The Effect of Community on Distributed Bio-Inspired Service Composition (*invited paper*)

Ray Carroll, Sasitharan Balasubramaniam, Dmitri Botvich, William Donnelly

TSSG, Waterford Institute of Technology,
Waterford, Ireland
{rcarroll, sasib, dbotvich,wdonnelly}@tssg.org

The Future Internet is expected to cater for both a larger number and variety of services, which in turn will make basic tasks such as service lifecycle management increasingly important and difficult. At the same time, the ability for users to efficiently discover and compose these services will become a key factor for service providers to differentiate themselves in a competitive market. In previous work, we examined the effect adding biological mechanisms to services had on service management and discovery. In this paper we examine the effects of community on services, specifically in terms of composing services in a distributed fashion. By introducing aspects of community we aim to demonstrate that services can further improve their sustainability and indeed their efficiency.

1. Introduction

Adding biological mechanisms such as migration, replication, death and gradient emission to services has proved beneficial [1, 2] to service management and discovery. In this paper our search becomes more complex, as now we are attempting to form composed services in a distributed fashion. Distributed service composition relies heavily on service discovery, as each composition would need to perform a large number of searches. In service network with a large number of nodes (e.g. Data centres) this search can become inefficient, even with a gradient-based approach, as the number of messages required becomes large. Also, as distributed composition is so complex, limiting the set of services available for selection to those within a restricted proximity can lead to reduced composition completion.

With this in mind we adapt ideas from social communities to reduce the search overhead and improve the overall success of composition. More successful composition leads to higher service utilisation and improved sustainability of services (since services gain extra revenue from fulfilling requests). The benefit of community is that it provides a common pool of information and resources that members can utilise. One application of community is in learning the capabilities and relationships of its members by monitoring their interactions. By monitoring the interactions of services within the network, we aim to provide more directed/focussed service recommendations when attempting service composition. Although numerous research works have addressed service composition [3] [4] [5], including bio-inspired

approaches [6] [7] [8], our solution focuses on autonomous behaviours of these services to support dynamic environments (e.g. users changing requirements, service functionalities changes). We believe that applying biologically inspired approaches allows services to adapt and change, mimicking the way living organisms are able to collectively and individually survive through harsh environments.

This paper is organised as follows. Section 2 gives an overview of our biologically inspired service management solution. Section 3 then describes the community concepts we applied to services. In section 4 we discuss the simulations carried out and their results. Finally, section 5 presents our conclusions.

2. Bio-Inspired Service Management

In this section we will briefly outline the basic elements of our biologically inspired service management solution, where more details can be found in [2]. Services (or Service Agents, note we will use the terms *agent* or *service* interchangeably) are augmented with biological behaviour including replication, migration, death and gradient emission. Services provide energy to the nodes on which they run for utilising resources (e.g. cpu) and receive energy from users in return for serving requests. If an agent is not being utilised it runs out of energy and dies, hence removing redundant services from the environment. When service demand is high, agents can replicate in order to meet the increased demand. Also, when the load on a given node becomes too high agents can migrate to other, less loaded nodes.

Another feature of services is gradient emission. Agents create a gradient field by *diffusing* service advertisements to nodes in close proximity, which attracts requests towards the advertising agent (see Fig. 1a for illustration of gradient field formed by SA_1). When a service is utilised it gains energy, and uses some of this energy to maintain and extend the gradient field. The more popular a service is, the larger its gradient field and the more visible it is within the network. If a service agent is searching for another service to compose with, the gradient mechanism floods the network with search messages looking for the gradient of the required agent. By looking for the gradient, as opposed to the agent itself, the search is more efficient.

2.1. Distributed Service Composition

In our previous work the creation of the composition plan was done in a centralised manner and then passed to the service network to discover specific services. In this paper however, we adopt a distributed service composition approach. Attempting service composition in a distributed fashion makes finding the optimal composition even more difficult, as we do not have complete knowledge of the services in our network. In this paper we provide only basic optimisation of our composition plan, with more sophisticated optimisation to be addressed in future work.

Most service composition approaches view a service in terms of its input and output parameters, and/or state information such as pre- and post-conditions (e.g. [5] [6]). For example, a travel booking composed service might take a date as an input

and return a flight and hotel reservation as the output, where the actual service was provided by two individual services. For the sake or our simulations we model services and service requests only in terms of input and output parameters. Service requests describe the basic input provided and output required from the service, and from this we attempt to determine a service chain that provides the required functionality. Fig. 1b depicts the iterative composition search process.

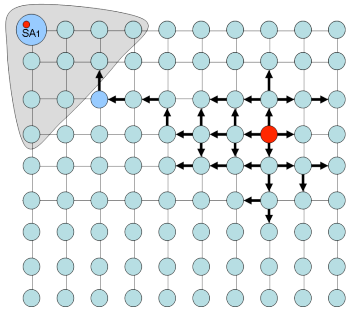

Fig. 1a - Non-Community Structure

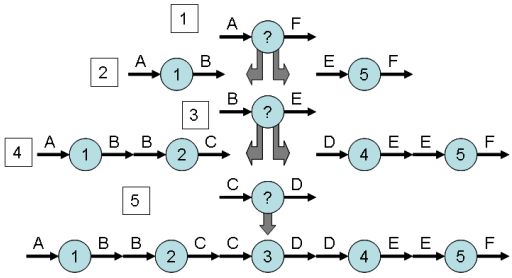

Fig. 1b - Service Composition Process

The service request arrives, describing the required service in terms of its input and output (line 1). If no atomic service is found then we try to compose a service to fulfil the request. We do this by searching for two separate services which provide the required input and the required output. Note, however, that it is not mandatory that both the input and output service are found, the process will continue if just one of these is discovered. When these services are found (line 2, service 1 and 5) we examine if they can be joined directly, if not then our next iteration must search for a *link service* to join the two (line 3). If no atomic link service is found then we again break the service into separate input and output parts and search for these individually (line 4). This process continues until a suitable link service is found (line 5) and both ends of the chain can be joined, thus forming a composed service.

3. Community

The overall goal of the community is to increase the survivability of individual services by increasing their ability to serve requests. Just as living organism communities serve to enhance the lives of its members, service communities support its service members by providing a source of group knowledge that every member can utilise. In effect, when trying to satisfy some request, a community provides a facility where community members can get advice about how best to satisfy a request. The more communities a service agent is part of, the more sources of information the agent has at its disposal.

In society more than one type of community exists, and these are typically defined by what the community's goal or purpose is. While in future work we will address multiple community types, for this paper we limit our solution to just one community type. This we refer to as the interaction community. In this community services that

interact regularly to fulfil tasks form communities. As an analogy, imagine a work place where a large number of employees, each with different expertise, interact to carry out some task. Finding and determining who has the required expertise for the task takes time, so remembering member interactions and capabilities for this task might lead to quicker job completion in the future. We can apply the same logic to services, where services that constitute a composed service remember their interactions, thereby forming an interaction community. This information can be queried later to determine suitable services for a given service request. An underlying assumption for this behaviour is that some services will be requested frequently, so as the same (or similar) requests re-occur, recording the interactions becomes beneficial.

3.1. Community Structure and Formation

The structure of a community can take many forms. These can vary from hierarchical communities with many levels, to completely flat, peer-based communities [9] (Fig. 2a and 2b). These different structures determine how knowledge is distributed throughout the community and ultimately how a search of the community is performed. If the community is completely flat (no hierarchy) then community knowledge is distributed among the services themselves, with each individual service maintaining its own view of the community. Therefore, querying the community involves querying individual services. As such, searching for a composed service would require querying individual services in a hop by hop fashion, following a chain of recommended interactions through many nodes without a broader view of the overall process. Also, since a service can only see other services within its gradient field or within a searchable distance, then the services community is also limited to this scope. This problem could be alleviated by maintaining a high gradient value to somewhat broaden the community perspective, but higher gradients require higher energy to maintain the gradient field.

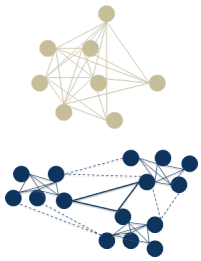

Fig. 2a - Member relationships in communities

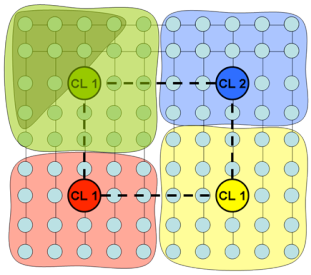

Fig. 2b - Community Structure

If a hierarchy is employed, the *community leader* (CL) retains a broader view of the community interactions and so is in a better position to make recommendations. Each agent records its own interactions and informs the CL of these. By maintaining a broader view of the community the CL can attempt to form composed services (or sections of a composed service) itself, without having to query multiple agents. In our approach we have adopted a 2-tiered hierarchy, where each community member has a

supervising CL. We create multiple CLs within the network in order to increase redundancy, where each is aware of the other community leaders within the network. The community structure employed in this work can be seen in Fig. 2b.

Since the community is based on service interactions, before community information can be utilised the community must be formed. If a community does not exist when a service request is received then the node attempts to form the composed service itself, using the normal gradient-based approach (Fig. 2a). Every time an agent interacts with another agent, as part of a composed service, this is recorded by each agent. Over time, if this interaction is used repeatedly and a certain threshold is met, the agents inform the CL of this interaction. The aim of informing the CL is to build up a more 'complete' view of all the interactions taking place within the community, thereby improving its ability to make recommendations. Once the community has been established, a service request that enters the network at a given node is forwarded to the nearest CL. If this community leader cannot satisfy the request itself, then it broadcasts the request to the other leaders in the network.

4. Simulation & Results

Our simulations evaluate two basic scenarios, employing the community and non-community based algorithms. In the non-community scenario, requests arrive at a node who attempts to compose the service by searching its list of known agents. If this is unsuccessful, it then employs the gradient-based search to look further into the service network. In the community-based model, when a request arrives, it is immediately sent to the nearest CL. In our simulation CLs are pre-selected. Future research will look at models for selecting these leaders through more 'natural' processes, where agents emerge as leaders [11]. The basic parameters used for the simulation are described in Table 1.

Parameter	Values
Arrival Rate (per sec)	15
Number of Nodes	100
IO Permutations	9-36
Community Threshold	5-20
Agents Density (per node)	1-5
Max CS Size	3-6
Gradient Size	3-5
Gradient Search TTL	5s

Table 1 - Parameter Table

For our simulation each service has only one input and one output. The number of possible types available for input or output is variable and range from 9-36. Hence, the number of possible unique service combinations range from 72 – 1260. There are a number of reasons that requests may fail. For example, if the search fails to find any part of a candidate link service then the search fails. If the composed service grows to a greater length than the defined maximum length (*csMax*), this also leads to failure.

Our search technique also employs a mechanism to avoid loops found in the service composition.

In Fig. 3a we show the success rate of forming service compositions for the community scenario versus non-community. For each scenario we show a number of simulations, each at a different gradient size. What we can see is that the community-based algorithm greatly out-performs the non-community. As the gradient varies we see minor variations in the completion rate for each scenario. In the non-community scenario, this is because as the gradient size drops we compensate by performing more gradient searches. Hence, as we see in Fig. 3b, the number of messages increases dramatically. In the community scenario the completion rate starts low, but steadily increases. This is a result of the time taken to populate the community and is also reflected in Fig. 3b, where the messaging rate for community is initially high since initial population utilises gradient-based search. Once this is complete the message rate drops-off dramatically. For the community case, gradient size does not greatly affect the search since search is primarily done via the CL. The varying gradient only has an effect when community is unable to recommend a service and so a gradient search is required.

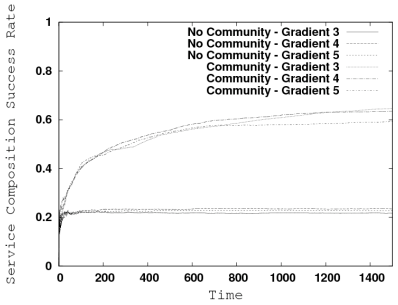

Fig. 3a - Community v Non-Community Composition Completion at varying gradients

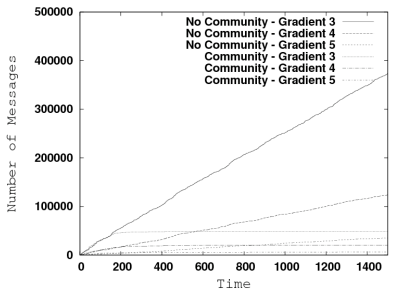

Fig. 3b - Community v Non-Community Message Volume at varying gradients

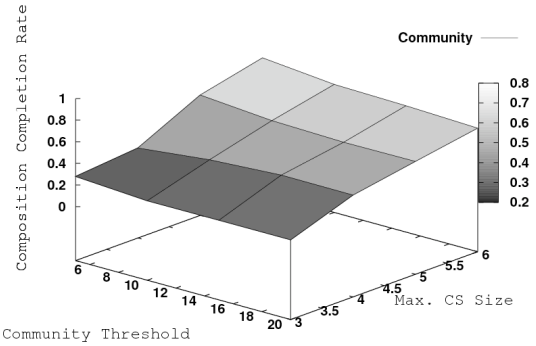

Fig. 4. - Composition Completion v Community Threshold v Max. Composed Service Size

In Fig. 3b we also see that the gradient size does affect the number of messages greatly, specifically in the non-community simulation. As the gradient size drops, more gradient searches are performed which greatly increases the number of messages used. For the community simulation, since it primarily relies on the CL, the

number of messages used is much lower. At the same time, the gradient size only has a minor affect when gradient searches are performed in the community simulation.

In Fig. 4 we compare the effect of the community threshold and the maximum composed service size parameters on the service completion rate using the community algorithm. As we can see from the graph, as the maximum composed service size is increased the completion rate increases dramatically. Increasing the size of the composed service from 3 to 6 increases the completion rate from approx. 25% to 70% (depending on the community threshold value). With regards to the community threshold we can see that altering this from 5 to 10 only has a mild effect on the completion rate. In general, the completion rate is higher when the threshold value is lower. This is as expected since a lower barrier to entering the community will lead to a community of higher population and hence more recommendations.

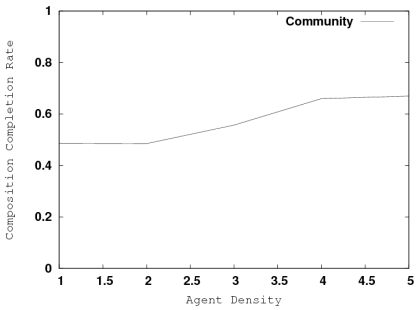

Fig. 5a – Composition Completion v Agent Density

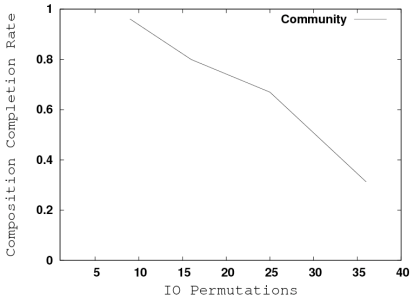

Fig. 5b – Composition Completion v IO Parameters

In Fig. 5a and 5b we examine how the composed service completion rate is affected in the community simulation by varying the agent density and IO parameters. For both these simulations the other parameters remain stable with the following values: *csMax* = 6; *Community Threshold* = 15; *Gradient Size* = 5, *Agent Density* = 5. The agent density is measured by the number of agents on each node. Fig. 5a shows that low agent density results in a low completion. This is due to less services (and hence service variation) being available for selection/discovery. As shown in the graph, as the density increase from just 1 service per node to 5, the completion rate increase from approx 48% to 67%. In Fig. 5b, as the IO permutations increase the completion rate decreases. Again this relates to the availability of suitable agents to match a request. The less variance in service types, the easier it is to find the required service. Therefore, when the number of possible types is low, the acceptance rate is high, and vice versa.

5. Conclusion

The main aim of communities in this work is to improve the sustainability of bio-inspired services by maximising the number of successful service compositions. To do this we proposed a system where services were grouped into communities and

assigned a community leader. The community leader then monitors the interactions taking place between services and records those that occur repeatedly. By maintaining this interaction knowledge, the community leader can recommend part, or even complete solutions for a given service request. As such, by utilising community we provide trusted advice through community experience. In essence, based on the community's experience of services working together, we can more quickly determine services that are a suitable match for a given request.

From the simulations carried out, we can see that our community-based solution out performs the basic non-community solution in terms of service completion. By increasing the number of successful compositions found we increase the profitability of the bio-inspired services, thereby increasing their ability to survive. At the same time, the hierarchical structure of the community ensures that the improvements made do not come at a cost, in terms of messaging overhead. This is particularly important for large topologies where flood-based searches can cause significant overheads. The results show that, by utilising community leaders to localise search, we reduce the overall cost on the network.

6. References

[1] S. Balasubramaniam, D. Botvich, R. Carroll, J. Mineraud, T. Nakano, T. Suda, W. Donnelly, "Adaptive Dynamic Routing Supporting Service Management for Future Internet", accepted for publication in proceedings of the Global Communication Conference (Globecom) 2009, Hawaii, 2009.

[2] S. Herborn, Y. Lopez, A. Seneviratne, "A Distributed Scheme for Autonomous Service Composition", in Proceedings of 1st ACM International Workshop on Multimedia Service Composition, Singapore, 2005.

[3] X. Gu, K. Nahrstedt, "Distributed Multimedia Service Composition with Statistical QoS Assurances", IEEE Transactions on Multimedia, 2005.

[4] S. Hu, V. Muthusamy, G. Li, H. Jacobsen, "Distributed automatic service composition in large-scale systems", in Proceedings of the Second International Conference on Distributed Event-Based Systems, DEBS 2008, Rome, Italy, July 1-4, 2008

[5] K. Fujii, T. Suda, "Dynamic Service Composition Using Semantic Information", Proceedings of 2nd international conference on Service oriented computing, NewYork, 2004

[6] D. Miorandi, L. Yamamoto, P.Dini, "Service Evolution in a Bio-inspired Communication System", International Transactions on Systems Sciences and Applications, vol. 2, no. 6, October 2006, pp. 573 – 587.

[7] J. Suzuki, T. Suda, "A Middleware Platform for a Biologically Inspired Network Architecture Supporting Autonomous and Adaptive Applications", IEEE Journal on selected areas in Communications, vol. 23, No. 2, February 2005.

[8] T. Nakano, T. Suda, "Self-organizing Network Services with Evolutionary Adaptation", IEEE Transaction on Neural Networks, vol. 16, no. 5, September 2005.

[9] P. Hui, J. Crowcroft, E Yoneki, "BUBBLE Rap: Social-based Forwarding in Delay Tolerant Networks", in Proceeding of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), HongKong, May, 2008.