# Adaptive Workload Distribution for Local and Remote Clouds

Lan Wang, Olivier Brun, Erol Gelenbe

# Adaptive Workload Distribution for Local and Remote Clouds

Lan Wang
Imperial College London
Email: lan.wang12@imperial.ac.uk

Olivier Brun
LAAS/CNRS
Email: brun@laas.fr

Erol Gelenbe, *FIEEE*
Imperial College London
Email: e.gelenbe@imperial.ac.uk

***Abstract*—Cloud systems include both locally based servers at user premises and remote servers and multiple Clouds that can be reached over the Internet. This paper describes a smart distributed system that combines local and remote Cloud facilities. It operates with a task allocation system that takes decisions to allocate tasks dynamically to the service that offers the best overall Quality of Service and a routing overlay which optimizes network delay for data transfer between clouds. Internet-scale experiments exhibit the effectiveness of our approach in adaptively distributing workload across multiple clouds.**

***Index Terms*—Cloud Computing, Adaptive Networked Systems, Quality of Service, Performance Management, Global Internet, World-Wide Experiments, Smart Intercontinental Networks**

## I. Introduction

The wide-spread usage of IP networks allows large-scale Cloud providers such as Amazon EC2[1], Microsoft Windows Azure[2] and Google Compute Engine[3] to build Cloud infrastructures at a global scale which facilitates the deployment of applications spanning multiple regions around the world for improving reliability and provisioning services with lower latency and a better experience for the end users. This enables service providers to distribute workloads across multiple Clouds for balancing load and offering better Quality of Service(QoS) to all the requests it receives. The selection of a Cloud for an end user also relies on other considerations such as security, cost, and energy consumption. For example, cloud bursting refers to "an application deployment model in which an application runs in a private cloud (on local servers) and bursts into a public cloud when the demand for capacity spikes"[4].

Using the Internet as the network medium between Cloud regions is the default choice offered by many Cloud providers. It is known that the routes set up by IP do not necessarily result in the best performance [5], and that IP connections may have lower reliability than other paths [6], [7], [8]. The Cloud processing latency is affected by workload distribution inside the Cloud; this not always easy to assess due to the heterogeneity in both workload and machine hardware, and the dynamic changes of load conditions over time. Much research on the solutions to these issues require sophisticated optimisation algorithms with relatively high computational complexity [9], [10], [11]. Some approaches are also based on substantial performance measurements resulting in traffic overhead [12], [13], [14], [15] and potential overhead in the Cloud. Such approaches can be difficult to put into practice due to scalability problems for large systems and the need for computational efficiency when one must make real-time and on-line decisions.

### A. Approach of the current paper

In recent work [16] we have examined how a task allocation platform (TAP) that is *internal* to a Cloud can be used to dynamically make task allocation decisions to optimise QoS, and have suggested both model based and learning based approaches. In other recent work [17] we have shown that a limited amount of re-routing over an overlay network using a measurement and big data approach can substantially average end-to-end delay for internet traffic and also reduce the perceived packet loss.

Here we combine these two ideas. In this paper we focus on the QoS that tasks receive, in particular the overall response time which is determined by the network latency to access, forward data and programs, retrieve results and data, and which includes the local or remote Cloud processing delay. In particular, we propose a practical system for adaptive workload distribution across multiple Clouds over wide area networks. The system includes a TAP deployed in each Cloud that optimises user perceived QoS and exploits the routing overlay $SMART$ over Clouds [17] for improving network delay incurred by data transfer.

In this approach, requests of users are routed to a designated local Cloud, which may well be the geographically closest one, provided that it has enough available capacity to handle the request. When the workload at the local Cloud increases, the TAP at the local Cloud can decide to forward requests to remote Clouds. In the process, TAP will consider the effect of both the data transfer delay and Cloud processing delay, each being weighted for their relative importance. The estimate of data transfer delay used in our system also takes into account the measured packet loss which will result in extra network delay for applications that use TCP for data transmission.

In order to optimise Cloud delay and network latency, our approach is meant to be easily deployable over a large population of machines, and it should be able to make fast on-line decisions resulting in good quality of service (QoS) with low computational overhead. Although it requires measurement and monitoring both of network characteristics and of local

and remote Cloud delays, we limit the frequency and overhead related to the monitoring effort, and also limit the computational complexity of decision making using reinforcement learning [18].

We therefore propose an approach, applied both in TAP and to the routing overlay, which uses Reinforcement Learning [18] with the random neural network [19], [20], [21], [22] to make fast, judicious and efficient decisions based on the knowledge learned from the past observations, while adapting to changes in workload and on-going performance of the Cloud environment. Our approach benefits from limited measurement overhead as it probes the performance of subsystems which provide better QoS, while still exploring less frequently a wider range of alternative systems that can in the future prove to provide improved QoS if the current set of frequently used subsystems result in poor QoS.

Experiments were conducted on a real large scale system operating on the Internet at a global scale and we empirically evaluate the potential of our proposed algorithms for adaptively distributing workload across multiple clouds. The experimental results that we obtain, validate the adaptiveness and effectiveness of our proposed system for dynamic environments.

## II. Reinforcement Learning with the RNN

We now summarise our design for the decision algorithm the algorithm for allocating tasks using TAP that also exploits overlay routing so as to include both network travel times, task wait times, and task execution times, when a particular Cloud or server is selected. The algorithm uses the Random Neural Network (RNN) [20], [23] with a set of $n$ neurons, where each cell or neuron represents one of the choices that may be made at each step, and the choice that corresponds to the most activated or excited neuron is selected as the one that is expected to provide the best performance. This activation level is represented by the probabilities:

$$Q_i = \lim_{\tau \to \infty} Prob[k_i(\tau) > 0], \tag{1}$$

where $k_i(\tau)$ is the activation or excitation level at time $\tau$ and in the RNN this is a non-negative integer. are uniquely obtained from the expression:

$$Q_i = \frac{\Lambda(i) + \sum_{j=1}^{n} Q_j \omega^+(j,i)}{r(i) + \lambda(i) + \sum_{j=1}^{N} Q_j \omega^-(j,i)}, \tag{2}$$

where the $\omega^+(j,i) \geq 0$ and $\omega^-(j,i) \geq 0$ are the excitatory and inhibitory weights from cell $j$ to cell $i$, and the netowrk also receives external excitatory and inhibitory signals to each cell that are represented by the parameters $\Lambda(i)$ and $\lambda(i)$ , respectively. Furthermore we have $r(i) = \sum_{j=1}^{n}[\omega^+(i,j) + \omega^-(i,j)]$.

User defined QoS criteria lead to a value $G$ that we call the "goal function" which can be measured as $G_i^t$ for an instant $t \geq 0$ at some decision point, typically on some input server $i$, while the Reward Function is the $\Gamma_i^t = \frac{1}{G_i^t}$ so that te reinforcement algorithm that we use processes these by updating the weights:

- Compute the historical value of the reward $T$:

$$T \leftarrow \alpha T + (1-\alpha)\Gamma_i^t \tag{3}$$

  where $0 < \alpha < 1$ allows us to weigh "past history" versus recent data.
- When $\Gamma_i^t > T$, then:

$$\begin{aligned} \omega^+(j,i) &\leftarrow \omega^+(j,i) + \Gamma_i^t \\ \omega^-(j,k) &\leftarrow \omega^-(j,k) + \Gamma_i^t/(N-2),\ if\ k \neq i \end{aligned}$$

- $else\ if\ \Gamma_i^t < T$

$$\begin{aligned} \omega^+(j,k) &\leftarrow \omega^+(j,k) + \Gamma_i^t/(N-2),\ if\ k \neq i \\ \omega^-(j,i) &\leftarrow \omega^-(i,j) + \Gamma_i^t, \end{aligned}$$

and the $Q_i$ are then computed and TAP chooses the server or Cloud host that coincides with the biggest value of the $Q_i$, resulting in the smallest value of the goal.

The $SMART$ algorithm for overlay network routing uses a similar approach to select paths that can minimise end to end delay in large networks [17].

## III. Task Allocation Platform for multiple clouds

In recent work [16] we proposed a practical task allocation platform (TAP) that is *internal* to a Cloud. It includes a centralized controller for distributing incoming workloads and a measurement agent on each machine for observing service performance and server health state related to user-defined QoS goals. The controller receives the requests from end users and makes fast and judicious allocation decisions for optimising the specified QoS requirements based on the knowledge leaned from the observations using learning-based algorithms. We have validated that TAP is able to adaptively distribute workloads across available servers within a cloud in response to user required QoS when there is a great diversity in the types of jobs, the class of QoS goals and the resources which are required by workloads and which are possessed by servers.

TAP is scalable as it only needs limited measurement overhead for monitoring the performance of sub-systems which provide better QoS, while still exploring less frequently a wider range of alternative systems that can in the future prove to provide improved QoS if the current set of frequently used subsystems result in poor QoS. We implemented TAP as a Linux based portable software module so that it can be easily installed on a machine with Linux OS.

In this section, we present an extension of the TAP designed for workload distribution across multiple clouds over wide area networks. As shown in Figure 1, user requests are routed to the local, which is perhaps the geographically closest cloud, if the cloud has enough capacity. The dispatcher at the local cloud receives the incoming workload and adaptively selects the best possible server based on the user-defined QoS (e.g. the response time).

If the workload in the local cloud increases, the dispatcher could decide whether to forward the subsequent requests to the remote clouds in order to balance the load and offer better QoS to all the requests it receives. The decision would rely on a variety of considerations, such as security, cost, QoS and energy consumption.

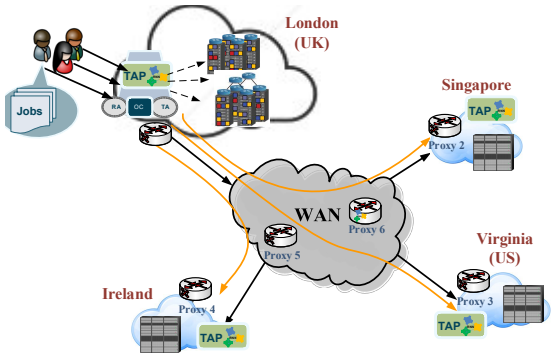


Fig. 1. The overview of the TAP for workload distribution across multiple clouds over wide area networks

We currently only concern ourselves with the QoS that tasks receive, in particular the response time observed by tasks. Obviously this response time will be determined by the network delay incurred by the access to local or remote clouds, which includes the network delay to process the request, the network delay to forward the task with its possible code and data, plus the time it takes to return the results to the sender after execution, plus the waiting time and service time inside the clouds. That is to say, the dispatcher would select a remote cloud (say the j-th cloud) to share the workload of the local cloud by considering the response time (denoted by $D^j_{job}$) which consists of the data transfer latency (denoted by $D^j_n$) which depends on the traffic conditions on the connections to the remote clouds and the proximity of the remote cloud to the local cloud as well as the response time within the cloud (denoted by $D^j_e$) which is related to the viability or health state of the cloud. Hence, a goal function for the decision can be presented as,

$$D^j_{job} = aD^j_n + bD^j_e \tag{4}$$

where $a$ and $b$ are the relative importance being placed on the data transfer latency on network connections and the processing delay inside the cloud. It should be noted that the data transfer latency for the local cloud, though not zero, may be negligible. Each cloud can report its health state via TAP regularly, including the response time within the cloud. In the following section, we present the approaches we used to predict the data transfer latency via measuring the network delay on the connections to all the external clouds. This of course is easier said than done because it depends on the nature of the transfers and on the other traffic in the connections.

### A. Data transfer delay estimation

To simplify matters, we measure the round-trip delay (denoted by $T^j_p(t)$ for the j-th cloud at time $t$) and the packet loss (denoted by $L^j(t)$ ) on each connection via pinging every a certain time interval (e.g. 1 second) thereby obtaining the network delay. A weighted average of the measurements for the round-trip delay is used as

$$T^j_p = \alpha T^j_p + (1-\alpha)T^j_p(t) \tag{5}$$

where $0 < \alpha < 1$ is a parameter which reflects the relative importance between the past values and the most recent measurement.

The packet loss needs to be considered because some applications (e.g. HTTP requests) utilize TCP to transfer the data and require retransmissions for the lost packets, which results in extra delay. The loss is measured as follows:

$$L^j(t+1) = \frac{\alpha t L^j(t) + (1-\alpha)t1_{[loss\ on\ the\ t-th\ ping]}}{t+1} \tag{6}$$

It should be noted that there is a loss on the connection to the j-th cloud if the pinging delay at the j-th cloud $T^j_p$ is larger than some fixed value $T_{timeout}$.

Therefore, the network delay on the connection to the j-th cloud considering packet loss can be expressed as

$$D^j_p = L^j(D^j_p + T^j_p) + (1-L^j)T^j_p \tag{7}$$

where $D^j_p$ appears on both sides of eq. (7) because the retransmission of a lost packet might on the average surfer the same network delay $D^j_p$ .

Therefore, we finally get

$$D^j_p = \frac{T^j_p}{1-L^j} \tag{8}$$

The data transfer time of a request to a remote cloud and its response travelling back to the local cloud can be approximated using the corresponding network delay which is obtained from the ongoing measurements on the same connection:

$$D^j_n = (\frac{S}{M})D^j_p = \frac{ST^j_p}{M(1-L^j)} \tag{9}$$

where M is the maximum packet size (bytes), and S is the total data size (bytes).

Upon the arrival of each job at TAP, the allocation decision is made using the simple greedy algorithm which selects the cloud offering the minimal response time.

$$\arg\min_j (a\frac{ST^j_p}{M(1-L^j)} + bD^j_e) \tag{10}$$

### B. Routing Overlay for Improving Data Transfer Performance

To improve data transfer performance on the connections between a local cloud and remote clouds, we use the routing overlay, "SMART", which has been proposed in [17]. It is a self-healing, self-optimizing and highly scalable routing overlay that accepts customized routing policies formulated by distributed applications according to their own needs.

The SMART overlay network is formed by software agents (denoted by "Proxy" in Figure 1) that are deployed at Virtual Machines (VM) in different sites. In our system the links between neighbouring overlay nodes use the conventional Internet protocol (IP), while multi-hop links between overlay nodes are dynamically updated based on Reinforcement Learning using Random Neural Networks. This approach only needs a limited monitoring effort by probing a few paths which have been observed to offer low packet transmission latency and yet achieves significant improvement in the end-to-end latency and asymptotically the same performance as the best path. Therefore, it can be widely deployed over a sizable population of routers.

### C. Experimental results

To validate our proposed system, we built an experimental system at the global intercontinental level which includes the local cloud in the test-bed of Imperial College London and the three remote clouds located in Ireland, Virginia and Singapore provided by Amazon AWS.

The web requests are originated using Httperf for retrieving a file of size 128K from the web servers, which generates I/O bound workload on the web servers. In the local cloud, the TAP is deployed for optimizing the web request allocation across the three web servers with distinct I/O capacity. In the remote clouds, there are other web servers deployed for load balancing. Between the local and remote clouds, the routing overlay–SMART–is used for routing the web requests and responses with the optimized network delay.

The measurement of the network delay on the connections to all the remote clouds is carried out every one second and the average response time for the Httperf requests inside each cloud is also reported every one second. As shown in Figure 2, the network connection to the cloud located in Ireland has the lowest network delay because this cloud is closer to the local cloud in London than the others, and the traffic on the connection appears to be low.

The web browser requests were originated from a client inside the local cloud at the rate of 0.5 requests per second. We varied the background workloads, which consume I/O capacity on the web servers in the local cloud over time in order to observe whether our TAP is able to adapt to the changing load conditions. Therefore, the local cloud is loaded lightly, modestly, heavily and finally lightly during 200 seconds for each of these successive conditions.

As shown in Figure 3, the incoming web requests were dispatched to the local cloud when it was under light and modest load conditions. Our autonomic TAP learned the optimal request allocation based on the online measurement

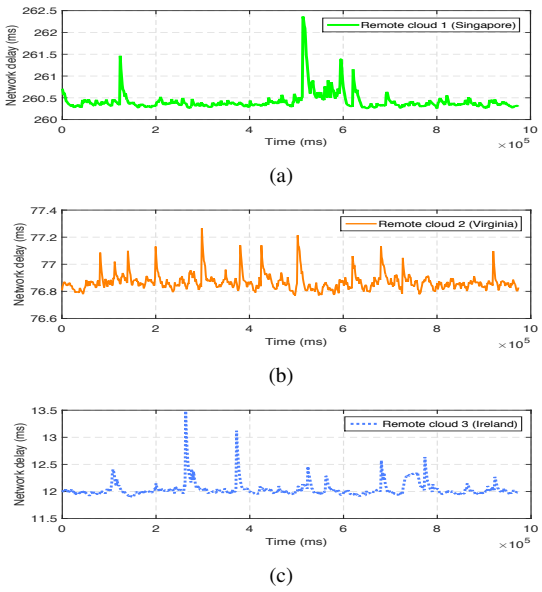


Fig. 2. The weighted average network delay over time on the connections to the three remote clouds; it is derived from the measurement of the round-trip delay and loss via pinging.

and directed the requests to the web server which provided the fastest response. As the workload in the local cloud increased to a certain high level which resulted in a response time that was significantly greater than that of one of the remote clouds (including network delays), the TAP selected the remote cloud for the subsequent web requests until it detected that the response time offered by the local cloud time had dropped significantly due to the offloading of workload from background tasks.

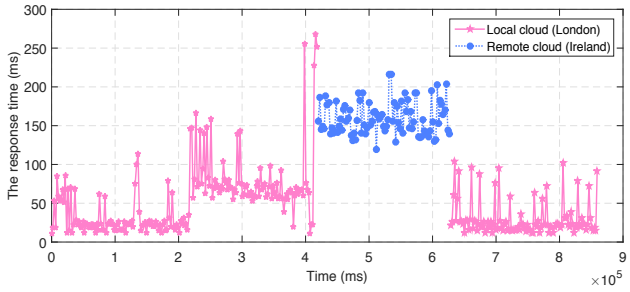


Fig. 3. The measured response time from our experiment, measured for each web request as time elapses; the different colours represent the different clouds where the requests are allocated and processed.

The improvement of network performance achieved using our overlay network is shown in Figures 4 and 5. We deployed our overlay network across Europe, Asia, North and South America, and Australia using 19 overlay nodes. The RTT observed between Moscow (Russia) and Dublin (Ireland) using the direct IP route was roughly 175 ms, whereas the RTT of the minimum latency path obtained with our overlay based adaptive routing was approximately 81.7 ms.

The results show that our proposed overlay routing algo-

rithm learns the optimal path in terms of the latency very fast, and tracks it throughout the 5-day experiment. Similarly, the available throughput between Virginia (USA) and Tokyo (Japan) using the direct IP route was 10.85 Mbps on the average, whereas the average throughput of the optimal path turned out to be 40.2 Mbps.

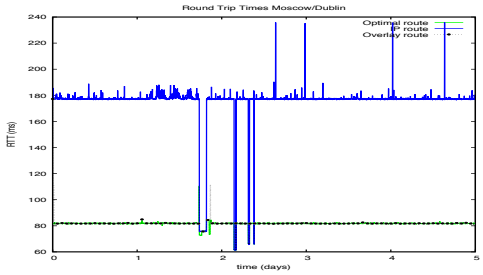


Fig. 4. RTD in milliseconds measured for the Japan-Chile connection in an experiment lasting 5 successive days.

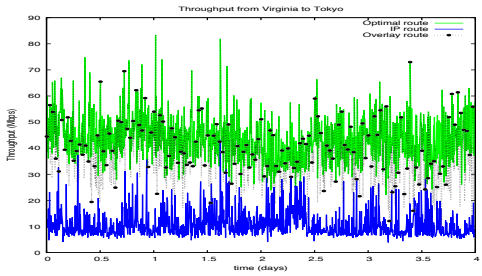


Fig. 5. Throughput (Mbps) measured from Virginia (USA) to Sydney (Australia) over 4 consecutive days.

## IV. Conclusions

This paper proposes a practical system for adaptive workload distribution across multiple clouds over wide area networks. It uses a Random Neural Network (RNN) based adaptive learning algorithm to optimize both the cloud delay within a cloud and data transfer latency between local and remote clouds in order to improve the responsiveness to user requests.

Internet-scale experiments conducted on a real large scale system at a global scale demonstrate the adaptiveness and effectiveness of our proposed system in dynamic environments. In future work, we will also take into account the energy consumption of jobs to take decisions for load distribution among local and remote Clouds.

## Thanks

Thus research was funded by the EC 7th Framework Programme PANACEA Project (www.panacea-cloud.eu), in collaboration with LAAS/CNRS (France), ATOS, IRIANC and QoS-Design.

## References

[1] "Amazon ec2," http://aws.amazon.com/ec2/.
[2] "Windows azure," http://www.windowsazure.com/.
[3] "Google compute engine," https://cloud.google.com/compute/.
[4] "cloud bursting," http://searchcloudcomputing.techtarget.com/definition/cloud-bursting.
[5] V. Paxson, "End-to-end routing behavior in the internet," in *in Proc. ACM SIGCOMM'96*, Stanford, CA, USA, August 1996, pp. 25–38.
[6] C. Labovitz, R. Malan, and F. Jahanian, "Internet routing instability," *IEEE/ACM Transactions on Networking*, vol. 6, no. 5, pp. 515–526, 1998.
[7] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed internet routing convergence," *SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, pp. 175–187, Aug. 2000. [Online]. Available: http://doi.acm.org/10.1145/347057.347428
[8] M. Dahlin, B. Chandra, L. Gao, and A. Nayate, "End-to-end wan service availability," in *In Proc. 3rd USITS*, 2001, pp. 97–108.
[9] H. Topcuouglu, S. Hariri, and M. you Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002. [Online]. Available: http://dx.doi.org/10.1109/71.993206
[10] S. Pandey, W. Linlin, S. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, April 2010, pp. 400–407.
[11] S. Zaman and D. Grosu, "A combinatorial auction-based dynamic vm provisioning and allocation in clouds," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, Nov 2011, pp. 107–114.
[12] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, ser. SOSP '01. New York, NY, USA: ACM, 2001, pp. 131–145. [Online]. Available: http://doi.acm.org/10.1145/502034.502048
[13] L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," in *in Proceedings of the 3rd ACM Workshop on Hot Topics in Networks (HotNets-III)*, November 2004.
[14] J. Touch, Y. Wang, L. Eggert, and G. Finn, "A virtual internet architecture," ISI, Tech. Rep. ISI-TR-2003-570, March 2003.
[15] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, "The case for separating routing from routers," in *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, A. Press, Ed., 2004.
[16] L. Wang and E. Gelenbe, "Adaptive dispatching of tasks in the cloud," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2015.
[17] O. Brun, L. Wang, and E. Gelenbe, "Big data for autonomic intercontinental overlays," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 575–583, March 2016.
[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.
[19] E. Gelenbe and S. Timotheou, "Random neural networks with synchronized interactions," *Neural Computation*, vol. 20, no. 9, pp. 2308–2324, 2008. [Online]. Available: http://dx.doi.org/10.1162/neco.2008.04-07-509
[20] E. Gelenbe, "Stability of the random neural network model," *Neural Computation*, vol. 2, no. 2, pp. 239–247, 1990.
[21] ——, "Random neural networks with negative and positive signals and product form solution," *Neural Computation*, vol. 1, no. 4, pp. 502–510, Dec 1989.
[22] E. Gelenbe and J. Fourneau, "Random neural networks with multiple classes of signals," *Neural Computation*, vol. 11, no. 4, pp. 953–963, 1999.
[23] E. Gelenbe and S. Timotheou, "Synchronized interactions in spiked neuronal networks," *The Computer Journal*, vol. 51, no. 6, pp. 723–730, 2008.