An Artificial Chemistry for Networking

An Artificial Chemistry for Networking

Computer Science Department, University of Basel
Bernoullistrasse 16, CH–4056 Basel, Switzerland
{th.meyer,lidia.yamamoto,christian.tschudin}@unibas.ch

Abstract. Chemical computing models have been proposed since the 1980ies for expressing concurrent computations in elegant ways for shared memory systems. In this paper we look at the distributed case of network protocol execution for which we developed an online artificial chemistry. In this chemistry, data packets become molecules which can interact with each other, yielding computation networks comparable to biological metabolisms. Using this execution support, we show how to compute an average over arbitrary networking topologies and relate it to traditional forms of implementing load balancing. Our long-term interest lies in the robust implementation, operation and evolution of network protocols, for which artificial chemistries provide a promising basis.

Keywords: artificial chemistry, network protocols, distributed algorithms, load balancing, Fraglets.

1 Introduction

Chemical computing models have been proposed since the 1980ies for expressing concurrent computations in a natural way [1–5]. In parallel, artificial chemistries [6–8] were constructed to model chemical phenomena related to life and its origins. Some of these chemistries express and evolve computer programs [7, 9], potentially representing new models of computation. However the vast majority of these artificial chemistries have remained at the level of simulations, where the actual pace of the chemical system with respect to the real time and among different devices is not an issue.

In this contribution we examine the potential of such chemically-inspired computation models for networking. On one hand, we extend our Fraglets language [10] to a full artificial chemistry setting. On the other hand, we extend the artificial chemistry concept to a distributed system, in two steps: The first step is to extend it to an online environment where time synchronization becomes essential for responding to external events with the expected impact. The second step is to take the network topology into account, extending the centralized analysis methods to a distributed system.

Finally, we show a case study on a distributed equilibrium algorithm that is able to find the equilibrium concentrations of a target molecule among a set of nodes in an arbitrary topology. The algorithm is applied to a load balancing problem in which tasks should be equally distributed among all nodes.

pre-print of paper appeared in:
Bio-Inspired Computing and Communication, First Workshop on Bio-Inspired Design of Networks (BIOWIRE 2007), Cambridge, UK, April 2–5, 2007.
In LNCS vol. 5151, Revised Papers, Lio, P.; Yoneki, E.; Crowcroft, J.; Verma, D.C. (Eds.), 2008, pp. 45–57.

Many sophisticated approaches to load balancing exist [11–15]. A classification can be found in [12]. More recently, a chemotaxis-inspired load balancing approach was proposed [15]. Although also chemically-inspired, the approach in [15] did not rely on a general purpose artificial chemistry. In our system, in contrast, the load balancing algorithm implicitly emerges as an effect of the chemical reaction network that is constructed and distributed among nodes. We do not claim that the resulting algorithm is superior to the current state of the art in load balancing. However, it offers a different perspective upon problem solving in distributed systems: instead of pre-programming the system numerically to achieve a desired stable state, an equilibrium can be reached autonomously via the exchange of virtual molecules among nodes.

This paper is structured as follows: Section 2 provides some basic background information on artificial chemistries and chemical computing, that will be needed for the rest of the paper. Section 3 briefly describes the Fraglets language and the instructions used in the load balancing case study, which is then presented in Section 4. Section 3 also defines Fraglets as an artificial chemistry, then proposes extensions of artificial chemistries to online and distributed systems. The equilibrium study in Section 4 also relies on these definitions and extensions.

2 Artificial Chemistries and Chemical Computing

Chemical computing [4, 5, 8] includes real (wet computation with real molecules) and artificial models inspired by chemistry but executing top of conventional computer architectures. This paper focuses on the latter only, and their extension to networked environments.

In [8] chemical computing models are classified within *Artificial Chemistry*, the subfield of Artificial Life devoted to the dynamics of chemical phenomena related to life and organizations in general.

The term *artificial chemistry* also refers to the specific chemical model used. In this sense, an artificial chemistry [8] is defined by a triple (S, R, A), where S is the set of molecules, R is the set of reaction rules, and A is an algorithm that determines how the rules are applied to the molecules. For example, in [7] the set S contains expressions from λ-calculus, the set R contains conditions under which two molecules from S may react and the way reactions take place: the reactants remain in the reactor, the corresponding products are inserted, and two other molecules are chosen at random for decay. The algorithm A just picks two molecules at random and performs the reaction or not depending on the conditions in R. Such a simplified chemistry can nevertheless show the spontaneous emergence of self-sustaining organizations out of an initial “soup” of random molecules. However, the algorithm A becomes computationally expensive if the probability of two random molecules reacting with each other is small.

For simulating real chemistries in an efficient way, variants of the Gillespie algorithm [16] are widespread. This algorithm simulates the stochastic dynamics of a real-world well-stirred chemical reactor tank. For each time step iteration, it calculates: *(i)* the next reaction to occur, taking into account for each reaction rule, the collision probabilities of their reactants; *(ii)* the virtual time τ when the chosen reaction is expected to occur. The complexity of this algorithm is $\mathcal{O}(|R|)$ for each iteration step: it is of course more complex than just choosing two molecules at random at every iteration step, but on the other hand, it only selects those molecules that do react in fact, and does not spend cycles on inert molecules. Therefore it is more efficient when only a small subset of those molecules in the reactor may actually react.

3 Organizing Interacting Packets as Chemical Reactions

In computer networking, the most frequently executed action on data packets is the rewriting of header fields. For example, on each leg of a packet's route through a sequence of Ethernets, the packet must obtain a new destination field to reach the next hop. Fraglets are a special form of data packets, based on the same principle: By rewriting a packet's header fields, we can implement distributed computations like communication protocols or a load balancing algorithm.

In this section, we first describe a communication environment called Fraglets before in Section 3.2 we show how Fraglets form an artificial chemistry. Relating artificial time with real time and by interconnecting the artificial chemical reactors, we obtain a distributed artificial chemistry that is able to implement network functionality in ways beyond classic forwarding tasks, as we show in Sections 3.3 and 3.4, respectively.

3.1 Fraglets

Formally, a string rewriting system is a pair (Σ, P) where Σ is a finite alphabet of symbols and P is a set of production rules. A production rule is string substitution pattern that operate on words $w \in \Sigma^*$. In Fraglets, we limit ourselves to substitution patterns which, on their left side, only depend on the first symbol of a word. For example, the rule

`[exch S T U TAIL]` $\rightarrow$ `[S U T TAIL]`

when applied to the word `[exch a b c d]` will result in `[a c b d]` – that is, two symbols are swapped. The `exch` acted as a prefix command for the rest of the word whereas the new left-most symbol '`a`' serves as a continuation pointer for further processing of the result.

This type of string rewriting systems, where the left most symbol identifies the rule to apply, is also called a tag system. Unlike Post's original tag system [17], which operates on one initial word and asks about the system's expansion, we place ourselves in a multiset context where the production rules are applied to all words in a multiset. In Fraglets, we interconnect several multisets such that they form a network of packet processing nodes. Thus a Fraglets system is a tuple (Σ, P, N, E) where N is a set of nodes $n_1, \ldots, n_k$, each containing a multiset of words over Σ to be transformed according to the rules P, and the nodes being interconnected according to edges $(n_i, n_j) \in E$. An excerpt from the set of production rules for Fraglets is shown in Table 1.

Op	*input*		*output*
exch	`[exch S T U TAIL]`	$\rightarrow$	`[S U T TAIL]`
node	$_{n_i}$`[node S]`	$\rightarrow$	`[S ni]` *(get node's name)*
send	$_{n_i}$`[send` $\mathtt{n}_j$ `TAIL]`	$\rightarrow$	$_{n_j}$`[TAIL]` *(if $(n_i, n_j) \in E$, ϵ otherwise)*
	$_{n_i}$`[send any TAIL]`	$\rightarrow$	$_{n_j}$`[TAIL]` *($\exists j : (n_i, n_j) \in E$; anycast)*
	$_{n_i}$`[send all TAIL]`	$\rightarrow$	$_{n_j}$`[TAIL]` *($\forall j : (n_i, n_j) \in E$; broadcast)*
split	`[split PART1 * PART2]`	$\rightarrow$	`[PART1], [PART2]`
sum	`[sum S` $\mathtt{i}_1$ $\mathtt{i}_2$ `TAIL]`	$\rightarrow$	`[S` i_1+i_2 `TAIL]` *(do. for mult etc)*
match	`[match S TAIL1],[S TAIL2]`	$\rightarrow$	`[TAIL1 TAIL2]`
matchp	`[matchp S TAIL1],[S TAIL2]`	$\rightarrow$	`[matchp S TAIL1], [TAIL1 TAIL2]`
mmatchp	`[mmatchp` n $\mathtt{S}_1$ ... $\mathtt{S}_n$ $\mathtt{TAIL}_0$`],` `[`$\mathtt{S}_1$ $\mathtt{TAIL}_1$`],...,[`$\mathtt{S}_n$ $\mathtt{TAIL}_n$`]`	$\rightarrow$	`[mmatchp` n $\mathtt{S}_1$ ... $\mathtt{S}_n$ $\mathtt{TAIL}_0$`],` `[`$\mathtt{TAIL}_0$ $\mathtt{TAIL}_1$ ... $\mathtt{TAIL}_n$`]`

Table 1. Selected production rules of a Fraglets system. `S`, `T` and `U` are placeholders for symbols $\in \Sigma$, `TAIL` stands for a potentially empty word $w \in \Sigma^*$.

As an example, the fraglet `[split a b * c d e]` will result in two fraglets `[a b]` and `[c d e]`. The `match` rule lets two fraglets react together which share a common symbol at the second and first position, respectively:

```
[match a b c],[a x y z] → [b c x y z]
```

and the result is the concatenation of the two tails. The special persistent form of `match` is called `matchp` and permits to define "catalytic" processing rules that are not consumed during their reaction.

As a final example the following execution trace shows how the `send` tag can be used to implement traditional packet forwarding:

```
[send n2 send n3 send n4 my payload] sender's fraglet executes at n1
[send n3 send n4 my payload]         at n2
[send n4 my payload]                 at n3
[my payload]                         arrived at n4
```

This example demonstrates source routing where the sending node n_1 lets a packet work itself through a chain of nodes $n_2 \dots n_4$.

3.2 Fraglets as an Artificial Chemistry

In accordance with the definition in [8] explained in Section 2, an artificial chemistry is characterized by the triple (S, R, A), which we now define for Fraglets in the following way:

The set of molecule species S corresponds to the set of all possible fraglets, i.e. all words $w \in \Sigma^*$. Thus $S \equiv \Sigma^*$. Similarly, the set of reaction rules R is equivalent to the set of production rules P: $R \equiv P$.

The algorithm A, which selects which molecules to process at each round, only takes the molecules' matching heads into account. This leads to a two-level hierarchy: the actual molecular species correspond to fraglets which are strings $w \in \Sigma^*$ of arbitrary length, each of which may occur several times in the multiset of a given node $n \in N$. The reactor algorithm only looks at their headers, defining the second level of hierarchy where all the molecules with the same matching head symbol are considered as the same reactant for the choice of the next reaction to perform. This makes the algorithm scalable in spite of a potentially large number of different fraglets.

The original reaction algorithm in Fraglets did not take into account the full dynamics of molecule concentrations as in [7, 16]. This restricted its applicability to cases where linear dynamics would suffice. We have now extended the Fraglets interpreter with variants of the Gillespie algorithm [16] such that more complex dynamics can be expressed. This is essential for an analytically tractable control of molecule concentrations. The header matching scheme is preserved in any case, since the algorithm only operates at the second level of hierarchy, which inspects only the fraglet headers.

3.3 Online Artificial Chemistry

In this section we introduce the concept of an *online artificial chemistry* for an artificial chemistry that is embedded in a real world environment in which it has to react in a timely manner.

In chemistry, the amount of reactions that may happen in parallel is only limited by the amount of molecules present and their collisions. For instance, autocatalytic reactions may lead to exponential growth in substrate concentration. From an information processing perspective, this is equivalent to an exponential growth in processing capacity. This powerful property is largely exploited when computing with real molecules such as DNA computing.

Algorithms such as Gillespie's [16] emulate chemistries on top of classical von Neumann computers with fixed processing capacity. The elastic processing capacity of chemical systems is emulated with the help of a virtual time which is inversely proportional to the total sum of the products of concentrations of all potential reactants. Virtual time steps can be made arbitrarily small as the reactant concentrations grow. If such virtual time is used for making decisions such as in a robot or network, then mapping it to physical time in a coherent way is mandatory for the device to present consistent reaction times.

It follows that an online artificial chemical system has to map the calculated virtual time (τ) which would have elapsed until the next reaction in a real tank reactor, to a physical time (τ'), the time that will actually elapse. A simple algorithm is to assume a one-to-one mapping $\tau = \tau'$, and just sleep for $\tau - T$, where T is an estimation of the real time it takes to process the reaction in practice. This simple algorithm obviously requires $\tau \geqslant T$, meaning that there is an upper bound of molecules, above which the algorithm is unable to keep the pace of a real chemistry. On the other hand, there is also a lower bound below which the sleeping time becomes too large for the system to react to the real world and to the inflow of new molecules. Finding this compromise is crucial to have artificial chemistries running online, and deserves further research.

3.4 Distributed Artificial Chemistry

In order to use an online artificial chemistry in computer networks, we must expand it into a model of distributed reaction systems. In this section we show the challenges of the distributed case, and that the resulting distributed artificial chemistry can analytically be treated like a local one.

First we define a network of artificial chemical reactors (nodes). The network topology, which interconnects the reaction vessels, describes a high-level structure, conceptually one layer above the reaction system inside a certain node. A single node still reflects a well-stirred reactor that does not consider spatial neighborhood of molecules.

In this sense, a network of reactors can be described as a undirected graph $G = \{N, E\}$, where $N = \{n_1, \dots, n_k\}$ is the set of all nodes in the network. Each node $n_i \in N$ is an independent reactor, driven by an individual CPU. The edges $E = \{e_1, \dots, e_l\}$ are bidirectional network links and connect neighbor nodes. Two nodes n_i and n_j are neighbors iff $\exists e = (n_i, n_j) \in E$. In this case we define $\mathrm{adj}(i,j) = \mathrm{adj}(j,i) = 1$, otherwise $\mathrm{adj}(i,j) = \mathrm{adj}(j,i) = 0$. Each node is able to emit molecules, which are sent along the path of a link to one of the neighbor nodes using unicast, broadcast, or anycast transmission primitives.

Since each node is simulated independently, a single node n_i is an individual artificial chemical reactor with its own (S_i, R_i, A_i), i.e. each node is defined by an individual set of molecules S_i, reaction rules R_i, and algorithm A_i. We write W_i for a molecule $W \in S_i$. The exchange of molecules between two nodes n_i and n_j can be treated like reactions that map the set of molecules S_i to S_j, for example $W_i \to W_j$. Thus, in addition to a local reaction network, each node also contains reaction rules that send molecules to other nodes. The overall reaction network is then defined by (S, R, A), where $S = \bigcup_{i \in N} S_i$ and $R = \bigcup_{i \in N} R_i$.

Consider the following network $G = \{N, E\}$, where $N = \{n_1, n_2\}$, and $E = \{(n_1, n_2)\}$. The reaction system over the overall set of molecules $S = \{W_i, W_j\}$, depicted in Figure 1, achieves a balance in concentration of molecule W between the two nodes.

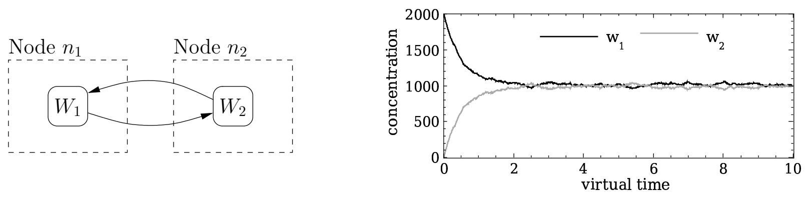

Fig. 1. Equilibrium reaction for a two node network topology

The corresponding program in Fraglets uses unicast transmission to let each node send one molecule W at the time to the other node.

```
in node n1: [ matchp W send n2 W ]
            [ W ]2000 (i.e., 2000 copies of [W])
in node n2: [ matchp W send n1 W ]
```

Even when starting with an unequal distribution of molecules W, the reaction system drifts into a state where both nodes contain the same amount of W. The stochastic simulation of this distributed reaction system is shown in the right side of Figure 1.

There is one essential requirement for the algorithm A_i of a distributed reaction system: The virtual time evolution in all nodes participating the network must be the same. Generally, when a node sends molecules to another node, the sender generates a molecule stream of a certain rate with respect to the virtual time of its simulation algorithm. This rate is proportional to the concentration of the molecule. The molecule to be sent is then encoded as payload of a network packet, and is sent to the destination node, which is usually driven by another CPU. There the payload is converted back to a molecule, and is injected into the destination reaction vessel, which simulates its own virtual time evolution.

Therefore if the nodes of a network are allowed to be driven by different CPUs, it is important that the virtual time evolution in all nodes of the network is the same. Otherwise, a molecule stream may be received with an other rate than originally generated. If such a "synchronization" is achieved, for example by locally synchronizing the virtual time of each node to the physical time, then, the resulting distributed reaction system that is spawned across all nodes can analytically be treated like a local reaction system. This includes methods to analyze the topology of the network, stoichiometric analysis [18], metabolic control analysis [19], as well as results from the chemical organization theory [20].

4 A Chemical Protocol for Load Balancing

In this section we introduce a novel approach to balance work load in a network that exploits the dynamics of molecule reactions in an artificial chemistry. To this end we install flows of "job molecules" that seek to level out different concentration of job molecules on each node. The salient point is that load differences never need to be computed explicitly, as differences in packet rates are sufficient to steer the system into an equilibrium state.

Figure 2 depicts a typical network topology with columns representing the amount of jobs before and after load balancing. The left side shows the situation right after injecting a given amount of job molecules W into node n_1, whereas the right side delineates the job molecule distribution after the distributed reaction system reached its steady state.

We assume that all jobs are either independent and self-contained, or that we have job tokens which a node uses to request the actual job, that is: our system does not have to maintain a queueing policy for job molecules.

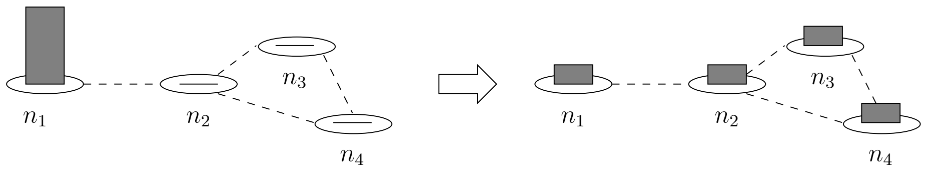

Fig. 2. Load balancing for a four node network topology. The columns represent the concentration of job molecules W in each node.

4.1 Distributed Equilibrium Algorithm

The basic idea of our algorithm is to *(i)* publish the local concentration of job molecules W to all neighbor nodes. Every node then *(ii)* requests job molecules W from overloaded neighbors in a stochastic manner. *(iii)* This promotes another reaction that carries job molecules from heavily to lightly loaded nodes.

We introduce three different families of molecule species: W_i molecules represent the work load, S_i are signalling molecules whose rates indicate to neighbors the current load level, while R_i molecules are requests from a node to obtain more work. Our algorithm can be expressed formally by the following abstract reaction system, where $X_i^{(j)}$ denotes molecule X, currently residing in node i, originally created and sent by node j.

$$\begin{aligned} W_i &\longrightarrow W_i + \sum_{j \in N} \left(\mathrm{adj}(i,j) \cdot S_j^{(i)} \right) && (i) \\ S_i^{(j)} &\longrightarrow R_j^{(i)} && (ii) \\ R_i^{(j)} + W_i &\longrightarrow W_j && (iii) \end{aligned}$$

The numbering of the reactions corresponds to our list of principles introduced above. For example, reaction (ii) states that a received signaling molecule will be converted into a request molecule that is sent back to the originator of the signaling molecule.

The three reactions have a direct translation into the Fraglets language as shown below. Note that each of these three Fraglets rules must be present in all nodes of the network.

```
(i)   [ matchp W split W * split node N * match N send all S ]
(ii)  [ matchp S split node RCMD1 *
        split match RCMD1 exch RCMD2 W *
        split match RCMD2 RCMD R *
        match RCMD send ]
(iii) [ mmatchp 2 R W send ]
```

Before we explain in more details how these reactions work together, we introduce a graphical representation of the reaction network for a three-node

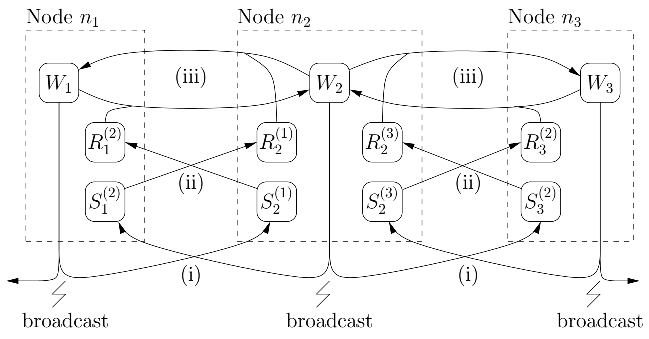

Fig. 3. Distributed reaction network

string topology. As can be seen in Figure 3, S molecules are deterministically transferred to a neighbor node and become a R molecule. Such a R molecule will stochastically bind to a W job molecule and drag it back to where the S molecule came from. The following paragraphs explain each of the three reactions in more detail:

(i) Concentration Tracking: Each node must be informed about the load of its neighbors. Therefore the first reaction's task is to broadcast signaling molecules S to all peer nodes. A signaling molecule contains the name of the source node, and is emitted with a rate that is proportional to the concentration of job molecules W. The opposite view is that each node receives a signal stream from its neighbors. The resulting concentration of signaling molecules S reflects the concentration of job molecules in the neighbor nodes.

(ii) Job Molecule Request: The second reaction stochastically picks and consumes one of the signaling molecules S_i. Since the Fraglets `matchp` reaction matches only the head symbol, this rule applies for all molecules $S_i^{(j)}$ received from any neighbor node j. This results in picking the signaling molecules of heavy loaded peer nodes more frequently, because the concentration of the signaling molecule reflects the concentration of job molecules W in the peer node, and due to the stochastic selection process. After picking a signaling molecule $S_i^{(j)}$, this reaction builds a request molecule $R^{(i)}$, tags it with the name of the local node, and sends it back to the node j, the originator of the signaling molecule. Request molecules are sent using unicast messages.

(iii) Job Molecule Transport: Request molecules promote the third reaction, that transmits a job molecule W_i to the neighbor node j that requested it.

The overall result of this reaction network is that if a node has a higher concentration of W molecules than any of its neighbors, it will emit more S molecules than the neighbors, thus be drained more aggressively through R molecules.

4.2 Results

The proposed three-stage mechanism regulates the exchange of job molecules W. Figure 4 shows the dynamic behavior of the concentration of molecule W in a four node network as depicted in Figure 2. This result was obtained with a Fraglets interpreter that simulated the behavior of the four nodes. We injected 4000 job molecules W at time $t = 10$ into node n_1. We ran the simulation of the above system such that all nodes are operating in non-saturated mode and that the network links are not afflicted with delay or packet loss.

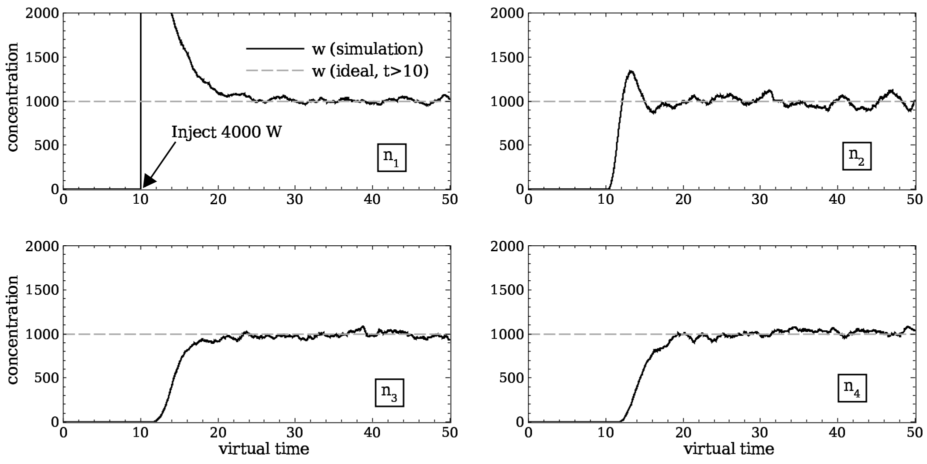

Fig. 4. Equilibrium of job molecules W for a four node topology without delay and packet loss. At time $t = 10$, a quantity of 4000 molecules of W is injected to node n_1.

The concentration of molecule W converges to the expected concentration of 1000 molecules in all nodes of the network. At $t = 13$ one can nicely see the onset of W molecules in node n_2: the rise of concentration is soon capped by nodes n_3 and n_4 which start to drain node n_2 from excess job molecules. The system quickly reaches the steady state where the concentrations fluctuate around an average value due to the stochastic notion of the reaction algorithm.

An important aspect of our approach is that we can verify the properties of chemical protocols by formal methods of flux base analysis [18]. In the case of the load balancing protocol, we studied the steady state of the system where the molecule concentrations do not change anymore. Solving the resulting equations for an arbitrary node shows that the concentration of W is equal to the average concentration of W in its neighbors. From this it follows that the concentration of job molecules must be equal in all nodes of the network for an arbitrary topology.

4.3 Discussion

In this paper we showed another, chemical way of looking at the problem of load balancing. In this section we first classify our algorithm and compare it to a similar chemotaxis-inspired method. Then we show why our initial approach, an even simpler, diffusion-based reaction network was not successful for an arbitrary network topology. Finally, we discuss the impact of imprecise virtual to physical time synchronization and network links with delay and packet loss.

Related work. The chemical algorithm proposed in this paper can be classified according to [12] as a distributed dynamic load balancing algorithm: distributed, because the algorithm does not rely on central knowledge of the job distribution, and dynamic, because jobs may be dynamically produced and consumed during operation. Our algorithm is similar to the chemotaxis-inspired load balancing algorithm proposed in [15], which lets fast signals diffuse into the network. These signals then act as attractors to move jobs from overloaded nodes to nodes with available capacity. Similarly, in our algorithm, every node broadcasts signaling molecules to its neighbors. The role of signaling molecules is to promote the transfer of jobs. Unlike [15] we fully rely on the stochastic selection of reactions and molecules. For example, our algorithm never explicitly calculates the exact arithmetic difference of job molecules between nodes. Instead, the Gillespie algorithm more frequently picks those molecules that manifest in higher concentrations: The artificial chemical reaction vessel intrinsically balances the execution probabilities of interdependent reactions, and thus the resulting balance is an emergent property of the distributed reaction network.

Load balancing with less than three reaction types? The initial idea was to let job molecules immediately diffuse to the neighbors using anycast transmission. We hoped that a single reaction, which stochastically picks and alternatively sends a job molecule to a neighbor, would already yield a balance of work load. However, a formal analysis of the resulting reaction network, which strongly resembles the diffusion mechanism in physics, showed that the equilibrium can only be maintained for a fully meshed topology. Already a simple chain topology as in Figure 3 results in an imbalance. At the end, we came up with the presented algorithm that obtains a work load balance for any network topology.

Real world considerations. So far we have assumed that a CPU that executes the artificial chemical reactor is infinitely fast, that its clock is precise, and that the network is not afflicted with delay or packet loss. In the remaining paragraphs, we show what happens if we relax these constraints.

In reality, CPU clocks are subject to jitter and drift which affects the virtual to physical time mapping suggested in Section 3.3. Stochastic jitter is not harmful for chemical algorithms since they do not rely on deterministic execution of reactions. In contrast, clock drift between distributed reaction vessels lead to shifted reaction weights. When dilating the virtual time on a certain node the rate of incoming molecules increases while the rate of outgoing molecule decreases with respect to the virtual time. In this case the proposed algorithm establishes a distribution of job molecules proportional to the virtual time dilatation of the participating nodes. For example, if there is one node in which a reaction takes twice as long as in the other nodes, that node contains twice as much job molecules as the other nodes in steady state. Therefore one must assert that a node's load does not affect the speed of the artificial chemistry reactor, as otherwise our protocol will not achieve the desired result.

In addition to imperfect time mapping, in a real network the algorithm has to cope with packet loss and delay. In case of overloaded links, some of the signaling molecules S will be dropped. Consequently, signaling molecules are received at a lower rate, which leads to a lower concentration of S in the neighbor nodes. Hence, the second reaction reduces its activity, and generates less request molecules R, whereupon the algorithm gradually decreases the exchange of job molecules W, allowing the link to recover from the overload situation. Although this is a desirable behavior, it points out the need for another "job conservation" protocol that can handle molecule leaks.

Another problem is that in networks with non-negligible delay, the concentration of S follows the peer concentration of W with that delay. In the proposed algorithm we use the concentration of S for feedback control. A consequence of having delay in the feedback loop are oscillations. However by adapting the reaction constants accordingly, we are able to stabilize the algorithm for these situations.

5 Conclusions

In this paper we showed how Fraglets, a tag matching system to design network protocols, can be mapped to an artificial chemistry. By mapping Fraglets to an artificial chemistry, and by extending artificial chemistries to an online distributed environment, we obtained a system that enables us to implement "chemical protocols". We demonstrated such a protocol for load balancing, where the desired result emerges from the combination of the distributed reaction network and the stochastic execution algorithm.

"Chemical protocols" are interesting because they offer a new way of coupling network functionality: Instead of explicit numeric values (e.g. load differences) we use rate difference, which is much more elastic and receptive for cross talk from other network functions. Ultimately, we could organize network stacks as an ensemble of intertwined metabolic pathways, which we hope will be more robust and adaptive than the current static assembly of protocol modules.

Acknowledgments

This work has been partially supported by the Swiss National Science Foundation and the European Union, through SNP Project Self-Healing Protocols and FET Project BIONETS, respectively.

References

1. Banâtre, J.P., Métayer, D.L.: A new computational model and its discipline of programming (1986) Technical Report RR0566, INRIA.
2. Berry, G., Boudol, G.: The Chemical Abstract Machine. Theoretical Computer Science **96** (1992) 217–248
3. Paun, G.: Computing with Membranes. Journal of Computer and System Sciences **61**(1) (2000) 108–143
4. Calude, C.S., Paun, G.: Computing with Cells and Atoms: An Introduction to Quantum, DNA and Membrane Computing. Taylor & Francis (2001)
5. Dittrich, P.: Chemical Computing. In: Unconventional Programming Paradigms (UPP 2004), Springer LNCS 3566. (2005) 19–32
6. Farmer, J.D., Kauffman, S.A., Packard, N.H.: Autocatalytic replication of polymers. Physica D **2**(1-3) (1986) 50–67
7. Fontana, W., Buss, L.W.: The Arrival of the Fittest: Toward a Theory of Biological Organization. Bulletin of Mathematical Biology **56** (1994) 1–64
8. Dittrich, P., Ziegler, J., Banzhaf, W.: Artificial Chemistries – A Review. Artificial Life **7**(3) (2001) 225–275
9. Dittrich, P., Banzhaf, W.: Self-Evolution in a Constructive Binary String System. Artificial Life **4**(2) (1998) 203–220
10. Tschudin, C.: Fraglets – A Metabolistic Execution Model for Communication Protocols. In: Proc. 2nd Annual Symposium on Autonomous Intelligent Networks and Systems (AINS), Menlo Park, USA (2003)
11. Cybenko, G.: Dynamic load balancing for distributed memory multiprocessors. Journal of Parallel and Distributed Computing **7** (1989) 279–301
12. Hosseini, S.H., Litow, B., Malkawi, M., McPherson, J., Vairavan, K.: Analysis of a graph coloring based distributed load balancing algorithm. Journal of Parallel and Distributed Computing **10** (1990) 160–166
13. Xu, C.Z., Lau, F.C.M.: Analysis of the generalized dimension exchange method for dynamic load balancing. Journal of Parallel and Distributed Computing **16** (1992) 385–393
14. Bahi, J., Couturier, R., Vernier, F.: Synchronous distributed load balancing on dynamic networks. Journal of Parallel and Distributed Computing **65** (2005) 1397–1405
15. Canright, G., Deutsch, A., Urnes, T.: Chemotaxis-Inspired Load Balancing. In: Proceedings of the European Conference on Complex Systems. (2005)
16. Gillespie, D.T.: Exact Stochastic Simulation of Coupled Chemical Reactions. Journal of Physical Chemistry **81**(25) (1977) 2340–2361
17. Post, E.: Formal Reductions of the Combinatorial Decision Problem. American Journal of Mathematics **65** (1943) 197–215
18. Sauro, H.M., Ingalls, B.P.: Conservation analysis in biochemical networks: computational issues for software writers. Biophysical Chemistry **109** (2004) 1–15
19. Hofmeyr, J.H.S.: Metabolic control analysis in a nutshell. In: Proceedings of the International Conference on Systems Biology, Pasadena, California (2000) 291–300
20. Dittrich, P., di Fenizio, P.S.: Chemical organization theory: towards a theory of constructive dynamical systems. Bulletin of Matematical Biology **69**(4) (2005) 1199–1231