Chemical Networking Protocols

Thomas Meyer and Christian Tschudin
University of Basel, Switzerland
{th.meyer,christian.tschudin}@unibas.ch

ABSTRACT

Beyond the mere collection of computers, a network is the home of competing and cooperating execution flows. In this paper we show how to design network protocols based on molecule-like entities such that the corresponding execution flows can by analyzed as if they were chemical processes. Our goal is to create *robust protocol implementations* which are resilient to unreliable execution. We introduce the metaphor of chemical networking protocols, demonstrate its benefits by a formal stability analysis of a gossip-style protocol and present a first example of a self-healing load balancing protocol that is resilient to code removal attacks.

1 INTRODUCTION

Every now and then, network engineers are making use of chemical terms to describe properties of their protocols: TCP's congestion control algorithm bases on the "conservation of packets principle" trying to bring a connection to "equilibrium" [14]. AntNets stochastically routes packets proportional to "pheromone concentrations" [6]. Gossip-style protocols like [16] adopt the dynamics of epidemic spreading to disseminate information in a robust way. Apparently, chemically inspired ideas are considered to be beneficial for the dynamic behavior of protocols.

For the sake of stimulating the discussion, we are taking the extreme position of using chemistry as the central dogma for designing protocols and explore the potential of this approach. We aim at transposing properties of chemical systems to protocols that are hard to obtain otherwise. This includes autonomously finding equilibria at the point of optimal operation or the protocol's robustness to internal and external perturbations.

We introduce a programming model to describe computation as well as network communication as abstract chemical reactions and show two examples of "Chemical Networking Protocols" (CNPs). The first example "chemically calculates" the average of distributed values. Because of the analogy to chemical reaction networks, we can make use of analytical tools developed over decades in chemistry to predict the behavior of such systems, like for example Metabolic Control Analysis [13] or Chemical Organization Theory [7]. We will give an example for this analysis potential by formally proving the protocol's stability.

In the second part of this paper we ask whether some higher-level properties, such as the capability of biological systems to heal themselves, can be achieved in our model. This is important with regard to future execution environments that may be unreliable, such as probabilistic chips [4] or deep space probes, where the question is how to write communication software that tolerates spurious execution errors. In this paper we show how to construct self-replicating molecules that bring forward self-healing software. We apply this to CNPs and demonstrate a path load balancing protocol that is robust not only to the loss of packets but also to the deletion of code.

2 CHEMICAL REACTION MODEL

Traditionally, protocol execution is handled by a state machine that upon the reception of a packet synchronously changes its internal state and performs some communication activity. Here we introduce a "molecule metaphor" where each packet is treated as a virtual molecule. Virtual molecules react with other molecules in a reaction vessel (node). A reaction may produce other molecules being delivered to the application or being sent over the network. In such a chemical perspective, we obtain a web of reactions that together perform a distributed computation (called network service).

2.1 Modeling Chemical Communication

Instead of encoding a deterministic state machine, or having a sequential program that processes an incoming packet, each network node contains a multiset $\mathcal{M}(\mathcal{S})$ of a finite set of molecules $\mathcal{S} = \{s_1, \ldots, s_n\}$ (=packets). In addition, each node defines a set of reaction rules $\mathcal{R} = \{r_1, \ldots, r_m\}$ expressing which reactant molecules can collide and which molecules are generated during this process. Such a reaction is typically represented as follows:

$$i: \quad C_i + X_i \longrightarrow C_i + X_j \tag{1}$$

The above reaction in node i consumes, if present, two molecules C and X from the local multiset, regenerates C and sends molecule X to neighbor node j. In a simple two-node network topology, the above example spans the following reaction network, also depicted in Fig. 1:

$$1: C_1 + X_1 \xrightarrow{r_1} C_1 + X_2 \tag{2}$$

$$2: C_2 + X_2 \xrightarrow{r_2} C_2 + X_1 \tag{3}$$

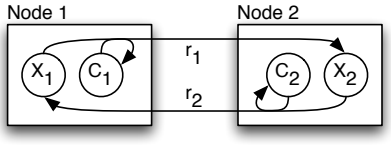

Figure 1: Distributed reaction network spanned by two identical local rules, (2) and (3), resp.

A received molecule is in a first step passively placed into the multiset of the node. For example, rule (3) is not executed immediately after node 1 receives a new

X molecule. It is rather scheduled for a later time determined by an exact stochastic reaction algorithm, such as [11, 12]. The role of this delayed execution is to enforce the "law of mass action" at the macroscopic level. In chemistry, the "law of mass action" states that the reaction rate is proportional to the concentration of reactant molecules: Molecules C_1 and X_1 react with an average rate equal to the product of their concentration $r_1 = c_1 x_1$. The rate of packets sent from node 1 to node 2 is equal to r_1 while the packet stream in the opposite direction exhibits a rate of $r_2 = c_2 x_2$. It can be shown that the overall reaction system spanned by the two local reaction rules strives towards the equilibrium where the number of X molecules in either node is inversely proportional to the number of the corresponding C molecules.

2.2 Representation-free Encoding

Unlike traditional protocols, our chemical networking protocols use a representation-free encoding of information. Traditional protocols store their local state symbolically[1] in variables, such as integers or flags, ultimately encoded as bit patterns. Such symbolic information is then piggybacked to packets in order to send information to distant nodes. CNPs encode protocol states as concentrations, here the abundance of molecule X in the multiset. This encoding is more robust: When removing a molecule, the concentration only changes marginally and state information is not lost, only disturbed.

Due to the law of mass action, the packet *rate* reflects the concentration of its originating chemicals (e.g. $r_1 \propto x_1$). Thus, the packet rate itself, not the symbolic information inside the packet, is used to communicate state information among nodes.[2] This rate coding scheme akin to the nervous system [5] results in a higher resilience to the loss of packets. The law of mass action plays an important role: It mediates between the local and global world by proportionally mapping molecule concentrations to packet rates and vice-versa.

3 Computing Aggregates as Chemical Equilibria

In this section we present an example of a chemical networking protocol (CNP) that calculates the average of distributed values. We compare our chemical approach to epidemic protocols.

3.1 The Gossip-style *Push-Sum* Protocol

Recently, gossip-based or epidemic protocols gained attention because of their potential to disseminate information in a robust way. For example, the *Push-Sum* protocol [16] averages out locally stored values by means of a simple local algorithm: Node i stores sum and weight as tuple (s_i, w_i), starting with $(x_{i0}, 1)$ where x_{i0} is the node's initial (sensor) value. In each round, i.e. after a fixed time interval, each node i first sends the tuple $\left(\frac{1}{2}s_i, \frac{1}{2}w_i\right)$ to a randomly chosen neighbor and to itself and collects the tuples $\{(s_{ir}, w_{ir})\}$ received from neighbors in this round. Then it sums up the received values $s_i := \sum_r s_{ir}$ and $w_i := \sum_r w_{ir}$. Like this, the fraction $x_i = s_i/w_i$ asymptotically approaches the average over all values of the network. Although the protocol is simple, the "proof of the approximation guarantee is non-trivial" [16].

3.2 A Chemical *Disperser* Protocol

In the following we present an alternative, chemical implementation that is elegant and intuitively graspable. It also greatly simplifies the analysis such that the convergence proof keeps on less than half a page. We use the usual network representation as undirected graph $G = (V, E)$ where V is the set of $|V|$ vertices (nodes), E is the set of edges (links). $A = [a_{ik}]$ is the adjacency matrix of the graph where $a_{ik} = a_{ki} = 1$ if $(i, k) \in E$, or $a_{ik} = 0$, otherwise.

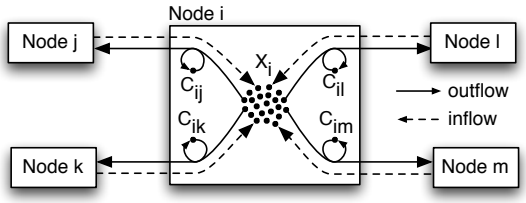

Figure 2: The chemical *Disperser* protocol calculates the average concentration of X molecules in a network of nodes.

Figure 2 schematically depicts the components of the chemical *Disperser* protocol. Each node $i \in V$ contains a multiset of the following molecule types: The concentration of X_i represents the computed average, which is initially set to the local value x_{i0}. For each link $(i, k) \in E$ to i's neighbors there is a single instance of molecule C_{ik} that reacts with the local X_i molecule and, by doing so, sends the X to the corresponding neighbor node k. Formally, such a reaction is represented by the following chemical equation:

$$C_{ik} + X_i \xrightarrow{r_i} C_{ik} + X_k \qquad \forall (i, k) \in E \qquad (4)$$

We can intuitively feel that the global reaction network should lead to equilibrium: The more neighbors a node has, the greater is the outflow of X, since there are more C molecules to react with. On the other hand, a node with higher degree also receives X molecules from more neighbors.

Formal Convergence Proof

To formally prove the stability of the chemical algorithm, we perform a perturbation analysis at the fixpoint. Therefore we first write down the differential equation for the concentration of X_i:

[1] Symbol = meaningful task-related reduction of information [3].

[2] If more than one molecule type is exchanged, however, we have to pass the identity of the molecule along with the packet to distinguish them at the receiver's side.

$$\dot{x_i} = \overbrace{\sum_{k \in V} a_{ki} x_k c_{ki}}^{\text{inflow}} - \overbrace{\sum_{k \in V} a_{ik} x_i c_{ik}}^{\text{outflow}} \qquad \forall i \in V \qquad (5)$$

To find a fixpoint we set $\dot{x_i} = 0$. Since there is only one control molecule C_{ik} per link we simplify (5) by setting $c_{ik} = a_{ik}$. Solving this equation w.r.t. x_i yields

$$x_i = \frac{\sum_{k \in V} a_{ki} x_k}{\deg(i)} \qquad \forall i \in V \qquad (6)$$

Hence, x_i is equal to the average concentration of X in i's neighbors, which only holds iff

$$x_i = \frac{\sum_{k \in V} x_k}{|V|} = \langle x \rangle \qquad \forall i \in V \qquad (7)$$

Consequently, at chemical equilibrium the X molecules are equally distributed over the network.

This equilibrium is stable if the system returns back to the fixpoint after a small perturbation. For the corresponding analysis, we calculate the $|V| \times |V|$ Jacobian matrix of (5):

$$J = [j_{ik}] = \left[\frac{\partial \dot{x}_i}{\partial x_k}\right] = -L(G) \qquad (8)$$

where $L(G)$ is the Laplacian of the network graph

$$L(G) = [l_{ik}] = \begin{cases} \deg(i) & \text{if } i = k, \\ -a_{ik} & \text{otherwise.} \end{cases} \qquad (9)$$

Since any Laplacian has positive real eigenvalues, the eigenvalues of (8) are negative and the fixpoint in (7) is asymptotically stable for arbitrary network topologies.

3.3 Protocol Comparison

The *Push-Sum* as well as our *Disperser* protocol rely on a kind of *mass conservation*. In *Push-Sum*, half of a node's sum s is sent, the remainder is kept, but the overall sum remains constant. For *Disperser*, this conservation is obvious: The number of X molecules is conserved by all reactions. The convergence time is the same for an appropriately chosen parameter set. The two protocols differ in how they asymptotically approach the equilibrium: While *Push-Sum*'s code is executed isochronously, moving half of the value to a neighbor, *Disperser* transfers only one molecule per reaction, this rate being controlled by the inter-reaction time interval, which is inversely proportional to the concentration.

Note that neither of the two protocols is robust against the deletion of messages. However, while a lost message in *Push-Sum* results in the loss of half of a node's value, a packet loss in *Disperser* only decreases the value by one. Robustness is increased, however, at the cost of a higher message complexity. For a variant of the *Disperser* protocol where the nodes do not require information about their neighbors, see [21].

4 Fraglets — A Chemical Programming Language

So far we demonstrated static reaction networks where abstract reaction rules were "installed" permanently in each node. In this section, we extend this model aiming at dynamically changing the set of reaction rules. We present an excerpt of the Fraglets language [1, 25], an artificial chemistry [8], whose corresponding chemical machine is executable and which serves as a simple platform to run chemical protocols.

Each molecule $s \in \mathcal{S}$, or packet, is a string of symbols over a finite alphabet Σ. The first symbol of the string defines the string rewriting operation applied to this molecule by the virtual chemical machine: It can be thought of an assembler instruction. For example, the molecule `[fork a b c d]` transforms itself and splits into the two molecules `[a c d]` and `[b c d]`. The following list shows some of these instructions and their actions:

$$\begin{aligned} [\texttt{match}\ \alpha\ \Phi] + [\alpha\ \Omega] &\longrightarrow [\Phi\ \Omega] \\ [\texttt{fork}\ \alpha\ \beta\ \Omega] &\longrightarrow [\alpha\ \Omega] + [\beta\ \Omega] \\ [\texttt{nop}\ \Omega] &\longrightarrow [\Omega] \\ [\texttt{send}\ k\ \Omega]_i &\longrightarrow [\Omega]_k \quad \text{if } (i,k) \in E \end{aligned}$$

$\alpha, \beta \in \Sigma$ are arbitrary symbols, $\Phi, \Omega \in \Sigma^*$ are symbol strings and $i, k \in V$ are network nodes. Molecules starting with **match** or any non-instruction identifier are in their *normal form*. The **match** instruction can be used to join two molecules by concatenating the second to the first after removing the processed headers. Subsequent instructions immediately reduce the product further until they again reach their normal form. For example, the two molecules `[match pkt send 2 pkt]` and `[pkt data]` in node 1 imply the following reaction:

$$\begin{aligned} &[\texttt{match pkt send 2 pkt}]_1 + [\texttt{pkt data}]_1 \\ &\qquad \longrightarrow [\texttt{send 2 pkt data}]_1 \\ &\text{(transmission to neighbor 2)} \quad \longrightarrow [\texttt{pkt data}]_2 \end{aligned}$$

Such a chemical language allows us to "program" the reaction graph. Molecules now have a structure, i.e. they can *contain* information such as piggybacked user data. However, the dynamics of the reaction network is still governed by the law of mass action and thus, the protocol's behavior is chemically controlled.

5 Self-Healing Code

In order to heal itself, program code must be self-referential and capable of regenerating itself. In addition to these structural and behavioral properties, the code dynamics must be steered by a control loop that recognizes defects and triggers code self-repair. In this section we show how both, structural and dynamical aspects can be realized using chemical programs.

5.1 The "Quine"

In our chemical programming model presented in Sect. 4, there is no distinction between code and data. Molecules may contain code or data or both, and therefore it is possible to modify code or generate it on the fly. An example that illustrates this concept is the following "Quine"[3], a program that generates its own code as output [26]: `[match x fork nop match x]`, `[x fork nop match x]`. These two molecules react and, by doing so, according to the rewriting rules, regenerate themselves as shown in Fig. 3.

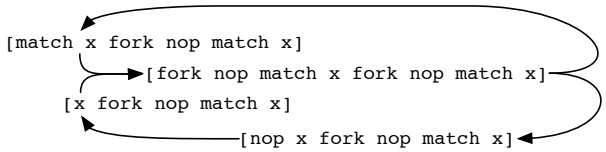

Figure 3: The chemical "Quine" is a set of molecules that replicates itself.

5.2 Code Homeostasis

The self-replicating quine above can easily be converted in one that generates two copies of itself in each round. Due to the law of mass action, its concentration rises exponentially in time. To limit this unbounded growth, we apply a "non-selective dilution flow" to the reactor which randomly destroys a molecule whenever the total number of molecules exceeds a certain threshold. This imposes selective pressure to the reaction vessel; consequently, molecules that are not part of a self-replicating set will eventually be displaced [19].

Interestingly and crucially, this harsh environment, where several quines have to fight for resources, makes them tamper-proof: Even when destroying some of a quines' instances, the surviving population grows again until the vessel capacity is reached. Note that the self-healing property does not require an external observer that monitors the system and plans interventions: The overall system rather intrinsically controls itself to maintain a stable state, that is the *homeostasis* of program code.

6 A Path Load Balancing Protocol

We now make use of quines as building blocks in order to assemble a self-healing protocol that balances a packet stream over two different network paths such that packet loss is minimized.

As depicted in Fig. 4, we inject packets at rate r into the source node where two quines, one for each path, compete for them and send them over the corresponding path. Instead of replicating as fast as possible, the quines wait for and react with acknowledgment packets. These acknowledgments are sent back over the reverse path by the third quine in the destination node that delivers the packet to the application.

[3]after the philosopher and logician Willard van Orman Quine (1908–2000) who studied indirect self-reference

Formal Convergence Proof

This scheme leads to a perfect packet balance among the two paths. When the source node's reaction vessel is saturated, its molecules either belong to quine 1 or 2, as other molecules have been squeezed out. Let's denote the relative concentrations by x_1 and x_2, respectively, satisfying $x_1 + x_2 = 1$. Since replication is triggered by received acknowledgments, these concentrations are $x_1 = r'_1/(r'_1 + r'_2)$ and $x_2 = r'_2/(r'_1 + r'_2)$ where r'_n is the rate of acknowledgments received over path pn.

Let's assume that the bandwidth of *p1* is infinite whereas *p2* drops packets exceeding a rate of B packets/s. We examine the overload situation where the total rate $r > 2B$. Consequently, the rate of acknowledgments is $r'_1 = r_1$ and $r'_2 = \min(r_2, B)$. Due to the law of mass action, the fraction of packets sent over *p1* is proportional to the concentration of quine 1: $r_1 = x_1 r = r_1 r/(r_1 + \min(r_2, B))$. Hence

$$r_1 = r - B \quad \text{and} \quad r_2 = r - r_1 = B \tag{10}$$

Quine 2 reduced its concentration so as to only forward packets up to the bandwidth limitation of path *p2*, as was to be proved.

6.1 Surviving Code Attacks

Like in Sect. 3, our load balancing CNP reaches a (chemical) equilibrium. Deviations from the equilibrium, for example by lost molecules on a network path, are compensated by increasing the population of quines that forward packets over the opposite path. Moreover, the code itself is organized in circuits of self-replicating molecules: If we forcefully destroy some code, the system will eventually regenerate it and, after some transition time, autonomously finds back to the equilibrium state. In fact, packet loss as well as code loss are treated by the same mechanism and in the same way.

Figure 5 depicts the protocol's response to such a deletion attack[4], simulated in OMNeT++ for $r = 200\,\mathrm{pkts/s}$ and $B = 40\,\mathrm{pkts/s}$. At $t = 50\,\mathrm{s}$ we removed 80 % of all molecules (code and data) from the source vessel. Note that the code regenerates itself within 10 s while the traffic distribution r_1/r_2 remains unchanged.

6.2 Discussion

Being a showcase, our protocol is not as elaborate as existing load-aware protocols in many respects. Emerging from microscopic chemical reactions, it realizes an *intrinsic* bandwidth estimation by using the rate of actual data packets akin to ACK pacing in TCP [2] whereas other existing bandwidth estimation techniques numerically calculate the bandwidth based on the rate of separate probe packets or on their inter-arrival time [10].

[4]Code mutations can also be caught by mapping them onto code deletion events using a simple instruction encoding guarded by a parity-bit.

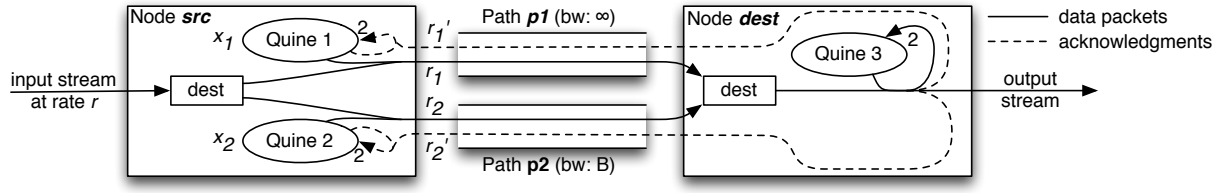

Figure 4: Data packets to node *dest* are injected at rate r into node *src*. Quines 1 and 2 (concentration x_1 and x_2, resp.) compete for and forward packets at rate r_1 and r_2 over paths *p1* and *p2*, resp. The quine's replication is controlled by acknowledgments received at rate r_1' and r_2', resp.

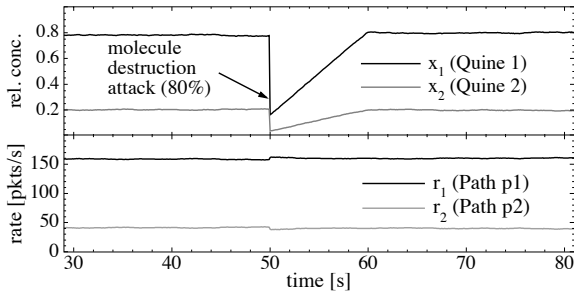

Figure 5: OMNeT++ simulation: Relative concentrations of the self-healing quines (forwarding code) in *src* during a deletion attack and the corresponding packet forwarding rate (unaffected) of the quines.

Our protocol converges for bursty traffic: The stochastic algorithm schedules the next reaction based on an exponential probability distribution; consequently, a burst of incoming packets appears blurred at the output, which helps estimating the bandwidth.

The load balancing protocol discussed in this paper may also be used for other tasks. In [20], it serves as a self-adapting forwarding engine of a self-healing routing protocol implementation.

7 Relevance, Impact, Future Work

In this section we will discuss some of the consequences of a chemical protocol mindset.

7.1 Determinism vs. Resilience

Although ODEs can be used to approximate the behavior of CNPs (as shown in Sects. 3 and 6) the underlying execution model is stochastic. Consequently, pure CNPs cannot be used for time-critical protocols. Additionally, CNPs cannot prevent packet reordering. However, the deterministic character of today's protocols is not needed everywhere in networking; there are network services where we can relax the level of certainty for the purpose of obtaining more resilience. This applies specifically to continuously running services like routing, load balancing and long lasting signaling streams. As we pointed out in Sect. 3, collective computations at large are also good candidates. The use of representation-free encoding mitigates the packet reordering problem since it does not matter which of the identical molecule instances is sent next.

Generally, we should try to soften the precision of the protocol's specification. As is conjectured in [24], this may avoid brittleness of the system and facilitates a CNP implementation. In return, we gain resilience to various perturbations, such as the partial loss of information in form of packets or code.

7.2 System Design and Analysis

Traditional flow-based protocols are analyzed as being composed of a fixed number of interconnected packet queues, scheduled by *deterministic* algorithms [17]. In CNPs we can identify similar entities, which however interact differently: Each node has a single packet buffer (the molecule multiset) and we impose a specific *stochastic* scheduling that is compatible with the law of mass action [11, 12]. We can regard the multiset as virtually divided into individual queues, one for each molecular type, that are interconnected by reaction rules. The rules may compete for the same type, akin to rule-based protocols [9, 18], and form a (distributed) closed network in the sense of Kelly [15].

While traditionally only the packet flows have been modeled as stochastic processes, we also apply (and impose) this to the packet processing. On the analysis side, this does not change much: the same mathematical framework can be applied. However, our environment forces protocol designers to come up with "fluid" protocols that can continuously track the environment and adapt to it. This might seem to be a challenging task. But, in addition to a sound mathematical foundation, a CNP approach potentially promotes good system-wide properties when composing such protocols. CNPs also provide a graphical notation, the reaction graph, which is intuitively graspable, and from which the underlying mathematical equations can be derived automatically. This is one of the biggest advantages of CNPs over traditional protocols. We conjecture that this may simplify protocol and system design as well as the analysis of the corresponding dynamics.

7.3 Resource Exploitation vs. Embodiment

The shown CNP examples have a higher message complexity and tend to exploit all available resources, leading to a competitive rather than a cooperative environment. A contribution to last year's HotNets workshop promoted this strategy as long as the marginal cost can be driven lower than the marginal benefit [23]. This trade-off, however, has to be identified and analyzed for each problem separately.

While exploiting resources, CNPs at the same time explore and dynamically adapt to the environment. Our

load balancing protocol is able to obtain bandwidth information, like TCP, *because* it fills the link with packets; the quines occupy the available memory but survive when reducing it. Hence, CNPs achieve a high degree of embodiment: computation may be outsourced to the environment, a concept already employed in robotics [22].

7.4 Towards Self-Optimizing Protocols

In Computer Science, bio-inspired approaches recently gained attention because of the promise of robustness and scalability. The ultimate property however, evolvability, is a challenge due to the lack of a formal theory and the non-determinism of the resulting solutions. With a chemical protocol design, a gradual path from static to self-healing, and even towards self-optimizing and evolving protocols, could be envisioned. Mutations could be useful to evolve protocols online, potentially enabling self-optimization and long-term adaptability. We conjecture that CNPs are better suited to automatic evolution than traditional protocols due to their inherent property of multi-stability that lets them glide into the next equilibrium according to environmental conditions.

8 Conclusions

In this paper we looked at networking protocols with the eye of a chemist. We examined how chemical concepts can be transposed to network design in order to obtain the same emergent properties that we find in chemical systems such as stable equilibria, nota bene gaining analyzability. This requires a paradigmatic shift from designing local state machines to weaving global reaction networks. For the first time, we demonstrated how to obtain a protocol that intrinsically heals itself from code deletion attacks. We believe that such techniques open the door to the design of truly robust networks which are able to survive even in environments where reliable code execution can not be taken for granted.

Acknowledgments

This work has been supported by the Swiss National Science Foundation through SNF Project Self-Healing Protocols (2000201-109563). We would like to thank Lidia Yamamoto for her continuous contribution to our research, and Christophe Jelger, Pierre Imai and the anonymous reviewers for their useful comments.

References

[1] *Fraglets Home Page*. `http://www.fraglets.net`.

[2] A. Aggarwal, S. Savage, and T. Anderson, *Understanding the Performance of TCP Pacing*, Proc. 9th Ann. Joint Conf. of the IEEE Comp. and Comm. Societies (INFOCOM 2000), 2000, pp. 1157–1165.

[3] R. Bajcsy and J. Košecká, *The Problem of Signal and Symbol Integration*, Advances in Artificial Intelligence, 1995.

[4] L. N. Chakrapani, P. Korkmaz, B. E. S. Akgul, and K. V. Palem, *Probabilistic System-on-a-Chip Architectures*, ACM Trans. Des. Autom. Electron. Syst. **12** (2007), no. 3, 1–28.

[5] P. Dayan and L. F. Abbott, *Theoretical Neuroscience*, MIT Press, 2001.

[6] G. Di Caro and M. Dorigo, *AntNet: Distributed Stigmergetic Control for Communications Networks*, J. Art. Intel. Res. **9** (1998), 317–365.

[7] P. Dittrich and P. Speroni di Fenizio, *Chemical Organization Theory*, Bull. Math. Bio. **69** (2007), no. 4, 1199–1231.

[8] P. Dittrich, J. Ziegler, and W. Banzhaf, *Artificial Chemistries - A Review*, Artificial Life **7** (2001), no. 3, 225–275.

[9] F. Dressler, I. Dietrich, R. German, and B. Krüger, *A Rule-Based System for Programming Self-Organized Sensor and Actor Networks*, Comput. Netw. **53** (2009), no. 10, 1737–1750.

[10] Y. Easwaran and M. A. Labrador, *Evaluation and Application of Available Bandwidth Estimation Techniques to Improve TCP Performance*, Proc. 29th Annual IEEE Int. Conf. Local Comp. Net., 2004, pp. 268–275.

[11] M. A. Gibson and J. Bruck, *Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels*, J. Phys. Chem. A **104** (2000), no. 9, 1876–1889.

[12] D. T. Gillespie, *Exact Stochastic Simulation of Coupled Chemical Reactions*, J. Phys. Chem. **81** (1977), no. 25, 2340–2361.

[13] J. H. S. Hofmeyr, *Metabolic Control Analysis in a Nutshell*, Proc. 2nd Int. Conf. Sys. Bio., 2001, pp. 291–300.

[14] V. Jacobson, *Congestion Avoidance and Control*, Proc. Symp. Comm. Arch. Prot., 1988, pp. 314–329.

[15] F. P. Kelly, *Reversibility and Stochastic Networks*, John Wiley and Sons Ltd., 1979.

[16] D. Kempe, A. Dobra, and J. Gehrke, *Gossip-based Computation of Aggregate Information*, Proc. 44th IEEE Symp. Found. Comp. Sc., 2003, pp. 482–491.

[17] J.-Y. Le Boudec and P. Thiran, *Network Calculus*, LNCS, vol. 2050, Springer, 2004.

[18] L. F. Mackert and I. B. Neumeier-Mackert, *Communicating Rule Systems*, Proc. IFIP WG6.1 7th Int. Conf. Prot. Spec., Test. and Verific., 1987, pp. 77–88.

[19] T. Meyer, D. Schreckling, C. Tschudin, and L. Yamamoto, *Robustness to Code and Data Deletion in Autocatalytic Quines*, Trans. on Comp. Sys. Bio. X, 2008, pp. 20–40.

[20] T. Meyer, L. Yamamoto, and C. Tschudin, *A Self-Healing Multipath Routing Protocol*, Proc. 3rd Int. Conf. on Bio-Insp. Models of Netw., Inform., and Comp. Sys. (BIONETICS 2008), 2008.

[21] T. Meyer, L. Yamamoto, and C. Tschudin, *An Artificial Chemistry for Networking*, Bio-Inspired Computing and Communication, 2008, pp. 45–57.

[22] R. Pfeifer, M. Lungarella, and F. Iida, *Self-Organization, Embodiment, and Biologically Inspired Robotics*, Science **317** (2007), no. 5853, 1088–1093.

[23] R. Raghavendra, R. Mahajan, J. Padhye, and B. Zill, *Eat All You Can in an All-you-can-eat Buffet: A Case for Aggressive Resource Usage*, Proc. 7th ACM Workshop on Hot Topics in Networks (Hotnets VII), 2008.

[24] M. Shaw, *"Self-healing": Softening Precision to Avoid Brittleness*, Proc. 1st Workshop on Self-healing Systems, 2002, pp. 111–114.

[25] C. Tschudin, *Fraglets - a Metabolistic Execution Model for Communication Protocols*, Proc. 2nd Symp. on Aut. Intel. Net. and Sys. (AINS), 2003.

[26] L. Yamamoto, D. Schreckling, and T. Meyer, *Self-Replicating and Self-Modifying Programs in Fraglets*, Proc. 2nd Int. Conf. on Bio-Insp. Models of Netw., Inform., and Comp. Sys. (BIONETICS 2007), 2007.