Scalability Metrics and Analysis of Mobile Agent Systems

Murray Woodside,
Dept. of Systems and Computer Engineering
Carleton University, Ottawa K1S 5B6, Canada
1-613-520-5721
cmw@sce.carleton.ca

ABSTRACT

Scalability is a many-sided property which can be captured in a scalability metric that balances cost, volume, timeliness and other attirbutes of value in the system, as a function of its size. Studies of typical metrics can reveal which parts of the agent infrastructure are most critical for scalability. Simple metrics are investigated for systems dominated by agent behaviour. As a system is scaled up, the length of the average tour increases and this has a major effect on performance and scalability limits. Senstivity experiments show that infrastructure improvements can improve scalability but they will not alter the general conclusion.

Keywords

Agents, mobility, performance, scalability, scalability metric, itinerary

1. INTRODUCTION

A scalability metric was recently defined for continuously and sporadically operating distributed systems [hicss][new], which generalized previous metrics which had been defined for highly parallel systems running a single computation. In the new metric a scaling plan is introcuced which defines the system configuration at different scale factors. For each scale factor the cost, throughput and performance or quality of service (QoS) are calculated, and the scalability metric depends on a kind of productivity measure:

Productivity P = (Throughput-per-time * Value-per-response-at-QoS) / Cost-per-time

The scalability from $k1$ to $k2$ is then the ratio

Scalability ($k1$, $k2$) = $P(k2)/P(k1)$

The performance calculation to predict QoS is the difficult part of this metric; the throughput and cost put it into perspective, as one can often improve QoS by enhancing the system. The metric emphasizes that scalability is an economic concept as well as a echnical challenge. The analysis may include optimization of the configuration at eachscale factor, by adjusting the investment in different factors, the distribution of the load, or any other parameters of the configuration. It is a very general framework, but so far it has been applied only to statically configured layered systems of servers for telecom applications.

The same approach is considered here to analyze scalability issues of systems of mobile agents. Each mobile agent executes a tour to carry out a task on behalf of a "user"; a task may require visits to several nodes, in an itinerary determined by the agent itself. Examples include

- agents for searching for data and assembling it,
- commerce agents for negotiating a price and terms for delivery,
- system management agents.

The performance analysis can take advantage of previous work by a number of authors. Strasser and Schwehm [7] developed a model for the delay for a single agent operation which visits several nodes. They considered the sources of delay and of processing steps in some detail, however they did not explicitly model contention. Puliafito et al [5] compared the performance of different kinds of access to remote data, by remote procedures, remote evaluation, and mobile agents. Rana [6], considered the costs of an agent meeting. Here we add the notion of the scaling plan and the scalability metric, and we consider a simple example as an illustration of the method.

2. MODEL

The scaling plan to be considered here is a particularly simple one; at scale factor k the system has k nodes with facilities for mobile agents, and a flow of Rk agents per sec. entering it. Nodes and agent flows increase together. Each agent follows a sequential tour itinerary of its own, and as k increases the length $I(k)$ of a tour increases also; for the purposes of discussion we assume an average tour length of $I(k) = \sqrt{k}$ nodes and $(1 + \sqrt{k})$ hops. Parallel operation is also possible, either by cloning an agent on entry, or during a tour.

The system cost will be taken as ck, where c is the constant cost per node. With value function $V(k)$ per response, the productivity measure is

$$\text{Productivity} = P(k) = Rk\,V(k) / ck = R\,V(k) / c\ .$$

Two value functions will be considered here, both based on the average delay T for a tour:

- $Vstep(k)$, defined as 1 if $T(k) < Tmax$, or 0 else. For a given k, if $Rmax$ is the largest R giving $Vstep = 1$, then the productivity is

  $$Pstep(k) = Rmax\,k /ck = Rmax /c.$$

- $Vsmooth(k) = T(k)/(T(k) + Tmax)$, where $Tmax$ is a target value rather than a hard maximum value. The productivity is

  $$Psmooth(k) = R\,k\,Vsmooth(k) /ck = R\,Vsmooth(k) /c.$$

A very wide range of evaluation functions could be used, for instance the average value of some function of each individual response time, dropping off as the delay becomes too long.

Each agent visit to a node requires execution resources which average S sec., including communications-related execution, installing the agent, and its operation at the node. There may also be storage operations, which we will not consider here. The amount of data to be communicated may increase over the itinerary, as the agent picks up data at each node. Communications also involves a latency in the network, for each hop in the itinerary, of L sec.

3. SCALABILITY EFFECTS OF *R*, *S*, AND *L*

Scalability is limited by delay that accumulates over a tour, which is due to work and contention. These three parameters dominate the scalability:

- R = input rate of agent tours at each node,
- S = work demand for a visit to a node,
- L = latency for a single hop.

No-Contention Evaluation

Without contention, the average delay is just $[I(k)\,S + (I(k) + 1)L]$ for a purely sequential tour. If $I(k) = \sqrt{k}$ this delay reaches $Tmax$ at the value $k = [(Tmax - L)/(L + S)]^2$. For the value function $Vstep(k)$, this is the scalability limit, without contention effects. The productivity however is unbounded, because without considering contention at the nodes there is no limit on the throughput capability of the system. And for the value function $Vsmooth(k)$ there is no exact scalability limit, just gradually declining productivity (for a given R) as k increases.

Effect of Contention

Node saturation may limit scalability first. This requires a model of contention, as a function of the traffic.

- Suppose that the computing nodes are all symmetrical, with the utilization level $U0$ for the background traffic (apart from the agent traffic). At scale factor k, the rate of agent arrivals at a node averages $R\sqrt{k}$ per sec, and the node utilization rises to $U = U0 + R\sqrt{k}\,S$.
- A simple queueing network model (see, for instance Jain [3]) of the nodes will give a sufficient illustration of the performance effect of contention. It has exponential service at every node, and an average delay at each node of $S/(1 - U)$. The average itinerary delay is then

$$T = (1+\sqrt{k}\,)\,L + \sqrt{k}\,S/(1 - U)$$

$$= (1+\sqrt{k}\,)\,L + \sqrt{k}\,S/(\,1 - U0 - R\sqrt{k}\,\,S)$$

provided the nodes do not saturate, that is provided that $R\sqrt{k}\,S < 1 - U0.$

We will consider a system with parameter values

- L = 40 msec for network latency,
- S = 30 msec for the execution at a node,
- $U0$ = 0.5 for 50% utilization of each node by other programs
- $Tmax$ = 5 sec

R is the arrival rate at each node, and will first be considered to be fixed. Thus Rk is the system throughput in agent tours/sec. For a fixed value of R below saturation, the delay T depends on k as shown in Figure 1. The no-contention limit for $Vstep$ is calculated as about 5020. . The values of R in the Figure are 2, 3, 4, and 5 agent dispatches per sec. from each node. As is typical for contention systems, $T(k)$ increases first gradually and then explosively. This makes the smooth value function $Vsmooth(k)$ drop gradually to zero. For each R the limit will occur where T crosses the

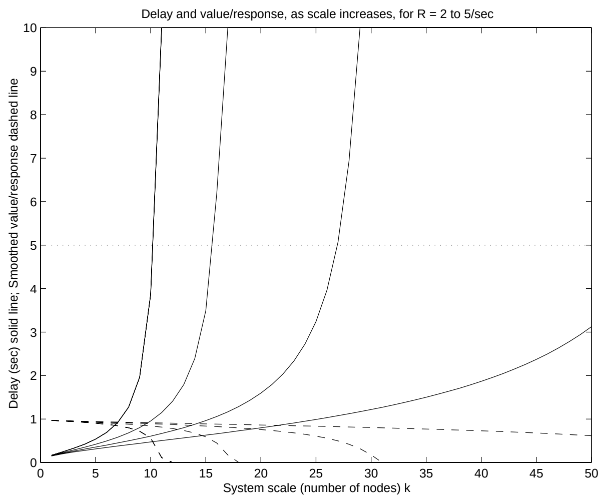

FIGURE 1. Delay versus scale factor k for various fixed values of R from 2 to 5. The dashed line is the Vsmooth value function for these delays.

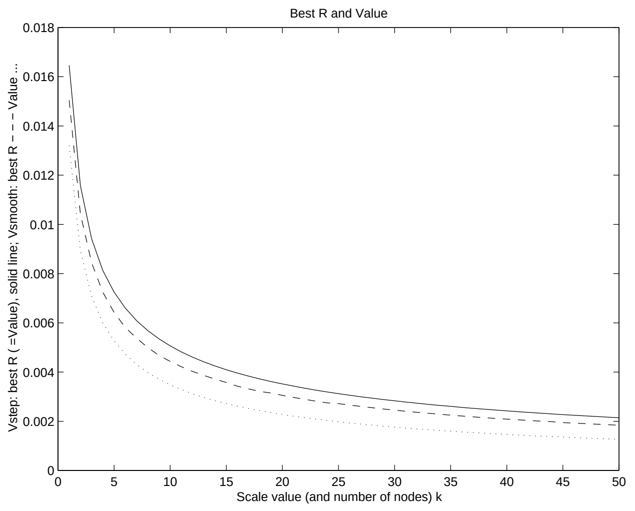

FIGURE 2. Best values of R for *Vstep* and *Vsmooth* (solid and dashed lines), and best *Psmooth* obtained (dotted), against scale factor *k*

value of *Tmax* (for the *Vstep* value function), or where *V* becomes too small (for *Vsmooth*).

We can see that for moderate values of *R* the limit is much smaller than 5020, so contention is important. However, productivity is also proportional to *R*. Is it more productive to have *R*= 4 and a scale about 15 (where the delay crosses the horizontal lime represtning the target), or *R* = 5 and k = about 10? In the former case the overall throughput is about 60 (4 x 15) and in the latter case it is about 50, so *R* = 4 is better.

We can do better by choosing *R* for each *k* so as to maximize the productivity *P(k)*. Taking the cost per node as 1, and value function *Vstep*, we have *Pstep(k) = R*, so it is just a matter of choosing *R* as large as possible to still satisfy *T* < *Tmax*. This value *Rmax* is given by

$$Rmax = (1 - U0)/(S\sqrt{k}) - 1/(Tmax - (1+\sqrt{k})\,L)$$

With value function *Vsmooth*, *R* must be found by a function maximization. The values of *R* which maximize *P* are shown for both cases in Figure 2, along with the optimal value of *Psmooth(k)*. We can see that the productivity drops off as *k* increases, essentially because of the inescapable extra delay for visiting additional nodes on a longer tour, in a larger system. The optimization of *R* means that larger rates can be served, per node, in the smaller systems.

The baseline case for estimating scalability should be a small system with more than a single node, so *k* = 5 was chosen as a baseline system. Then the scalability metric *P(k)/P(5)* was found for both value functions, and is plotted as the lower pair of curves in Figure 3. The decline has the same shape, and is dominated by the average tour length.

However there is more to scalability than just reponse time. A larger system adds value to each tour, by gathering more data from more sites. This could be considered in the productivity function. Either *Tmax* could increase with *k*, showing willingness to wait for the more valuable results, or a value multiplier could be attached to *V*. For instance, the upper pair of curves in Figure 3 show the same system but with a value multiplier proportional to the tour length. In this case:

$$Vstep(k) = \sqrt{k} \quad \text{for } T < Tmax, \text{ or } 0;$$

$$Vsmooth(k) = \sqrt{k}\;\; Tmax \,/\, (\, Tmax + T(k)\,)$$

With value determined this way, the investment in more nodes provides more valuable responses. The upper pair of curves in Figure 3 show that the system seen this way is highly scalable. .

4. SCALABILITY IMPACT OF AGENT BEHAVIOUR

In [7] Strasser and Schwehm derived a model to explain the execution demand *S* and the latency *L* for mobile agents, which will be used here to consider how some details of agent behaviour affect scalability. *S* and *L* were derived in terms of several attributes of the agent behaviour:

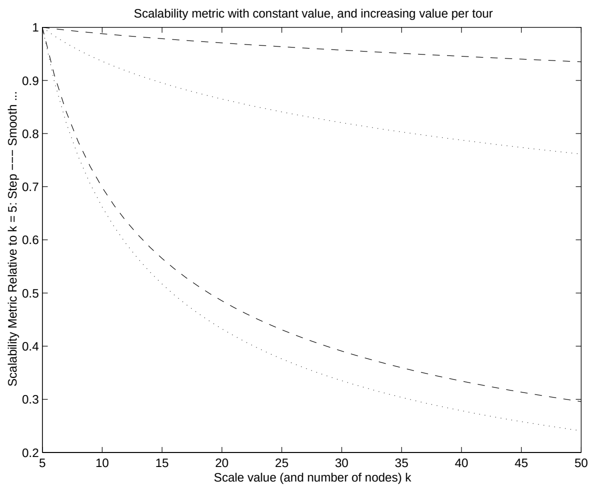

FIGURE 3. Scalability metric for the Step (dashed line) and Smooth (dotted) value functions. Lower pair: constant value per tour; upper pair: value proportional to tour length

- the size of the potential "reply" data that would have to be sent if a mobile agent were not used, and the selectivity σ of the agent (which reduces this size by a factor $(1-\sigma)$);
- the size of the code, state and initial data space of the agent,
- the probability *p* that the agent code must be transferred (i.e. that it is not cached in the destination node)

They considered that the agent state and basic data must be marshalled, transmitted and unmarshalled at each hop. The code need not be transmiited at all if it is cached at the destination, and if it is not, then it is assumed to already be available in a marshalled form at the origin of the hop, so only transmission and unmarshalling are needed. Applying their approach to our model, we can express the average demand and latency in the form

$$S = s1 + s2(1-\sigma)\sqrt{k} + s3\,p$$

$$L = d1 + d2\,(1-\sigma)\sqrt{k} + d3\,p$$

where *s1* includes CPU time for communications, marshalling and unmarshalling the state and data, and the agent procedures executed at the node, *s2* is for handling the "reply" data, which accumulates over the tour, *s3* is for unmarshalling the code if it has to be sent, *d1* is a fixed network latency, *d2* is the "reply" transmission time, and *d3* is the code transmission time. The analysis in [7] did not consider the procedure execution time at the node as part of the agent workload (because it is a constant across the alternatives they were comparing), but it has been included in the analysis here.

The service-demand parameters used here are a 20-ms execution demand, plus marshal and unmarshal times of 1 ms for state (making *s1* = 21 ms), 5 ms for reply data (*s2*), and 4 ms for code (*s3*). The latency parameters are a 30-ms pure latency *d1*; 0.5 ms for *d2*, to transmit 50 K bytes/node in the reply, before selectivity reduces it; and *d3* = 0.4 ms for 40 Kbytes of code. This gives:

$$S = 21 + 5(1-\sigma)\sqrt{k} + 4\,p$$

$$L = 30 + 0.5\,(1-\sigma)\sqrt{k} + 0.4\,p$$

Putting these expressions into *T*, we see (in Figure 4) that the scalability curves are slightly worse than before. This is because the parameters like S which were constant before, are now increasing with k and further limiting scalability.

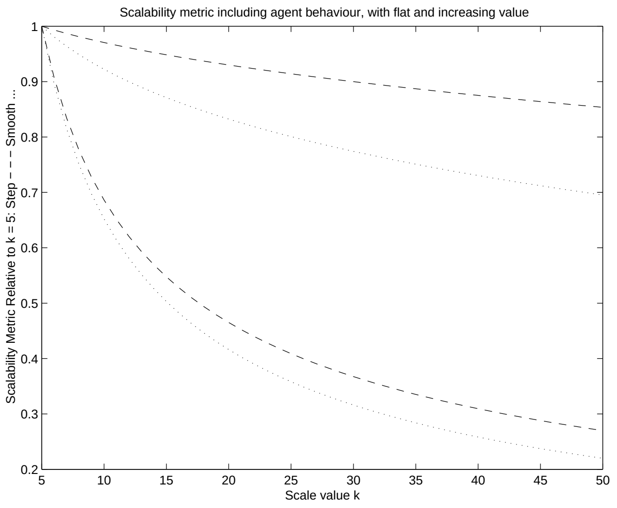

FIGURE 4. Scalability metric for the model with agent behaviour and data accumulation (cache miss 0.1, selectivity 0.9). Lower pair: constant value per tour; Upper pair: value proportional to tour length

5. SENSITIVITY TO CHANGES

Improved agent infrastructure and agent movement strategies play themselves out as modifications to *I(k), L* and *S*. Thus we can study the potential value of improvements by considering how sensitive the results are to changes in these parameters. There is not space enough here to report extensive sensitivity experiments, but it has been found that

- latency and the cache miss probability *p* for agent code made very little difference over the range studied,
- selectivity made an increasing impact at larger scales, so that at scale 50, the difference between σ = 80% and 100% gave a 50% increase in the scalability metric (see Figure 5). More selective agents improve scalability in this analysis mostly by avoiding the marshalling overhead to communicate the accumulated data collected over the itinerary.
- the reply size is similarly extremely important. Going from 50K to 500K bytes reduced the scalability at k = 50 by a factor of about 3 (Figure 6). Reply size is also more important at larger scales, because the reply is assumed to accumulate more data.

This last point emphasizes the importance of the design of the application, and the workload imposed by the users, as well as the agent strategies and infrastructure, in influencing scalability. An application that does not flood the agent with data will be better. .

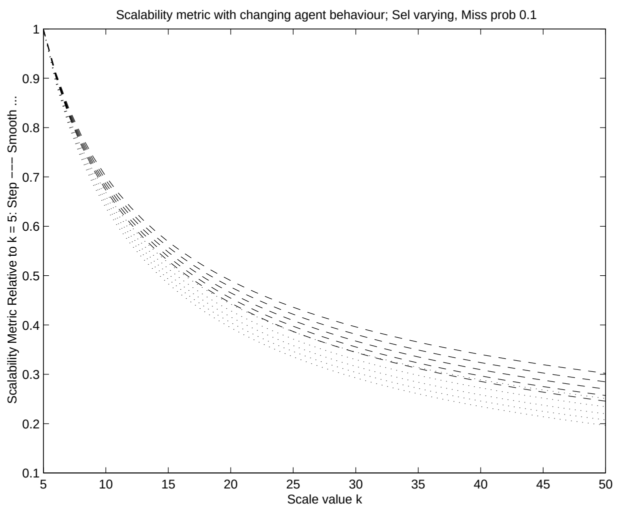

FIGURE 5. Sensitivity of the scalability metric to the *selectivity* of the agent in accumulating data during its tour. Selectivity values of 0.8 to 1.0 gave the five curves for each case (top to bottom). Dashed line for *Vstep*, dotted for *Vsmooth.* Tour value was fixed, independent of tour length.

If the value function rewards longer tours proportional to their length, we have already seen that the metric shows much better scalability, in Figure 3. The sensitivities in this case may be different, so they are recalculated in . These two figures correspond to Figure 5 and Figure 6, but with

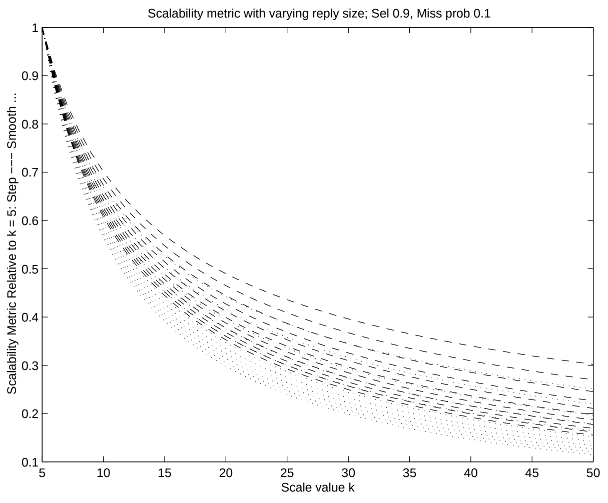

FIGURE 6. Sensitivity of the scalability metric to the *size of the "reply" data*, from 0 to 500 K bytes in steps of 50. Scalability decreases as size increases. Dashed line for *Vstep*, dotted for *Vsmooth.* Tour value was fixed, independent of tour length

V(k) including a factor $\sqrt{k}$. They show that while the scalability is much better it is still quite sensitive to the selectivity (in Figure 7) and to the size of the reply data, in Figure 8.

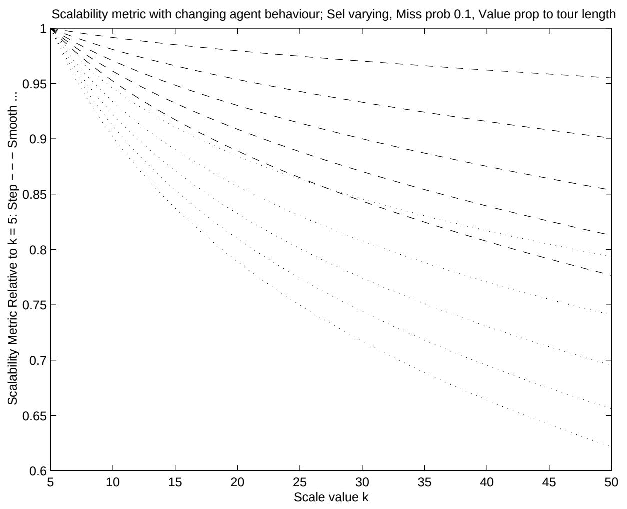

FIGURE 7. Sensitivity of the scalability metric to agent *selectivity* (0.8 to 1.0) with response value proportional to tour length $\sqrt{k}$. Dashed lines for Vstep, dotted for Vsmooth.

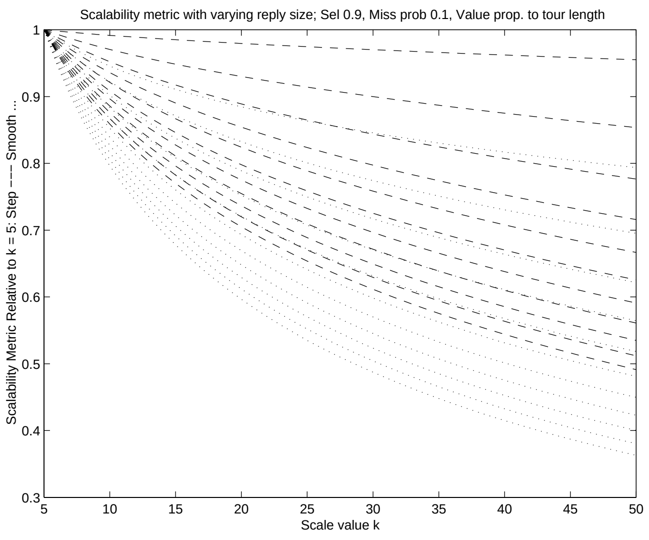

FIGURE 8. Sensitivity of the scalability metric to the *size of the "reply" data*, from 0 to 500 K bytes in steps of 50, with response value proportional to tour length $\sqrt{k}$. Dashed lines for Vstep, dotted for Vsmooth.

6. GENERALIZATIONS

It is straightforward to use the scalability metric with other models for delay, and with richer descriptions of the system. Agent communications and synchronization, cloning of agents for parallel operations, systems with hot spots, additional node resources besides processors, and network bottlenecks can all be accomodated. Queueing networks are not the only suitable performance model either. Timed Petri net models, as used in [5] and [6], can equally well provide the delay calculations; detailed simulation models or measurements on real systems can also be used. Finally, the value function can express different important attributes of the system responses as well, not just value based on the mean response time.

7. CONCLUSIONS

The scalability metric described here gives a flexible framework for capturing the essence of a scalability problem. It encourages the analyst to capture all the relevant factors, and to balance them together. It has been applied to a class of mobile agent systems, using basic and rather robust models for the workload and delay.

Some very simple models have been analyzed to discover the main features of agent system scalability, at least for independent agents that roam in a system of a given size. If the value of a tour is independent of its length, the results point to a steadily declining scalability measure over a moderate range of scales. However it is possible that a larger system with longer tours gives more valuable responses. For instance, in an e-commerce system with agents that collect data on products, a larger system would offer more selection, and a better chance to find a good match with one's exact requirements, or a good price. If the value of a response is proportional to the length of the tour, the scalability was found to be good up to many tens of nodes.

The analysis can be adapted to different needs. The examples described here have emphasized simplicity in the analysis. However the three aspects of the productivity function (throughput, cost and value per response) can be described in any degree of detail. The throughput can be divided into classes of operations with different values. The value of each completed operation can depend in the scale of the system, the delay in completing the operation, and any other quality factor that might be affected by scale or by a scaling strategy. The cost can include all parts of the system (cost of communications, which was ignored here, and also software, manpower, physical space, etc.). To illustrate this flexibility the value of an operation was changed to reflect the increased value of longer itineraries, which resulted in very different appraisal of scalability.

The examples considered here were also based on a homogeneous system and a queueing network performance model, however any kind of performance estimation could be used, and the system may contain many types of nodes of different capability.

The role of a *scaling strategy* in analysing scalability has been emphasized. In the present examples, the strategy was to adjust the traffic level R to give maximum overall productivity. The strategy could also include modifications to the agents' operational strategy as the system grows, or to the agent size. Any necessary adaptation to accommodate the increased size can be included in the analysis.

Acknowledgements

This research was supported by NSERC, the Natural Sciences and Engineering Research Council of Canada, through its program of Industrial Research Chairs.

8. REFERENCES

[1] C. Ghezzi, G. Vigna, "Mobile code paradigms and technologies: a case study", *Proc. 1st Int Workshop on Mobile Agents,* Berlin, April 1997.

[2] P. Jogalekar and C.M. Woodside, "Evaluating the Scalability of Distributed Systems", *Proc. of Hawaii Int. Conference on System Sciences,* January 1998.

[3] R. Jain, “The Art of Computer Systems Performance Analysis”, Wiley, 1991.

[4] P. K. Jogalekar, C. M. Woodside, “Evaluating the scalability of distributed systems”, *IEEE Transactions on Parallel and Distributed Systems*, to appear 2000.

[5] A. Puliafito, S. Riccobene, M. Scarpa, “An Analytical Comparison of the client-server, remote evaluation, and mobile agents paradigms”, *Proc ASA/MA 99*, Palm Springs, 1999.

[6] O. Rana, “Performance of mobile agent systems”, Proc MASCOTS 99, Univ of Maryland, Oct. 1999.

[7] M. Strasser, M. Schwehm, “A Performance Model for Mobile Agent Systems”, *Proc. Int. Conf on Parallel and Distributed Processing Techniques and Applications (PDPTA97)*, Las Vegas, 1997.