One of the distinct advantages of the mobile process calculi, in general, and the π-calculus in particular, is that they come equipped with a new kind of logic, sometimes called Hennessy-Milner logic [39] [40] [41], [42] , that enables specification and reasoning about information flow policy. In the same way that the mobile process calculi provide new tools in the toolbox for reasoning about security and privacy [43], these new logics provide new approaches to addressing security and privacy in a distributed social network because they provide the means to write down and reason about information flow policy.

In particular, the combination of process calculus and logic provide a much better basis for a notion of smart contract. [44] To understand a little better how these logics provide a fresh approach to privacy and security, information flow policy and smart contracts in the distributed setting, let's consider a practical example.

Controlling information flow in mission critical situations

Consider a situation in which a doctor recently back from an aid mission to Sierra Leone leaves a hospital in a densely populated city in the US after having been erroneously cleared as ebola-free. The hospital discovers the mix up, and the doctor is at large, potentially spreading a deadly disease around the city.

In such a situation the CDC may wish to ensure that healthcare providers and city officials are notified before any member of the press is notified. How would such a policy be expressed and enforced in a distributed communication network like Synereo?

Here is a specification of (the relevant communications of) such a system, together with a specification of the properties we want the system to hold, such

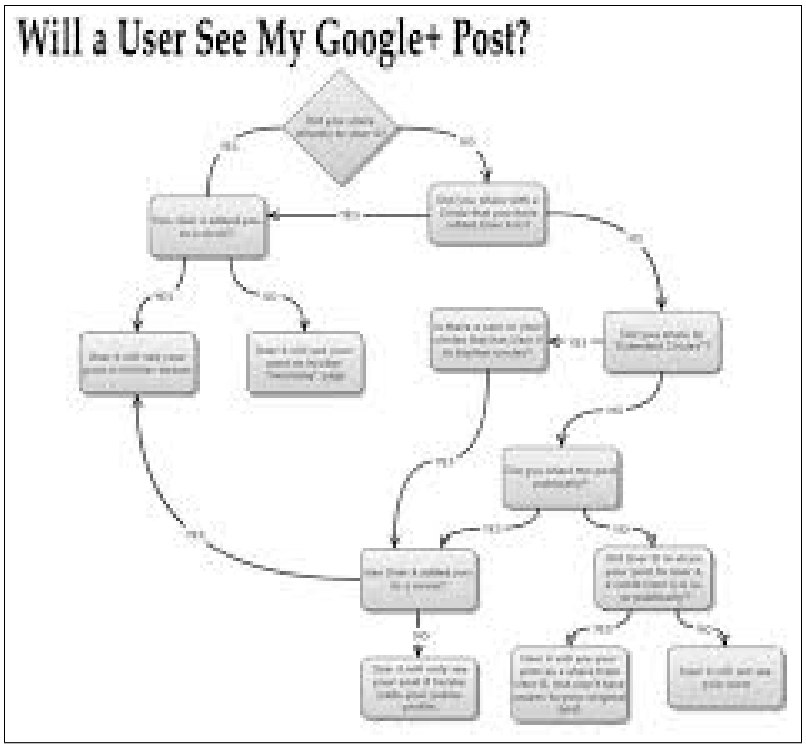


Figure 2.7: Who can see your G+ post?

as, eventually everybody gets the update, but the healthcare providers get it first.

```
 /* Emergency Update SYSTEM */

 defproc EmergencyUpdate(healthlinkb,healthlinke,newslink,info) =
   healthlinks!(info).healthlinke?(info).NewsUpdate(newslink,info)

 defproc HealthCareRing(link,next) = link?(info).next!(info).
     HealthCareRing(link,next);
 defproc NewsUpdate(newslink,newschannel) = newslink?(info).
     NewsBroadcast(newschannel,info)
 defproc NewsBroadcast(newschannel) = newschannel!(info).
     NewsBroadcast(newschannel,info)

 defproc System =
   new secret in
   new hlink1, hlink2, hlink3, hlink4 in
   new newslink, newschannel in
       (
           EmergencyUpdate(hlink1,hlink4,newslink,secret)
           | HealthCareRing(hlink1,hlink2)
           | HealthCareRing(hlink2,hlink3)
           | HealthCareRing(hlink3,hlink4)
           | NewsUpdate(newslink,newschannel)
       );

 /* PROPERTIES */

 check System |= 5 and (<>4);

 /***/

 defprop everywhere(A) = (false || (1 => A));

 defprop everybody_knows(secret) = everywhere(@secret);

 defprop everybody_gets_to_know =
   hidden secret.eventually everybody_knows(secret);

 defprop no_news_before_health_update(newschannel,healthlink) =
   not( <newschannel><healthlink>true )

 check System |= everybody_gets_to_know;

/* ---------- */
```

Each `defproc` declaration is a declaration of information flow policy. It is important to understand that each such policy statement can be provided *independently* by the entity that offers and conforms to that policy. Thus, in the example above, the CDC can offer its policy, the members of the health care provider network can offer theirs and the news agency members can offer theirs. The system is the *composition* of each such independently provided policy statement. This forms the basis for a practical smart contract system in a distributed system. There is no requirement for a global policy statement. Rather each participating party says what their requirements and guarantees are to participate in an engagement. Additionally, there is no requirement that the contract be 2-party, they can be n-party contracts.

Another crucial aspect of this approach is that the composite policy statement can be checked for properties, such as those described in the `defprop` declarations above. Again, these properties could be provided independently. Thus, in the example, the news agents can profer the property that everybody eventually gets to know the update, which is in their interest and alignment with their function as a service provider. Meanwhile, the CDC and health care provider can insist no news updates until all the health care providers in the network are alerted. This conforms with the CDC's mandate and the health care providers' obligation to promote public safety. The composite policy (the `System` declaration in the example above) can be automatically (i.e. mechanically) checked for these properties independently or in a variety of conjunctions.

Perhaps the most crucial aspect of this approach is that policy statements can be turned into system-level code. The relevant behavior is `proc` declarations can be installed as a part of an emergency update automation for Synereo. Thus, Synereo becomes a platform supporting an entirely new level of control over information flow. Users, programmers, vendors can extend Synereo in new and exciting ways while still providing an unprecedented level of visibility and control over what those extensions mean for information flow and the attention economy.

2.2.10 Comparison to other models

Synereo's network model is not the only way to provide a mechanism for improving signal in a network of communicating entities. Alternatives abound. In the field of biology, one of the best known models is so-called kinetic proofreading.

Kinetic proofreading

Review of kinetic proofreading The wikipedia article on kinetic proofreading summarizes the model this way:

> Kinetic proofreading (or kinetic amplification) is a mechanism for error correction in biochemical reactions, proposed independently by John Hopfield ... and Jacques Ninio ... . Kinetic proofreading allows enzymes to discriminate between two possible reaction pathways leading to correct or incorrect products with an accuracy higher than what one would predict based on the difference in the activation energy between these two pathways. Increased specificity is obtained by introducing an irreversible step exiting the pathway, with reaction intermediates leading to incorrect products more likely to prematurely exit the pathway than reaction intermediates leading to the correct product. If the exit step is fast relative to the next step in the pathway, the specificity can be increased by a factor of up to the ratio between the two exit rate constants. (If the next step is fast relative to the exit step, specificity will not be increased because there will not be enough time for exit to occur.) This can be repeated more than once to increase specificity further.

[45] [46]

Perhaps more colorfully, Mike Stay describes it this way.

> In molecular biology, everything's a hot mess. Molecules vibrate wildly, stuff is churning around at random. How does a cell manage to replicate DNA in such chaotic conditions?

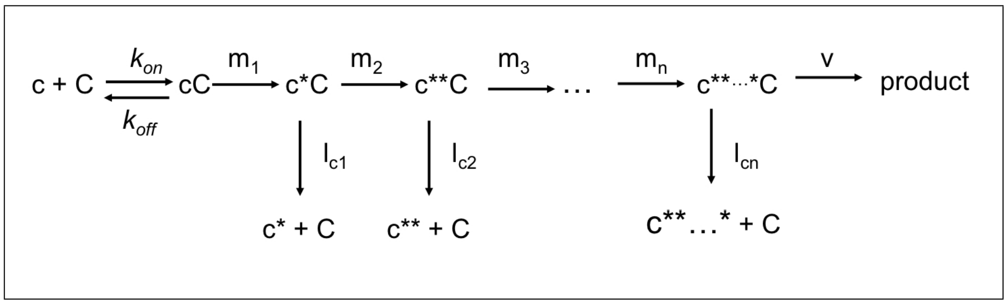


Figure 2.8: An example reo calculation protocol

> Part of the answer is a process called "kinetic proofreading." Imagine an art museum with a Rembrandt gallery, and suppose that fans of Rembrandt will spend ten minutes in the gallery, while non-fans will only spend seven. Suppose also that Rembrandt fans are relatively rare, perhaps 10% of the visitors. If we allow people to walk through the gallery at will, we'll rarely get more than around 25% of the people in the gallery to be fans.
>
> However, suppose that the entrance to the gallery can be shut for a time. If we shut it for eight minutes, then all the non-fans leave and everyone in the gallery will be a Rembrandt fan.
>
> Enzymes are chemicals that make reactions more likely to happen but don't get consumed in the reaction. Over a broad range, a higher proportion of enzymes to other molecules means a lower activation energy for the reaction. Kinetic proofreading is a process by which the proportion of enzymes can be increased tremendously, just like the proportion of Rembrandt fans in the gallery.
>
> The cost of kinetic proofreading is the energy required to keep the door shut. By opening and shutting a series of doors, a cell can use a much smaller amount of energy to get desirable outcomes.

Social network interpretation How does this relate to social networks? Here's a way to think about the application of kinetic proofreading processes in a social network. Consider a two stage kinetic proofreading process.

$$C + E \Leftrightarrow CE \rightarrow C^{*}E \rightarrow C^{**}E \rightarrow \text{product} + E$$
$$C^{*}E \rightarrow C^{*} + E$$
$$C^{**}E \rightarrow C^{**} + E$$

Think of E as a social network user. Think of C as content. Think of CE as the content showing up in the user's stream. Think of the fall-off reaction $CE \rightarrow C + E$ as the user never paying attention to the content post. Think of $C^{*}E$ as the user liking the post. Think of the fall-off reaction $C^{*}E \rightarrow C^{*} + E$ as the user liking the content post but never resharing. Think of $C^{**} + E$ as the user AMPlifying the content, compensating the content poster. Think of the fall-off reaction $C^{**}E \rightarrow C^{**} + E$ as the user liking and AMPlifying, but never resharing. Finally, think of $C^{**}E \rightarrow \text{product} + E$ as the user resharing.

Under this interpretation we can see that the cost of applying attention to content functions remarkably like a kinetic proofreading proofreading mechanism, informing the underlying mechanism of information flow in social networks and allowing them to improve signal fidelity. Specifically, when Facebook introduced the "like" mechanism whereby users could take an action that effectively marked attention deployed on a post, and thereby promote the post to a new status, they were knowingly or otherwise employing a form of kinetic proofreading to improve signal in the sense that content was more likely to flow to users with those with preferences for that kind of content because of this promotion mechanism and the cost of attending enough to click a button.

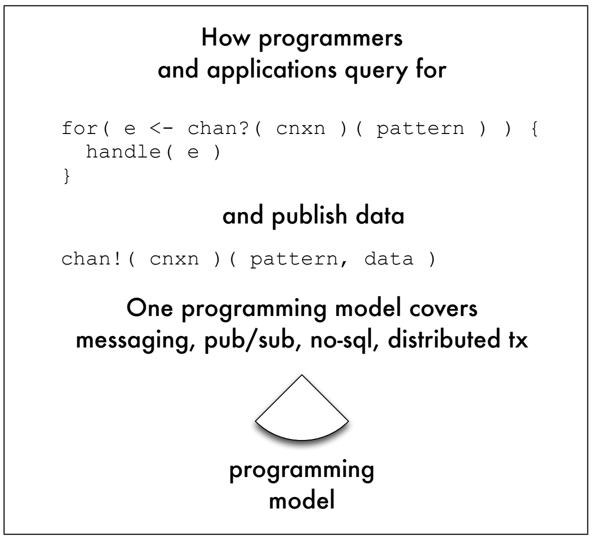
Again, though, why do we care about this interpretation? The interpretation becomes relevant if we want to quantify how much the signal improves. In particular, kinetic proofreading is really a recipe for turning energy into signal fidelity, and by recipe we mean one with exact proportions. The stoichiometric analysis provides a set of differential equations that when given the reaction rates say exactly what the energy cost is for a given improvement in signal. Using the social network interpretation kinetic proofreading provides a quantitative model of an attention economy. As such, it stands as a competitor to the Synereo model. That's why we care.

Comparison As mentioned in section 2.2.6, we hope to publish quantitative comparison results in a subsequent paper. More qualitatively, the question is, does Synereo's network model do anything to improve this basic mechanism? The short answer is that Synereo includes each of the attention ratcheting mechanisms and amplifies them. However, it balances that with AMPs. This acts as a kind of catalyst or enzyme working in the opposite direction to the natural signal improvement.

2.3 Software Stack

As the market begins to recognize the necessity of distributed applications and distributed platforms, it must simultaneously recognize that distributed applications, especially those that are also decentralized, are much harder to build. It's not just about the comms layer, or getting the bits from here to there. It's about having a conceptual apparatus for envisioning these kinds of applications and reducing that vision to practice. In short, what's needed is a programming model. Much in the same way that they provide a formal model for reasoning about concurrent and distributed systems, the mobile process calculi also provide such a programming model.

To be clear, we are not just talking about automated verification or simulation. Certainly, there are tools for reasoning about protocols specified in the applied π-calculus. One is called ProVerif [47]. Another such tool is the spatial logic model checker [48]. Additionally, there are tools for taking π-calculus specifications and simulating them at scale. One such tool, which has been mostly applied to reasoning about protocols (read signaling chains) in biological systems, is called SPIM [49]. However, there are a variety of runtime libraries and



How programmers
and applications query for

```
for( e <- chan?( cnxn )( pattern ) ) {
  handle( e )
}
```

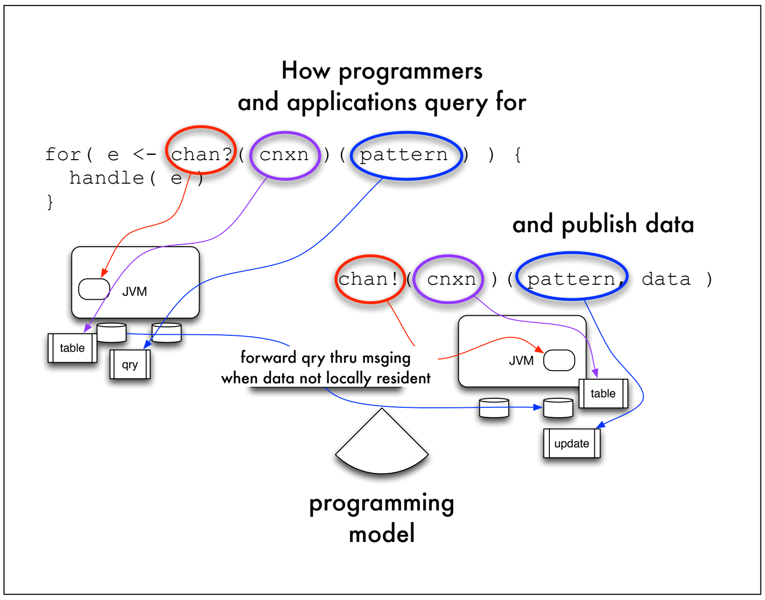
and publish data

```
chan!( cnxn )( pattern, data )
```

One programming model covers
messaging, pub/sub, no-sql, distributed tx

programming
model

Figure 2.9: SpecialK programming model



How programmers
and applications query for

```
for( e <- chan?( cnxn )( pattern ) ) {
  handle( e )
}
```

and publish data

```
chan!( cnxn )( pattern, data )
```

Figure 2.10: SpecialK distributed programming model

virtual machines that implement an execution model for the π-calculus, these include [50] and [51].

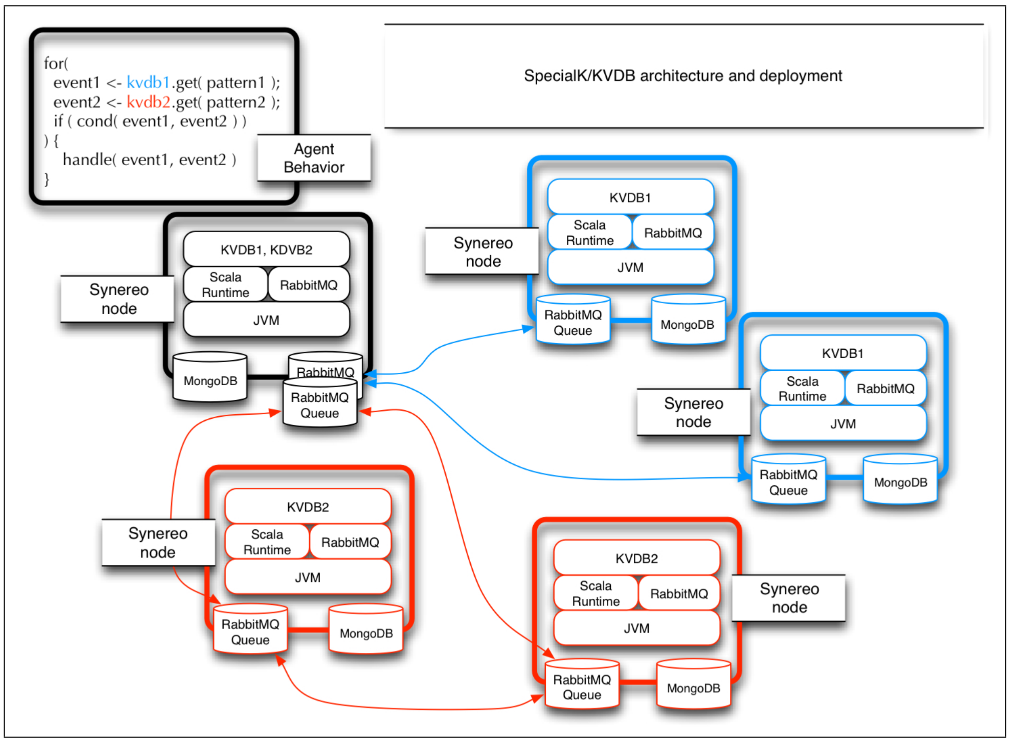
Of some interest here is SpecialK [52]. Like [50] and [51], SpecialK/KVDB takes the idea of ProVerif a step further by treating the applied π-calculus as a programming language. This allows designers and programmers alike to go from high level specifications directly to code that runs in a distributed setting. Unlike these other implementations, it does this by embedding a monadic DSL for the applied π-calculus into `Scala`. The core of this monadic DSL treats a π-calculus channel as if it were logically a persistent queue. Thus, where one would write $x?(y)P$ in the π-calculus, the equivalent expression in `Scala` extended with the SpecialK/KVDB library would be `for( y <-x ){ P }`; likewise, `x !(y)` is the `Scala` transliteration of the π-calculus expression $x!(y)$. Figure 2.10 shows an elaboration of this scheme for the *DendroNet* model.

Further, because of its mapping of the π-calculus semantics onto local storage and messaging layers, using the SpecialK/KVDB `Scala` library a programmer can turn π-calculus expressions into `Scala` programs that execute in a distributed manner. SpecialK is the DSL while KVDB comprises the underlying libraries and architecture that allow the distributed execution.

2.3.1 Example protocol implementation

The following code fragment encodes equations 2.18 and 2.19. A more complete (but still undocumented) version can be found in github at [53].

```scala
trait ContentRecipientBehaviorT extends ProtocolBehaviorT with
    Serializable {
  ...
def receiveContent(
  node: SynereoNode[PersistedKVDBNodeRequest,
      PersistedKVDBNodeResponse],
  cnxns: Seq[PortableAgentCnxn]
): Unit = {
  cnxns match {
    case recipientToSomebody :: _ => {
      val agntRecipientCnxnsRdWr =
        for( cnxn <- cnxns ) yield {
          (
            acT.AgentCnxn( cnxn.src, cnxn.label, cnxn.trgt ),
            acT.AgentCnxn( cnxn.trgt, cnxn.label, cnxn.src )
          )
        }

      for( ( agntCnxn, _ ) <- agntRecipientCnxnsRdWr ) {

        reset {
          for( ePublishWithReo <- node.subscribe( agntCnxn )(
              PublishWithReo.toLabel ) ) {
            rsrc2V[ReputationMessage]( ePublishWithReo ) match {
              case Left( PublishWithReo( sidPC, cidPC, p2cPC,
                  cntntPC, reoPC ) ) => {

                val agntP2CPC = acT.AgentCnxn( p2cPC.src, p2cPC.
                    label, p2cPC.trgt )

                val calulateReoRequest = CalculateReo( sidPC,
                    cidPC, p2cPC )
                node.publish( agntP2CPC )(
                  CalculateReo.toLabel( sidPC ),
                  calulateReoRequest
                )
                for( eHashedConnections <- node.subscribe( agntCnxn )( HashedConnections.toLabel ) ) {
                  rsrc2V[ReputationMessage]( eHashedConnections ) match
                  {
                    case Left( HashedConnections( sidPC, cidPC, seqOfCnxnsPC ) ) => {

                      val hashedConnections : Seq[PortableAgentCnxn] = provideHashedConnections()
                      val hashedConnectionsData = HashedConnections( sidPC, cidPC, hashedConnections )
                      node.publish( agntP2CPC )(
                        HashedConnections.toLabel( sidPC ),
                        hashedConnectionsData
                      )

                      val intersectionConnections : Seq[PortableAgentCnxn] = hashedConnections intersect seqOfCnxnsPC;
                      val intersectionConnectionsData = IntersectionResult( sidPC, cidPC, intersectionConnections);
                      for( eIntersectionResult <- node.subscribe( agntCnxn )( IntersectionResult.toLabel ) ) {
                        rsrc2V[ReputationMessage]( eIntersectionResult ) match
                        {
                          case Left( IntersectionResult( sidPC, cidPC, intersectCnxnsPC ) ) => {
                            node.publish( agntP2CPC )(
                              IntersectionResult.toLabel( sidPC ),
                              intersectionConnectionsData
                            )
                            val reoScore : Int = 1000; // placeholder for actual calculation
                            if ( reoScore != reoPC ) {
                            }
                            // reo is good, continue to publish
                          }
                          case Right( true ) => {
                            ...
                          }
                          case _ => {
                            ...
                          }
                        }
                      }
                    }
                    case Right( true ) => {
                      ...
                    }
                    case _ => {
                      ...
```



```
for(
  event1 <- kvdb1.get( pattern1 );
  event2 <- kvdb2.get( pattern2 );
  if ( cond( event1, event2 ) )
) {
    handle( event1, event2 )
}
```

Figure 2.11: SpecialK/KVDB deployment

```
                      }
                    }
                  }
                }
                case Right( true ) => {
                  ...
                }
                case _ => {
                  ...
                }
              }
            }
          }
        }
      }
      case _ => {
        throw new Exception( "at least one cnxn expected : " +
            cnxns )
      }
    }
  }
}
```

2.3.2 Deployment

Deployment of a Synereo node currently requires three core dependencies: JVM, MongoDB and RabbitMQ. However, there are plans for reducing all the Synereo deliverables to a single binary, with all the dependencies running in-process. Regardless of in-proc or external dependencies, Synereo will be delivered fully packaged, targeting Linux platforms initially (for all major distributions), with OS X and Windows platforms following suit. Any required services will be launched from the respective init system (e.g. systemd for Linux, plist for OS X, Windows services), thus requiring very little interaction from the user. Later releases targeting a general availability audience will have a matching user interface that simplifies usage for non-technical users, and does not require any command line knowledge. Synereo does not target full deployment on mobile clients at this phase, but will enable usage through centralised gateway service (see section 2.5.1).

2.3.3 Security

Synereo's security and privacy design is based on using tested and proven methods. New cryptography primitives or protocols are out of scope for Synereo, as they are notoriously hard to implement correctly, nor does Synereo actually require any new tools beyond the currently available methods that are known to be time-tested and battle-proven.

Messages in transit between Synereo nodes are encrypted using RabbitMQ's builtin SSL capabilities [54]. (A possible switch to ZeroMQ as core transport for Synereo messages will require an additional dependency for securing messages, as it does not have native SSL/TLS support. In this case, libNaCL [?] is a strong candidate for enabling secure messaging between nodes. Incidentally, libNaCL is used by CurveZMQ, the builtin protocol used for ZMQ authentication and encryption.) Keys are exchanged opportunistically between nodes, and no certificate authority is used to verify certificate authenticity. This is due to the centralised nature of a CA, which goes against Synereo's vision for a completely decentralised network. Of course, the lack of certification authentication is ripe for man-in-the-middle (MITM) attacks. Generally speaking, secure communication between two peers without prior agreement on a shared secret is a hard problem that is pervasive across many applications. Some techniques can be used to reduce the attack surface of Synereo nodes, such as pinning known certificates for each node in a local cache, and raising errors on any fingerprint anomaly, similar to the behaviour used, for example, in the OpenSSH implementations. Security-aware users can also make a habit of confirming node fingerprints upon first contact through an existing trusted channel. Later iterations might bring more attention to securing messages in transit in stronger ways that have more favourable trade-offs for the Synereo network.

2.4 Censorship Resilience

As with any new technology that challenges existing structures of power and profit, there must be some expectation for resistance and plans must be made accordingly. To that end, we review the resilience of existing peer-to-peer technologies to censorship attempts, and discuss Synereo's plans in this regard.

It is well known that where there is censorship, there first is surveillance. Dragnet surveillance of internet traffic - as we have come to learn through recent revelations by Edward Snowden and other courageous whistleblowers - generally classifies traffic roughly by type, and considers most P2P traffic - that is not otherwise targeted for selection - to be "high-volume, low-quality" [55]. Generally speaking, a passive adversary who is interested in conducting censorship operations, and has access to internet routing equipment (such as ISPs, law enforcement authorities, governments and spy agencies) can easily distinguish between different types of internet traffic, even if data is encrypted, by looking at patterns of $< SrcIP, SrcPort, DestIP, DestPort, Data >$ and making some intelligent guesses. P2P traffic, as well, can be analyzed in such a way. Thus, the first question is if Synereo traffic patterns can blend in with other P2P traffic, effectively making it very convenient to hide Synereo content in swathes of otherwise innocuous data. While not an insignificant effort, initial work to make Synereo more resilient to censorship will likely start with this target in mind. Additional efforts to conceal Synereo traffic can follow the path set by so-called "Pluggable Transports" which are used by the Tor network [56] to obfuscate Tor protocol traffic as a multitude of combinations of target protocols and providers such as SSL/TLS, Skype, Google and CloudFlare.

A persistent adversary that is interested in blocking Synereo at all costs will then have to consider blocking large portions of internet traffic, and deal with the political consequences of such an action. Wireless mesh networks to the rescue!

Wireless mesh networks - such as Freifunk (Germany), FunkFeuer (Austria), AWMN (Athens) and Guifi (Catalonia) - are community-based efforts to create an internet backbone based on off-the-shelf consumer wireless equipment. These networks are able to serve large geographical areas, spanning neighbourhoods, cities and even large metropolitan areas, and provide not only a shared uplink to the Internet, but can also serve as a self-sustained network serving the community in case no uplink exists, or in case of a crisis. Given a large enough user base in such an ad-hoc network, future work on Synereo will attempt to enable self-sustained deployment of isolated Synereo networks on such networks, effectively creating not one, but *many* Synereo networks that later able to merge back into the rest of the network once the political situation is more favourable.

2.5 Services

2.5.1 Gateway services

As part of the deployment of the Synereo network, we plan on enabling the setup of centralised Synereo gateways. Of course, while this is not the main focus of the Synereo network, we aim to provide an easy way for users with limited ability to deploy their own Synereo nodes, to use the network regardless. This is achieved through centralised services that will provide the necessary technical resources to enable end users to use Synereo as a simple web service, and access it with nothing more than a standard web browser.

Implementing centralised gateways requires multi-tenancy capabilities from the Synereo binaries, such that properties of security and anonymity are fulfilled to a strong enough degree. Multi-tenant deployments can be implemented over many architectures, and discussion of those options will be deferred to a later paper, as it becomes more relevant to the Synereo roadmap.

2.5.2 Third party services

As hinted throughout this paper, Synereo does not reside in a vacuum, and will rely on several surrounding technologies. The AMP ledger will be managed on a standard blockchain technology, either on a native Synereo blockchain to be deployed side-by-side next to the Synereo stack, or otherwise dependent on a third party service that enables quick and fast transactions on an existing blockchain. The Synereo ecosystem is based on many micro-transactions representing values of reputation and charge and thus has specific requirements from a highly performant ledger that carries minimal per-transaction overhead. To name an example, Factom [57] is a service which plans on delivering just that, and is a likely candidate for integration with the Synereo AMP ledger.

Another service that will possibly be integrated into the Synereo network is MaidSafe [58], which - amongst other offerings - provides a distributed content store solution. Synereo might rely on such a service to act as a so-called distributed content delivery network (CDN) where static objects are to be served from neighbouring nodes.

2.6 Conclusion

The reader who has made it this far is either a sly person who skipped to the end or someone whose attention is unlikely to be hacked and whose mind is spacious enough to hold onto the forest *and* the trees. Either way, congratulations! You made it here all in one piece. We've tried to make it clear that we're with you all the way. We know that social interaction over, around, and through the Internet has an enormous potential that we've only just begun to explore. We know that tapping into that potential means helping you, both the sly and spacious, and everyone in between, benefit from how you deploy your attention; and, more importantly, tappng into that potential means providing you a means to help your whole community benefit from a coherent, focused deployment of the collective attention. Quite simply, given the problems facing us at this point in time on planet Earth, we need a different quality of engagement if we are to make it through.

As we promised at the end of 1, this isn't a manifesto. We didn't lay out these lofty goals and then not tell you how we plan to get there. Instead, we've done our level best to provide a model for tapping into the collective attention and a map from that model onto a real distributed architecture with an actual programming model that has a solid mathematical foundation. (In fact we even provide not one, but two different ways to analyze the model, and compared it to possible competing models.) Needless to say, it's a big undertaking; but, we think it's unquestionably a worthwhile undertaking with a very sound underpinning that provides the basis for a step-by-step realization.

In point of fact, as biological models like kinetic proofreading make clear, a distributed approach to self-organization isn't a revolution. It's a time honored approach to evolution; and that's what this really is: an invitation to join the evolution.

Bibliography

[1] Adam D. I. Kramer, Jamie E. Guillory, and Jeffrey T. Hancock. Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences*, 111(24):8788–8790, 2014.

[2] Peter Nguyen. Social media privacy policy loopholes you need to know about. `http://blog.hotspotshield.com/2013/06/25/social-media-privacy-concerns/`, 2013.

[3] James Bamford. Edward snowden: The untold story. *Wired*, 2014.

[4] Statista. Number of social network users worldwide from 2010 to 2018 (in billions). http://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/.

[5] InternetLiveStats. Internet users in the world. http://www.internetlivestats.com/watch/internet-users/.

[6] Dan Noyes. The top 20 valuable facebook statistics. https://zephoria.com/social-media/top-15-valuable-facebook-statistics/.

[7] Leo Mirani. Millions of facebook users have no idea they're using the internet. http://qz.com/333313/milliions-of-facebook-users-have-no-idea-theyre-using-the-internet/.

[8] Inc Facebook. Facebook reports fourth quarter and full year 2014 results. http://investor.fb.com/releasedetail.cfm?ReleaseID=893395.

[9] Inc Twitter. Twitter reports fourth quarter and fiscal year 2014 results. https://investor.twitterinc.com/releasedetail.cfm?releaseid=894844.

[10] Gideon Rosenblatt. Your second job, as a facebook user. http://www.the-vital-edge.com/facebook-users/.

[11] Adrian Covert. Facebook buys whatsapp for 19 billion usd. http://money.cnn.com/2014/02/19/technology/social/facebook-whatsapp/index.html?iid=EL.

[12] Statista. Value per active user of selected social networks as of february 2014, based on market cap or acquisition price (in u.s. dollars). http://www.statista.com/statistics/289505/social-networks-value-per-active-user/.

[13] Reed Albergotti. Furor erupts over facebook's experiment on users. http://www.wsj.com/articles/furor-erupts-over-facebook-experiment-on-users-1404085840.

[14] Facebook community question. How does facebook track my recent non-facebook web-browsing. https://www.facebook.com/help/community/question/?id=10151697384894188.

[15] Harrison Weber. How the nsa and fbi made facebook the perfect mass surveillance tool. http://venturebeat.com/2014/05/15/how-the-nsa-fbi-made-facebook-the-perfect-mass-surveillance-tool/.

[16] Paul Boutin. Anti-facebook project rockets to 120,000 usd in online donations. http://venturebeat.com/2010/05/13/anti-facebook-project-rockets-past-120000-in-funding/.

[17] Steven Tweedie. Ello raises 5.5 usd million for its invite-only, ad-free social network. http://www.businessinsider.com/ello-raises-55-million-for-its-social-network-2014-10.

[18] R.A. Lanham. *The economics of attention: Style and substance in the age of information.* Univerity of Chicago Press, 2006.

[19] T. Terranova. Attention, economy and the brain. *Culture Machine*, 13:1–19, 2012.

[20] Wayne Wu. Attention as action for selection. *Attention: Philosophical and psychological essays*, pages 97–116, 2011.

[21] Wayne Wu. *Attention.* Routledge, London, UK, 2014.

[22] H.A. Simon. Designing organizations for an information-rich world. In M. Greenberger, editor, *Computers, communications, and the public interest*, pages 37–72. The Johns Hopkins Press, 1971.

[23] R. Mortier, H. Haddadi, T. Henderson, D. McAuley, and J. Crowcroft. Human-data interaction: The human face of the data-driven society. *Social Science Research Network Working Paper Series*, 2014.

[24] A. Sen. Well-being, agency and freedom: the dewey lectures 1984. *The Journal of Philosophy*, pages 169–221, 1985.

[25] David Foster Wallace. Some thoughts, delivered on a significant occasion, about living a compassionate life. In *This is water.* Little, Brown and Company, 2014.

[26] Ian Bogost. Cow clicker: The making of obsession. `http://bogost.com/writing/blog/cow_clicker_1`, 2010.

[27] J. Blow. Video games and the human condition. *CS Colloquium: Rice University*, 2010.

[28] M. Soroush, M. Hancock, and V.K. Bohns. Self-control in casual games. In *Proceedings of the IEEE Games, Entertainment, and Media (GEM) Conference 2014*, 2014.

[29] D. Estrada and J. Lawhead. Gaming the attention economy. In *Handbook of Human Computation*, pages 961–978. Springer, New York, NY, 2013.

[30] Bateson. ???????? *???????*, 2014.

[31] Paul Adams. Why cards are the future of the web. `http://blog.intercom.io/why-cards-are-the-future-of-the-web/`.

[32] Carl Hewitt. Actor model for discretionary, adaptive concurrency. *CoRR*, abs/1008.1459, 2010.

[33] Vincent Danos, Josée Desharnais, and Prakash Panangaden. Labelled markov processes: Stronger and faster approximations. *Electr. Notes Theor. Comput. Sci.*, 87:157–203, 2004.

[34] Silvano Dal Zilio. Mobile processes: A commented bibliography. *Lecture Notes in Computer Science*, 2067:206–??, 2001.

[35] Hendrik Pieter Barendregt. *The Lambda Calculus – Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984.

[36] Cédric Fournet and Martín Abadi. Hiding names: Private authentication in the applied pi calculus. In Mitsuhiro Okada, Benjamin C. Pierce, Andre Scedrov, Hideyuki Tokuda, and Akinori Yonezawa, editors, *Software Security – Theories and Systems, Mext-NSF-JSPS International Symposium, ISSS 2002, Tokyo, Japan, November 8-10, 2002, Revised Papers*, volume 2609 of *Lecture Notes in Computer Science*, pages 317–338. Springer, 2002.

[37] Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In Shmuel Sagiv, editor, *Programming Languages and Systems, 14th European Symposium on Programming,ESOP 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200. Springer, 2005.

[38] Lee Swagerty. Will a user see my google+ post? `http://ansonalex.com/tutorials/will-a-user-see-my-post-on-google-plus-google-flowchart/`, 2011.

[39] Marco Bernardo and Stefania Botta. A survey of modal logics characterising behavioural equivalences for non-deterministic and stochastic systems. *Mathematical. Structures in Comp. Sci.*, 18(1):29–55, February 2008.

[40] A. Ponse and Scott A. Smolka, editors. *Handbook of Process Algebra.* Elsevier Science Inc., New York, NY, USA, 2001.

[41] L. Gregory Meredith and Matthias Radestock. Namespace logic: A logic for a reflective higher-order calculus. In *TGC* [59], pages 353–369.

[42] Luís Caires. Behavioral and spatial observations in a logic for the pi-calculus. In *FoSSaCS*, pages 72–89, 2004.

[43] Martin Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *4th ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.

[44] Greg Meredith and Steve Bjorg. Contracts and types. *Commun. ACM*, 46(10):41–47, 2003.

[45] Wikipedia. Kinetic proofreading — Wikipedia, the free encyclopedia, 2014. [Online; accessed 23-May-2014].

[46] JJ Hopfield. Kinetic proofreading: a new mechanism for reducing errors in biosynthetic processes requiring high specificity. *Proc. Natl. Acad. Sci. U.S.A.*, 71 (10):4135–9, 1974.

[47] Bruno Blanchet. Proverif. `http://proverif.rocq.inria.fr/`. Online prover.

[48] Luis Caires. Spatial logic model checker. `http://ctp.di.fct.unl.pt/SLMC/`. webpage.

[49] Andrew Phillips. Spim. `http://research.microsoft.com/en-us/projects/spim/`. webpage.

[50] Benjamin C. Pierce and David N. Turner. Pict: A programming language based on the pi-calculus. In Gordon Plotkin, Colin Stirling, and Mads Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner*, pages 455–494. MIT Press, 2000.

[51] Louis Mandel and Luc Maranget. Programming in JoCaml — Extended Version. Research Report RR-6261, INRIA, 2007.

[52] Lucius G. Meredith. Specialk scala library. `https://github.com/synereo/special-k`. Launched April 2013.

[53] Yuval Adam Lucius G. Meredith. Sample reo calculation protocol. `https://github.com/synereo/agent-service-ati-ia/tree/master/AgentServices-Store/src/main/scala/com/protegra_ati/agentservices/protocols/reputation`. Written Fall 2014.

[54] Rabbitmq configuration. `https://www.rabbitmq.com/configure.html`.

[55] Ewen MacAskill et. al. Gchq taps fibre-optic cables for secret access to world's communications. `http://www.theguardian.com/uk/2013/jun/21/gchq-cables-secret-world-communications-nsa`, 2013.

[56] The Tor Project. Tor: Pluggable transports. `https://www.torproject.org/docs/pluggable-transports.html.en`, 2015. [Online; accessed 21-Jan-2015].

[57] Paul Snow, Brian Deery, Jack Lu, David Johnston, and Peter Kirby. Business processes secured by immutable audit trails on the blockchain. `https://github.com/FactomProject/FactomDocs/blob/master/Factom_Whitepaper.pdf?raw=true`, 2014.

[58] Tonglin Li, Xiaobing Zhou, Kevin Br, Dongfang Zhao, Ke Wang, Anupam Rajendran, Zhao Zhang, and Ioan Raicu. Zht: A light-weight reliable persistent dynamic scalable zero-hop distributed hash table. In *In Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium (IPDPS*, 2013.

[59] L. Gregory Meredith and Matthias Radestock. A reflective higher-order calculus. *Electr. Notes Theor. Comput. Sci.*, 141(5):49–67, 2005.

[60] Martin Odersky. Polarized name passing. In P. S. Thiagarajan, editor, *FSTTCS*, volume 1026 of *Lecture Notes in Computer Science*, pages 324–337. Springer, 1995.

[61] Matej Košík. *A Contribution to Techniques for Building Dependable Operating Systems.* PhD thesis, Slovak University of Technology in Bratislava, May 2011.

[62] Robin Milner. The polyadic π-calculus: A tutorial. *Logic and Algebra of Specification*, Springer-Verlag, 1993.

2.7 Appendix A: A brief introduction to the π-calculus

In this appendix we present a reflective version of the π-calculus. We choose this version primarily because – although it has more features – it's actually simpler to present. It's a smaller calculus, with fewer moving parts; and, there is a full and faithful translation of the plain vanilla π-calculus into this version. Before giving the formal presentation of the calculus, we begin with an example that illustrates a design pattern used over and over in software design and implementation: a mutable single-place cell for storing and retrieving state.

```
def Cell(slot, state) ⇒ {
  new (v) {
    v!(state)
    match {
      slot?get(ret) ⇒ {
        v?(s) ⇒ ret!(s)
        Cell(slot, s)
      }
      slot?set(s) ⇒ { Cell(slot, s) }
    }
  }
}
```

We read this as saying that a *Cell* is parametric in some (initial) *state* and a *slot* for accessing and mutating the cell's state. A *Cell* allocates a private channel v where it makes the initially supplied *state* available to its internal computations. If on the channel *slot* it receives a *get* message containing a channel *ret* indicating where to send the state of the cell, it accesses the private channel v and sends the value it received on to the *ret* channel; then it resumes behaving as a *Cell*. Alternatively, if it receives a *set* message containing some new state s, it simply continues as a *Cell* instantiated with accessor *slot* and state s.

Zooming out a bit we can write down a recursive description of all legal RHO-calculus programs.

- the io-bound processes (M or N)
  - 0 represents the stopped or inert process
  - if P is a process, then $x?(y_1, \ldots, y_N) \Rightarrow P$ represents an input-guarded process, or a process that is blocked waiting for input on channel, x, before becoming P; note that the input will be a tuple of channels $(y_1, \ldots, y_N)$
  - if $Q_1, \ldots, Q_N$ are processes, then $x!(Q_1, \ldots, Q_N)$ represents a process that is asynchronously sending as output a tuple of the code for processes $(Q_1, \ldots, Q_N)$
  - if M and N are io-bound processes, then $M+N$ represents a process that makes a non-deterministic choice between becoming the process M or the process N
- general processes (P or Q)
  - every io-bound process M is a general process
  - if P and Q are processes, then $P \mid Q$ represents the process in which P and Q are running concurrently
  - if x is the code for a process, then $*x$ represents the process created by executing that code
- channels such as x are identified with the code for a process $@P$

The rho-calculus

The RHO-calculus [59] is a variant of the asynchronous polyadic π-calculus. When names are polarized [60], the π-calculus is an ocap language. Pierce and Turner [50] defined the programming language Pict as sugar over the polarized π-calculus together with some supporting libraries; unfortunately, the authority provided by the libraries violated the principle of least authority. Košík refactored the libraries to create Tamed Pict, recovering an ocap language, then used it to write a defensively consistent operating system [61]. The RHO-calculus differs from the π-calculus in that names are quoted processes rather than generated by the ν operator. Both freshness and polarization are provided through the use of namespaces at the type level.

We let P, Q, R range over processes and x, y, z range over names, and $\vec{x}$ for sequences of names, $|\vec{x}|$ for the length of $\vec{x}$, and $\{\vec{y}/\vec{x}\}$ as partial maps from names to names that may be uniquely extended to maps from processes to processes [59].

$M, N ::= 0$	stopped process
$\mid x?(y_1, \ldots, y_N) \Rightarrow P$	input
$\mid x!(Q_1, \ldots, Q_N)$	output
$\mid M{+}N$	choice
$P, Q ::= M$	include IO processes
$\mid P \mid Q$	parallel
$\mid *x$	dereference
$x, y ::= @P$	reference

The examples from the previous section use mild (and entirely standard) syntactic sugar: `def` making recursive definitions a little more convenient than their higher-order encodings, `new` making fresh channel allocation a little more convenient and `match` for purely input-guarded choice. Additionally, the examples use {}-enclosed blocks together with line breaks, rather than | for parallel composition. Thus the expression $v!(\mathit{state})$ is actually a thread running in parallel with the `match` expression in the *Cell* definition. The principal difference between this and the applied π-calculus is replacing the tuples of names in guarded input terms $x?(y_1, \ldots, y_N) \Rightarrow P$ with Prolog terms. This is quite standard and presents no real technical difficulties. The interested reader is directed to [62], [36], and [59] for details.

2.7.1 Free and bound names

The syntax has been chosen so that a binding occurrence of a name is sandwiched between round braces, $(\cdot)$. Thus, the calculation of the free names of a process, P, denoted $\mathcal{FN}(P)$ is given recursively by

$$\begin{aligned}
&\mathcal{FN}(0) \coloneqq \varnothing \\
&\mathcal{FN}(x?(y_1, \ldots, y_N) \Rightarrow P) \coloneqq \\
&\quad \{x\} \cup (\mathcal{FN}(P) \setminus \{y_1, \ldots y_N\}) \\
&\mathcal{FN}(x!(Q_1, \ldots, Q_N)) \coloneqq \{x\} \cup \bigcup \mathcal{FN}(Q_i) \\
&\mathcal{FN}(P \mid Q) \coloneqq \mathcal{FN}(P) \cup \mathcal{FN}(Q) \\
&\mathcal{FN}(*x)\ x \coloneqq \{x\}
\end{aligned}$$

An occurrence of x in a process P is *bound* if it is not free. The set of names occurring in a process (bound or free) is denoted by $\mathcal{N}(P)$.

2.7.2 Structural congruence

The *structural congruence* of processes, noted $\equiv$, is the least congruence containing α-equivalence, $\equiv_\alpha$, that satisfies the following laws:

$$\begin{array}{rcl}
P \mid 0 & \equiv P \equiv & 0 \mid P \\
P \mid Q & \equiv & Q \mid P \\
(P \mid Q) \mid R & \equiv & P \mid (Q \mid R)
\end{array}$$

Name equivalence As discussed in [41] the calculus uses an equality, $\equiv_N$, pronounced name-equivalence, recursively related to α-equivalence and structural equivalence, to judge when two names are equivalent, when deciding synchronization and substitution.

2.7.3 Semantic substitution

The engine of computation in this calculus is the interaction between synchronization and a form of substitution, called semantic substitution. Semantic substitution differs from ordinary syntactic substitution in its application to a dropped name. For more details see [59]

$$(*x)\{\widehat{@Q/@P}\} \coloneqq \begin{cases} Q & x \equiv_N @P \\ *x & \textit{otherwise} \end{cases}$$

Finally equipped with these standard features we can present the dynamics of the calculus.

2.7.4 Operational Semantics

The reduction rules for RHO-calculus are

$$\frac{x_0 \equiv_N x_1, |\vec{y}| = |\vec{Q}|}{x_0?(\vec{y}) \Rightarrow P \mid x_1!(\vec{Q}) \to P\{@\vec{Q}/\vec{y}\}} \qquad (\textsc{Comm})$$

In addition, we have the following context rules:

$$\frac{P \to P'}{P \mid Q \to P' \mid Q} \qquad (\textsc{Par})$$

$$\frac{P \equiv P' \qquad P' \to Q' \qquad Q' \equiv Q}{P \to Q} \qquad (\textsc{Equiv})$$

The context rules are entirely standard and we do not say much about them here. The communication rule makes it possible for agents to synchronize at name-equivalent guards and communicate processes packaged as names. Here is a simple example of the use of synchronization, reduction and quasiquote: $x(z) \Rightarrow w!(y!(*z)) \mid x!(P) \to w!(y!(P))$. The input-guarded process, $x(z) \Rightarrow w!(y!(*z))$, waits to receive the code of P from the output-guarded data, $x!(P)$, and then reduces to a new process the body of which is rewritten to include P, $w!(y!(P))$.

2.7.5 The π-calculus to Scala in 1 page

For the developer the relationship of π-calculus to executable semantics is even simpler. Here's a translation of the plain vanilla π-calculus to `Scala` in 1 page.

This, plus delimited continuations, a little prolog-based pattern matching, and a mapping of that to datalog constitute the basis of SpecialK/KVDB's implementation of the applied π-calculus.

```
P,Q ::= 0                                   { }

     a![ v1, …, vn ]                        [| a |]( m ) ![ [| v1 |]( m ), …, [| vn |]( m ) ]

     a?( x1, …, xn )P                       for( [ x1, …, xn ] <- [| a |]( m ) ){
                                                [| P |]( m )( x1, …, xn )
                                            }

     P | Q                                  spawn{ [| P |]( m ) };spawn{ [| Q |]( m ) }

     (new a)P                               { val q = new Queue(); [| P |]( m[ a <- q ] ) }

     ( def X( x1, …, xn ) = P )[v1, …, vn]   object X {
                                               def apply( x1, …, xn ) = {
                                                 [| P |]( m ) ( x1, …, xn )
                                               }
                                             }

     X[v1, …, vn]                           X( [| v1 |]( m ), …, [| vn |]( m ) )

               [| - |]( - ) : ( π-calculus, Map[Symbol,Queue] ) -> Scala
```

Figure 2.12: π-calculus to ***Scala*** *in 1 page*