Spatial Coordination of Pervasive Systems through Chemical-inspired Tuple Spaces

Mirko Viroli[1], Matteo Casadei[1], Sara Montagna[1], Franco Zambonelli[2]
1) University of Bologna – 47023 Cesena (FC), Italy
2) University of Modena and Reggio Emilia – 42100 Reggio Emilia, Italy
mirko.viroli@unibo.it, m.casadei@unibo.it, sara.montagna@unibo.it, franco.zambonelli@unimore.it

***Abstract*—Pervasive computing calls for developing distributed infrastructures featuring large-scale distribution, opennes, context-awareness, self-organisation and self-adaptation. There, it is quite natural to see services (software functionality, data, knowledge, signals) as spatial concepts: they are naturally diffused in the network, and in each location they are sensitive to the context and compete with each other—as such, they can be active in one or multiple regions (niches) of the network.**

To support and engineer this scenario, we propose a nature-inspired coordination model of chemical-inspired tuple spaces. They extend standard tuple spaces with the ability of evolving the "weight" of a tuple just as it represented the concentration of a chemical substance in a biochemical system, namely, in terms of reaction and diffusion rules that adaptively apply to tuples modulo semantic match. We show that this model can be used to enact self-* properties in pervasive systems, through typical spatial patterns involving computational fields, paths, and segregation.

I. Introduction

Emerging network scenarios witness an increasing availability of pervasive sensing and actuating devices (RFID tags, PDAs, localisation devices), which will densely populate our everyday environments with digital information of users and locations, and will tightly integrate with the Web—seen both as a shared service space and as a platform for social networks. This will eventually lead to the emergence of a very dense, spatially distributed shared infrastructure for the provisioning of general-purpose digital services: traditional Web services (social networks, video broadcasting) enriched with new interaction means (smaller displays and cameras) and exploiting contextual information, pervasive location-based information services (findind nearest post-office, operas in museums, goods in a market), services to coordinate activities of situated users (intelligent lights and signs, user-adaptive advertisement displays), and so on. Such highly decentralised and dynamic scenarios require the infrastructure to feature self-organisation, self-adaptation and self-management, in order to tolerate opennes in service functionality, knowledge, and user data and needs.

There is a significant research literature applying nature-inspired approaches to tackle self-organising systems like the above ones, relying on various metaphors: physical metaphors [1], chemical metaphors [2], biological metaphors [3], together with metaphors focussing on higher-level ecological/social metaphors (e.g. trophic networks [4], [5]). Among them, we develop on chemistry for it enjoys two key properties: it is one of the areas where self-organisation properties have been studied first, and it allows for a simple and exact computational description thanks to the work of Daniel Gillespie [6], which represents the cornerstone of the whole Computational Systems Biology (CSB) research field [7].

Starting from works in CSB and ideas of existing self-organising coordination models (SwarmLinda [8] and Tota [1]), we propose a chemical-inspired tuple space model—namely *biochemical tuple spaces*, formally presented in [9]. This is a variant of standard tuple spaces, where tuples are associated with an activity/pertinency numeric value resembling a chemical concentration, and measuring the extent to which the tuple can influence the coordination state—e.g. a tuple with low concentration would be rather inert, i.e. taking part in coordination with low frequency. Chemical-like laws, properly installed into the tuple space, evolve concentration of tuples so as to make interesting self-organising coordination properties emerge. Although the idea of using chemical-inspired models is not new, this model is innovative due to the following ingredients, whose combination can provide a general and flexible coordination model for spatial pervasive computing:

- *exactness* — The stochastic behaviour of chemical laws is *exact* w.r.t. natural chemistry in the sense of [6]. As a consequence, the full power of natural/artificial chemistry is enabled, supporting scenarios including dynamic decay, reinforcement, aggregation, and "evolutionary" competition. Also, this allows to rely on the metaphor of population dynamics, as developed in prey-predator systems [10].
- *diffusion* — Chemical laws are enhanced with a mechanism of tuple transfer (from one tuple space to one in the neighbourhood) mimicking how chemical substances cross biological membranes. Due to this diffusion feature, tuples are reified as spatially-distributed structures in the coordination system (sort of clouds), which can be then suitably used to model the distributed structure/interface/state of services.
- *semantics* — Chemical laws are applied modulo semantic match (like e.g. in ontology-based web matchmaking [11], [12]), instead of syntactically. Accordingly, chemical laws (and hence the overall system behaviour) support opennes and context-dependency for they are fired depending on the degree of semantic match with tuples,

hence they can be seamlessly influenced by the ontology of the application domain, by contextual information in each location, service/request match, user profiles, and so on.

After presenting the necessary background on computational biochemistry (Section 2), and the proposed coordination model (Section 3), we focus on showing that the model promotes a view of coordination of pervasive services as a spatial computing scenario (Section 4), and then conclude.

II. Background on Computational Biochemistry

A. Chemical reactions

Computational biochemistry is based on the framework of CTMCs (Continuous-Time Markov Chains), which can be understood as state-transition systems in which transitions have a rate specifying the average frequency at which they occur—and assuming the memoryless property. The motivation for using this particular stochastic meta-model comes from the work of Gillespie [6], which argued that a stiff solution of chemical reactions can be simulated as a CTMC computational system. Consider a solution of substances X, Y and Z, and a chemical reaction of kind $\mathtt{X+Y} \xrightarrow{r} \mathtt{Z}$, then the rate at which the reaction fires is computed as $r * \#(\mathtt{X}) * \#(\mathtt{Y})$, i.e. the chemical rate multiplied by the number of possible combinations of molecules that can be involved in the reaction. In [6], an algorithm for simulating the dynamics of chemical solutions has been proposed, that is commonly used in CSB — directly or through some variant such as [13]. At each step, the markovian rate $r_1, \ldots, r_n$ of the set of n chemical reactions available is computed as seen above, and one reaction is probabilistically selected for application—i.e., the probability of picking law i is r_i/R where R is the sum of all rates. The whole process is executed over and over, and simulation time is increased at each step by Δt time units (i.e. seconds), computed – as a consequence of memoryless property – as $log(1/\tau)/R$ where τ is a random number between 0 and 1.

This algorithm is typically used to perform experiments, namely, to find actual instances of chemical system behaviour by simulation. In this paper we use this algorithm in a different way: not to simulate, but rather to define the on-line behaviour of the coordination "machine" that runs tuple spaces.

The motivation for promoting this view of *coordination through a biochemical-like medium* is the possibility of getting inspiration from at least three different kinds of metaphor: *(i)* natural chemistry (e.g. self-regulation behaviour of given biological pathways), *(ii)* artificial chemistry (chemical reactions explicitly designed to achieve certain computing patterns, as envisioned e.g. in [14]), and *(iii)* population systems (e.g. systems with preys, predators, and food as suggested in the work by Lotka and Volterra [10])—the reactions presented in this paper mostly follow the latter style, though this does not harm the generality of the model or the potential of other metaphors.

It should be noted that, although existing works apply chemical behaviour to devise distributed computing systems – see e.g. [15] – to the best of our knowledge ours is the first approach aimed at leveraging exact chemical behaviour (namely, exact chemical dynamics) to devise self-organising computing systems.

B. Chemical transfer

In this paper, other than chemical reaction dynamics, we need to precisely characterise how tuples are to be diffused from one space to neighbouring ones—which is key to support distributed coordination. To keep connection with biochemistry as exact as possible, we shall rely on mechanisms mimicking chemical transport through biological membranes, though they do not seem to be computationally characterised as chemical reactions are (after [6])—fixed chemical transfer rate is often assumed. We instead rely on a more refined chemical transfer model, combining Nernst equation [16] of Electromagnetic gradient with the idea that transfer rate cannot grow indefinitely—a maximum transfer bandwidth involving two neighbouring compartments is to be considered due to their physical characteristics.

As a result, we introduce a characterisation of chemical transfer as follows: *(i)* topological connection between compartments is reflected into a (unidirectional) *link* concept, characterised by a link rate r that measures the maximum transfer frequency for molecules, *(ii)* the actual rate transfer may be affected by a *gradient substance* G (namely, by the concentration ratio of G in the source/target compartment), *(iii)* gradient shape and strength can vary: when *gradient factor* f is 0 transfer rate is not influenced by any gradient, and remains fixed to r; when f is positive, a molecule would tend to ascend the gradient created by G; when f is negative it instead descend such a gradient—the actual gradient slope increases with the absolute value of f.

Among the various functions that can be used to model the actual transfer rate, we adopt the one shown in Figure 1 ($f = 10, r = 1$): transfer rate is bound to a low $noise$ value (10^{-2} in the Figure) when the target/source concentration ratio of G (namely c_t/c_s) is smaller than 1, while it goes like $\frac{|f|c_t/c_s}{1+|f|c_t/c_s}$ elsewhere. Negative values of f result in switching c_t with c_s, while r is treated as a multiplicative factor for the transfer rate. Constant *noise* measures the probability that a molecule is transferred even if the gradient is completely oriented in the opposite direction—i.e. noise defines a sort of annealing mechanism.

III. The Coordination Model

The proposed chemical tuple-space model is an extension of standard LINDA settings with multiple tuple spaces. A LINDA tuple space is simply described as a repository of tuples (structured data chunks like records), which is used as a coordination medium provided to external "agents": such agents coordinate their behaviour by accessing tuple spaces through primitives out, rd, and in, used respectively to insert, read, and remove a tuple. Operations rd and in can specify a tuple template – a tuple with wildcards in place of some of its arguments –, and their execution blocks until a matching tuple is found.

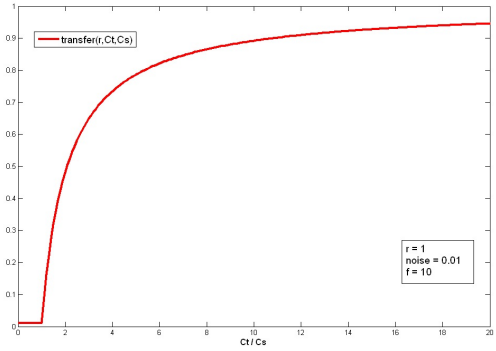

Fig. 1. Chart of the function adopted to model chemical transfer rate.

This model is used in distributed systems to provide agents with mediated interaction, supporting spatial and temporal uncoupling, and has already been used as "space" where to enact self-organisation behaviour, as e.g. in SwarmLinda [8] and Tota [1].

The basic idea underlying the proposed model is to attach an integer value called "concentration" to each tuple: such a value can be thought of as a measure of the pertinency/activity of the tuple—the higher it is, the more likely and frequently the tuple is retrieved and hence influences system coordination. Primive out can now be used to inject a tuple with any initial concentration: if the same tuple was already occurring in the space, the two tuples will join and their concentrations summed—chemically speaking, out amounts to injecting a chemical substance into a solution. Primitive in can be either used to entirely remove a tuple (if no concentration is specified), or decrease the concentration of an existing tuple—in amounts to removing (partially or entirely) a chemical substance from a solution. Primitive rd is similar to in but just reads instead of removing tuples—i.e. rd amounts to observing a chemical substance in a solution in order to know its concentration.

What is essential in our model is the fact that concentration of tuples "spontaneously" evolves exactly as in biochemistry. Coordination rules in the form of chemical reactions can be installed into each tuple space: the only difference with respect to standard chemical reactions is that they now specify tuple templates instead of molecules—a chemical reaction is applied when for each specified reactant there currently exists a matching tuple in the space. As such, the installed coordination rules affect the concentrations of tuples over time precisely as described in Section II-A, namely, the tuple space runs as a sort of chemical simulator, picking reactions up probabilistically, and applying them so as to change concentrations, and correspondingly waiting a given time interval before proceeding again.

A whole coordination system is deployed as a set of tuple spaces – ideally one in each node of the network – characterised by a concept of topological structure induced by the neighbouring relation of tuple spaces. Interaction between tuple spaces is achieved through a special kind of chemical reaction that, other than just changing tuple concentration, fires some tuples to a tuple space in the neighbourhood, chosen probabilistically. Such reactions specify in their right-hand side not just tuples, but also so-called *firing tuples* in the form `firing`(*tuple*, *gradient-tuple*, *gradient-factor*): when a firing tuple is created, transfer rates of all outgoing links is computed according to the mechanism introduced in II-B, and are used to selected the neighboring space where the tuple has to be moved to—the higher the rate, the higher the probability of moving. As this mechanism mimics chemical transfer through membranes, it allows one to conceive systems as biological-like networks of nodes – ultimately justifying the term "*biochemical* tuple spaces" – and support concepts proper of spatial computing like data/service diffusion or gradient generation as in computational fields [1], as shown in next sections.

Differently from standard tuple spaces, our matching mechanism for retrieving tuples and for applying chemical reactions is not fixed and syntactic, but: *(i)* it is application-dependent; *(ii)* it is not discrete (a tuple matches a template entirely or not at all) but it returns a *vagueness* value between 0 and 1 [12]; *(iii)* it can be based on semantics rather than being syntax-oriented [11], [12]. In particular, chemical reactions are applied modulo match, which means that the actual rate of the chemical reaction is to be multiplied by the degree to which the reaction's reactants match the concrete tuples to which the reaction is applied—e.g., if match yields 0.5 the chemical rate (i.e. velocity) is actually halved.

A suitable approach to provide powerful semantic matching in the context of the Web (and of pervasive computing) is that of fuzzy description logic [12]: an OWL ontology describes the application domain as a language of concepts, and matching amounts to retrieve the degree to which a tuple (seen as an individual of the application domain) is an instance of the concept expressed by the template. As an example based on a car selling system taken from [12], we could check if the tuple representing a given car matches the template of a user's preferences, e.g., cars costing less than 26000 euros (27000 if they have an alarm system), having air conditioning, and either black or green external colour (these three conditions being e.g. to be treated as equi-relevant)—if only the colour does not match, an intermediate result like 0.66 could be provided. In this paper we do not focus on any particular implementation of matching, although we recognise that – as far as open pervasive systems and the Web are concerned – the approach in [12] is particularly suited. We hence simply assume that the actual rate of chemical reactions is automatically tuned (decreased) by the semantic information carried by the tuples to which the reaction is applied.

IV. Coordinating Spatial Services

As discussed in Introduction, chemical-inspired tuple spaces find applications in the context of open pervasive computing, as a means to coordinate the behaviour of spatially distributed digital services. We start from the idea of disseminating each location of the pervasive system with a tuple space, programmed with proper chemical reactions. Services – software functionality, data, knowledge, signals, user requests – reify

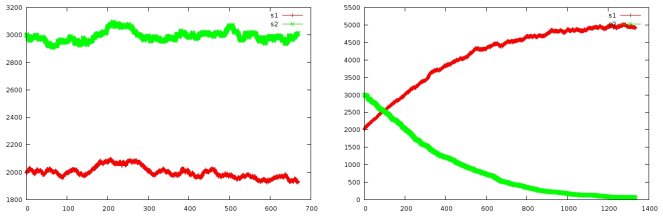

Fig. 2. Service symmetric (left) and asymmetric (right) competition.

their relevant information (structure, interface, state) through proper tuples, which get combined and diffused by such reactions. Namely, tuples are evolved and subject to change of concentration in each location, and this behaviour can be understood in terms of survival/extinction/regionalisation of services, and of spatial computing patterns as presented in the following.

An example application scenario for this kind of services is in pervasive displays infrastructures, where news and advertisement services are required to self-organise their spatial location, content, and visualisation policy, to the context-dependent information available, such as users' profile.

A. A Reference Scenario of Competing Services

We initially consider a simple yet interesting scenario in which a single tuple space mediates the interactions between services and their users in an open and highly dynamic system—this examples will later be evolved into multiple tuple spaces. In this context there is no knowledge about which services will be deployed and how extensive their use will be, i.e. whether they will successfully attract client needs, hence semantic matching will be needed to dynamically bind services to clients.

We aim at equipping the system with a self-adaptive behaviour such that: *(i)* services that do not attract users fade until eventually disappearing from the system, *(ii)* successful services attract new users more and more, and accordingly, *(iii)* overlapping services compete one another for survival, so that some of them eventually come to extinction. Note that such a competition behaviour is not mandatory in our framework: it should be considered as a relevant example of self-organising coordination policy.

An example protocol for service providers can be as follows: tuple `publish(service(Ids,Desc))` is first inserted in the space to model publication, then – cyclically – any tuple `toserve(service(ids,Desc),request(Idc,Req))` is searched that is meant to contain information about an interested client `Idc`, and accordingly its request is served eventually producing a result `reply(Idc,Rep)`. Dually, a client inserts a request as a tuple `request(idc,req)`, and accordingly retrieves a reply. Note that the tuple space is charged with the role of matching a request with a reply, creating tuples `toserve`, where a request is semantically matched with the service that is chosen to serve it. Most notably, the strategy by which this matching is performed is responsible for determining successful services—some services might end up with having been never exploited, some others may become intensively used.

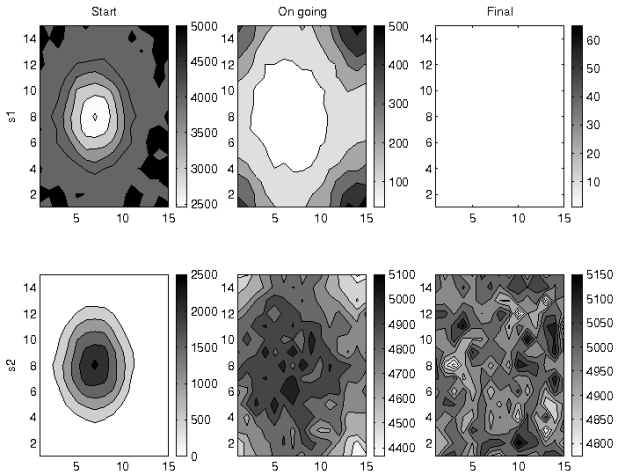

Fig. 3. Spatial competition: after an initial pointwise injection, service `s2` (down) globally overcomes `s1` (top).

The rules that enact the described behaviour are as follows:

$$\texttt{(DECAY)} \qquad \texttt{SERV} \xrightarrow{r\_decay} 0$$

$$\texttt{(USE)} \qquad \texttt{SERV} \mid \texttt{REQ} \xrightarrow{r\_use} \texttt{SERV} \mid \texttt{SERV} \mid \texttt{toserve(SERV,REQ)}$$

Rule (DECAY) states that any service tuple may fade with a negative exponential dynamics: since matching result is a multiplicative factor for the overall *decay rate* of a service, the latter is in between 0 and r_decay depending on the matching degree. On the other hand, rule (USE) has a twofold role: *(i)* it first semantically matches a service and a request (the higher the match, the more likely service and request are combined), and accordingly creates `toserve` tuple and removes requests; and *(ii)* increases concentration of service, so as to provide a positive feedback to contrast the negative one of decay—this rule resembling the prey-predator system described by Lotka-Volterra equations [10], [6]. We refer to *use rate* of a couple service/request as r_use multiplied by match degree as usual.

Consider a scenario with two services $s1$ and $s2$ matching the same requests: both with decay rate 0.01 and use rate 0.05, but initially having concentration of 2000 and 3000 respectively. The example reported in Figure 2 (left) shows that, upon a fixed incoming rate of requests (50 per time unit in this case), the two services remain in equilibrium at the initial state.[1] Such an equilibrium is however unlikely in practise, unless the two services are really identical, i.e. two instances of the same service. In fact, if the services feature even a slightly different use rate, then one of the two loses and starts fading until completely vanishing, as shown in Figure 2 (right) where the use rates for $s1$ and $s2$ are 0.06 and 0.04, respectively.

B. Spatial competition

Now suppose that instead of a single tuple space, we have a network of tuple spaces, all programmed with the above set

[1]For the sake of our exploration, we developed a prototype ad-hoc simulator directly implementing Gillespie's algorithm, and charted simulation results by using tools like `gnuplot`.

of chemical laws plus a simple diffusion law for service tuples (where no gradient information is specified):

$$\texttt{(DIFFUSE)} \quad \texttt{SERV} \xmapsto{r\_diff} \texttt{firing(SERV,null,0)}$$

The resulting system can be used to support a pervasive computing scenario in which the infrastructure coordinates users and services, such that when a service is injected into a node of the network (e.g. the node where service developer resides), it starts diffusing around on a step-by-step basis, until possibly covering the whole network—hence becoming a global service.

In particular, we are interested in observing the dynamics by which the injection of a new and improved service (an upgrade) may eventually result in a complete replacement of previous version. In the following we assume as a reference a torus-like network of 15×15 nodes (namely, a grid where nodes on the boundary are actually connected to nodes on the other side, so that every node features 4 neighbouring nodes) with link rate equal to 10^6 and noise equal to 10^{-4}—in next sections a 20×20 torus is used instead. In every node, requests for using a service are supposed to arrive at a fixed rate for simplicity – though in actual systems we should expect such a rate to be highly context-dependent – and a service called `s1` is the only available to match the requests. In particular, in every node we consider a situation featuring a concentration of approximately 5000 $s1$, in spite of diffusion.

Another service `s2` is at some point developed that can serve the same requests of $s1$ with use rate 0.1 instead of 0.05 (the use rate of $s1$), namely, it is a service developed to more effectively serve requests. This service is injected into a randomly chosen node of the network, according to a very low concentration (10 in our experiments). Figure 3 shows in each column a different snapshot (from left to right), reporting concentration of `s1` on top row and `s2` on bottom row: we can observe that `s2` starts diffusing where it is injected, until completely overcoming service `s1`.

Note that in our model, as soon as a service vanishes, it serves requests with decreasing rate, until getting completely unused. Put it simply, `service` tuples are a reification of the spatial service state as enacted by the coordination infrastructure: the resulting system features self-adaptation (the best service actually wins), self-optimisation (unused services fade and are sort of garbage-collected), context-awareness (success of a service in a location depends on requests there), and opennes (the arrival of new services is not foreseen at design time).

C. Field-based Attraction

We now consider a typical retrieval scenario of spatial computing (see e.g. [1]). A device d located in a node (modelling the availability of a device there) pumps a signal f that locally diffuses so as to create a field; a requester r situated elsewhere is attracted by the device (in order to use it), and ascends the field to retrieve it. This behaviour is obtained

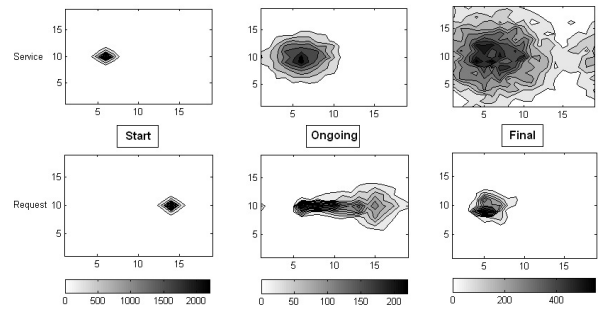

Fig. 4. Field-based attraction: a service enacts a field (top) used by requests to reach a device (down).

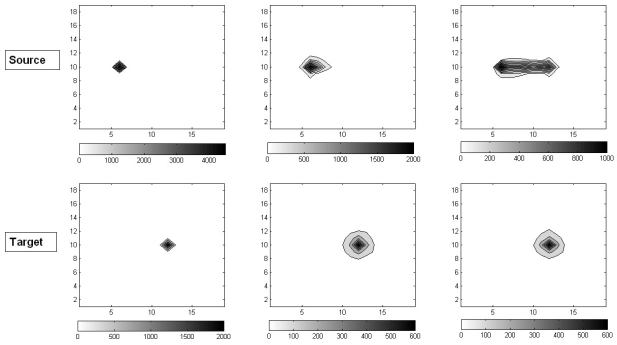

Fig. 5. Path-driven relocation: the target enacts a field (down), which attracts the source field (top).

by the chemical rules:

$$\begin{array}{lrcl}
\texttt{(PUMP)} & \texttt{DEV} & \xmapsto{r_pump} & \texttt{DEV} \mid \texttt{FIELD} \\
\texttt{(DECAY-F)} & \texttt{FIELD} & \xmapsto{r_decay} & 0 \\
\texttt{(DIFFUSE-F)} & \texttt{FIELD} & \xmapsto{r_diff} & \texttt{firing(FIELD,null,0)} \\
\texttt{(DIFFUSE-R)} & \texttt{REQ} & \xmapsto{r_attr} & \texttt{firing(REQ,FIELD,f)}
\end{array}$$

A sample evolution corresponding to this scenario is depicted in Figure 4: in particular, the bottom row reports field diffusion, while the top row shows request relocation—the adopted simulation parameters are: $f = 10.0$, $r_{diff} = r_{attr} = 0.01$, $r_{decay} = 0.0099$, $r_{pump} = 10$, initial request concentration $= 5000$. Rules (PUMP,DECAY-F,DIFFUSE-F) are responsible for creating the field: the device acts as a catalyst producing field molecules that diffuse around and decay until leading to a stable (though noisy) spatial structure—more extensive simulation results show that the number of molecules of a field tends to $r\_pump/(r_{diff} - r_{decay})$, and that r_pump drives the the field amplitude in its pumping origin, while $r_{diff} - r_{decay}$ drives the field horizon. Rule (DIFFUSE-R) is instead responsible of moving requests on a step-by-step basis so as to ascend the field—namely, at each step it is more likely to move to a node with a higher field value. Once reached the node containing the device, a specific instance of (USE) rule is used to make requests exploit the device.

Note that the above rules are presented as specialised to the attraction case for the sake of understanding, but the same behaviour can be supported by general rules of pumping, decaying and diffusion, automatically and adaptively instantiated to the application at hand thanks to semantic matching.

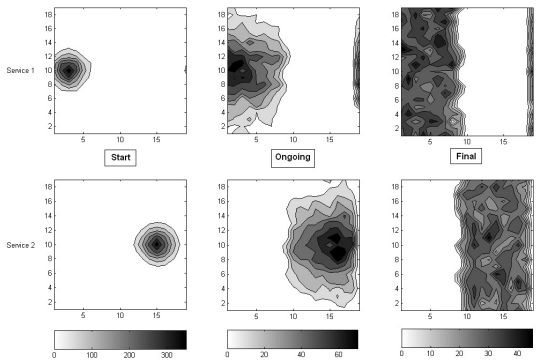

Fig. 6. Self-adaptive segregation based on a descending gradient diffusion.

D. Path-driven Relocation

A symmetric variant of the above case is the pattern in which a stable path from a source to a target location is to be created that can be used by services that has to move back and/or forth to reach the context-dependent information (or resource) available in the endpoints. This is achieved by letting both source and target pump fields, and by having source's be attracted by the target's, creating a true trail that can be either ascended or descended depending on the direction to be taken. See Figure 5.

Note that both fields and paths in these examples are self-healing, in the sense that they automatically self-adapt to changes in the network topology (if any), movements of source (resp. target), and so on.

E. Service Segregation

Consider now the case in which a network is to be adaptively divided in two (or generally more) regions, creating different contextual niches—useful e.g. to cluster data. This can be achieved by injecting different contextual information in the network, namely, two contextual services `s1` and `s2` that should logically divide the network in two segregated regions. This behaviour is obtained as shown in Figure 6.

Two locations on the opposite side of the network are supposed to be elected as centres of the two regions: from them, `s1` and `s2` are pumped as if they where fields. However, reaction rules are to be designed such that `s1` and `s2` descend each other's field. The result is that, although they diffuse due to the noise management (recall chemical transfer model in Section II-B) they will stay well separated. Note that the size of the two regions can be tuned by properly choosing diffusion rates, e.g., creating asymmetric regions.

V. Conclusions

The proposed chemical-oriented extension of the tuple space model can be regarded as a ground for building self-organising coordination infrastructures for open, spatial-oriented service systems. This infrastructure could be seen as a distributed virtual machine playing the role of an exact biochemical executor [6], which enacts semantic-oriented chemical reactions that evolve the representation of state/interface/structure of pervasive services.

In this paper we showed examples of field creation and climb that very much resemble physical metaphors as in [1] and of path creation and region segregation that more resembles biological approaches as of stigmergy [17], [18]: the biochemical approach presented in this paper seemingly appears to be able to cover the expressiveness of both metaphors along with the chemical one. In future works, we plan to further study and investigate on such self-organisation patterns: the main goal is to tackle concrete application domains in pervasive service ecosystems as also envisioned in [19].

References

[1] M. Mamei and F. Zambonelli, "Programming pervasive and mobile computing applications: The tota approach," *ACM Trans. Softw. Eng. Methodol.*, vol. 18, no. 4, pp. 1–56, 2009.

[2] A. P. Barros and M. Dumas, "The rise of web service ecosystems," *IT Professional*, vol. 8, no. 5, pp. 31–37, 2006.

[3] J. Beal and J. Bachrach, "Infrastructure for engineered emergence on sensor/actuator networks," *IEEE Intelligent Systems*, vol. 21, no. 2, pp. 10–19, 2006.

[4] G. Agha, "Computing in pervasive cyberspace," *Commun. ACM*, vol. 51, no. 1, pp. 68–70, 2008.

[5] C. Villalba, A. Rosi, M. Viroli, and F. Zambonelli, "Nature-inspired spatial metaphors for pervasive service ecosystems," in *Workshop on Spatial Computing*, Venice Italy, Oct. 2008, informal Proceedings.

[6] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *The Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.

[7] J. Fisher and T. A. Henzinger, "Executable cell biology," *Nature Biotechnology*, vol. 25, pp. 1239–1249, Nov. 2007.

[8] R. Menezes and R. Tolksdorf, "Adaptiveness in linda-based coordination models," in *Engineering Self-Organising Systems: Nature-Inspired Approaches to Software Engineering*, ser. LNAI. Springer, January 2004, vol. 2977, pp. 212–232.

[9] M. Viroli and M. Casadei, "Biochemical tuple spaces for self-organising coordination," in *Coordination Languages and Models*, ser. LNCS. Springer-Verlag, Jun. 2009, vol. 5521, pp. 143–162.

[10] A. A. Berryman, "The origins and evolution of predator-prey theory," *Ecology*, vol. 73, no. 5, pp. 1530–1535, October 1992.

[11] A. Bandara, T. R. Payne, D. D. Roure, N. Gibbins, and T. Lewis, "A pragmatic approach for the semantic description and matching of pervasive resources," in *Advances in Grid and Pervasive Computing*, ser. LNCS, vol. 5036. Springer, 2008, pp. 434–446.

[12] F. Bobillo and U. Straccia, "fuzzyDL: An expressive fuzzy description logic reasoner," in *2008 International Conference on Fuzzy Systems (FUZZ-08)*. IEEE Computer Society, 2008, pp. 923–930.

[13] M. A. Gibson and J. Bruck, "Efficient exact stochastic simulation of chemical systems with many species and many channels," *J. Phys. Chem. A*, vol. 104, no. 9, pp. 1876–1889, March 2000.

[14] L. Cardelli, "Artificial biochemistry," 2006, technical Report TR-08-2006, University of Trento Centre for Computational and Systems Biology.

[15] J.-P. Bonâtre and D. Le Métayer, "Gamma and the chemical reaction model: Ten years after," in *Coordination Programming*. Imperial College Press London, UK, 1996, pp. 3–41.

[16] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Molecular Biology of the Cell*, 4th ed., ser. Garland Science Textbooks. Garland Science, Jun. 2002.

[17] M. Casadei, M. Viroli, and L. Gardelli, "On the collective sort problem for distributed tuple spaces," *Science of Computer Programming*, vol. 74, no. 9, pp. 702–722, 2009.

[18] M. Mamei, R. Menezes, R. Tolksdorf, and F. Zambonelli, "Case studies for self-organization in computer science," *Journal of Systems Architecture*, vol. 52, no. 8-9, pp. 443–460, 2006.

[19] F. Zambonelli and M. Viroli, "Architecture and metaphors for eternally adaptive service ecosystems," in *IDC'08*, ser. Studies in Computational Intelligence. Springer Berlin / Heidelberg, September 2008, vol. 162/2008, pp. 23–32.