The Performance of Multiple Traders Operating in the Same Domain

V. Tschammer, A. Wolisz* and M. Walch#

GMD FOKUS, Hardenbergplatz 2, 1000 Berlin 12, Germany
#) TU Berlin, FB 20, Franklinstr. 28-29, 1000 Berlin 12, Germany

Abstract

We consider the case of trading in an environment of autonomous components. Trading is the process of matching a service request with the offers to support that particular service made by potential servers. In the case of multiple matching offers a selection of the most suitable server can be made according to the properties of the server. Among the possible selection policies we concentrate on those taking into account the actual load of the server. The selection process is usually delegated to a specialised type of component, called the trader. Various aspects influence the performance of the traders, including the delay of the information on the server load, the selection strategy, and the number of concurrent traders. We investigate several trading strategies in a scenario where traders receive the load information periodically and where multiple traders operate concurrently.

1. Introduction

The installation of wide-spread fibre-optical networks and the development of broadband communication techniques provide the technological basis for very large distributed systems. These systems will encompass numerous components which belong to a variety of interconnected subnets and organisational domains. Mostly, the components will be installed, operated, and used according to local requirements, and, therefore, will be characterised by design heterogeneity, different administration policies, and different operational characteristics such as availability, performance, and load.

This so-called open services environment requires adequate concepts for communication and distributed processing which provide new networking services, stimulate new forms of resource sharing, and support an open market of services [7]. In order to enable and to support this market, components must have access to remote services and must be able to delegate functionality to external entities. Likewise, mechanisms must be provided for introducing new components, for offering services to external users, and for coordinating access to resources.

A principle which distinguishes such an environment from traditional distributed systems built on LANs is the fact that the components generally develop independently at different locations and operate under the influence of different authorities. Each component has its own local environment which it can monitor and influence and which it is familiar with. Within the local environment, the behaviour of the component is determined by local applications and management concepts. The environmental characteristics and the internals of the components are mostly hidden from outside observers and the privacy of resources and little interdependence with the outside world is in contrast to global cooperation [13]. Correspondingly, components outside the local environment are hard to monitor and to influence by local entities. Assumptions about the properties and the behaviour of such outside components must be kept minimal. Global information and coordinated behaviour between entities operating in different environments is hard to achieve, and the principles for the development of cooperative applications must consider local autonomy, delayed information about components, and limited global coordination.

In view of this background various standardisation activities, e.g. CCITT/ X.900; ECMA/ ODP-SE; ISO/ ODP [3,4,5], and research and development project, e.g. ESPRIT/ ISA, HARNESS; RACE/ ROSA [14,15,16]; Y [7], are developing models, design principles, architectures, and integrating infrastructures for the support of the development and the operation of cooperative applications.

The general questions are how to integrate distributed applications from autonomous components and how to

* On leave from the Institute of Theoretical and Applied Informatics of the Polish Academy of Sciences, Gliwice, Poland

0-8186-2755-7/92 $03.00 © 1992 IEEE

harmonise the global application context with the local environment of the components. Main elements of the integrating architecture and infrastructure are enterprise, information, and computational models which describe the distributed information processing structures and the interactions between cooperating components in an inter-organisational environment. Another central aspect is the so-called trading concept. It has been adopted by most of the research and standardisation activities mentioned above.

2. Trading and Traders

Trading [1] is the process of matching a service request with the offers to support that particular service which are made by potential service providers. Service requests and service offers are characterised by dedicated sets of parameters which describe the attributes of the service as well as the properties and the context of the service requesting and service offering components. For example, attributes of a print service could be paper size, speed, and command language, properties of the service provider could be location and length of the printer queue, and the context of both interacting components could be access regulations and costs, respectively budget constraints. Trading is required in a open services environment because of several reasons, including:

Transparency. Trading allows the service requesting component to abstract from all the parameters which it is not interested in, or which it does not know the actual values of. In the most comfortable case, only the required service is identified in the request, maybe accompanied by a scope which delimits the search for matching offers. The handling of all other parameters, particularly the identifier, location, and status of the service providers, and the matching of interfaces are then delegated to the trading process. In this way, trading supports the establishment of associations between components of an open services environment which have only a limited view and knowledge about the environment outside their local domains.

Flexibility. Trading allows the dynamic establishment of associations between components. This makes allowance for variations in the open environment such as servers changing their service offer or their location, properties or context, and it also allows for the introduction and use of new services and service offers as well as for the adaption to the removal of existing services and components.

Tolerance of Failures and Autonomous Behaviour. In the case of component failures the trading entities can support reconfigurations and the re-establishment of associations. Likewise, autonomous behaviour such as communication and execution autonomy [12] can be tolerated by means of trading, i.e. if service providers reject communication or execution requests across established links, new associations may be created with the support of trading entities.

Optimization. Sophisticated trading includes the dynamic selection of the *best* service provider for a given request by taking into account the actual value of frequently changing service attributes and server properties. Particularly an adaption to the actual status and load patterns of the servers can be achieved, allowing for load balancing between the servers as well as for minimal waiting times of service requests.

The trading process can be performed either by the service requesters and service providers themselves or with the aid of specialised third parties, called traders. The first case is often called two-party-trading in order to distinguish it from the more usual case of trader incorporation. In the following, we will call the latter three-party-trading if it is necessary to emphasize the involvement of a trader.

3. Trading Environment and Design Criteria

Each trading process is performed within a specific context and environment. There is still an open question in research and standardisation whether trading should generally include the trading of entire services or interface trading, only. In the following we consider the more complicated case of service trading, i.e. service attributes and server properties are included as described above. Associated with the trading process, i.e. the matching of service requests and offers and the selection of suitable servers, is the requirement for service and server administration and for information about service offers and servers. The administration must allow for the introduction, removal, and modification of service offers and servers, and the information system must provide the information necessary to check and compare service offers as well as to identify and locate the servers and to check their properties.

The administrative functions may be performed with the aid of the traders or by separate entities. The information handling functions, too, may be delegated to the traders, or may be performed by separate entities. A very useful solution is suggested by [7], which uses standard X.500 directory services to handle static information, such as service attributes, server names and addresses, and X.700 management support services to deal with dynamic information, such as server status and load data.

One important factor influencing the trading process is the fact that the information about the dynamic state of the server originates at the location of the individual server and its acquisition by the trading entity causes additional costs and network load. Furthermore, the selection making

entity receives the data always with a certain delay, and, therefore, the information is at least slightly outdated. Usually, there is a trade-off between the timeliness of the information and the network load and, therefore, the effort necessary to incorporate actual or timely status information in the trading process must be balanced against the quality of the results achieved. Consequently, certain trading strategies and selection mechanisms include the use of dynamic server status information but there are also trader operations and selection strategies which rely on static information about suitable offers only.

Usually trading incorporates the following self-contained functions which can be used in sequence or independent of each other:

1. Acquisition and handling of information about service offers and servers.
2. Choice of suitable service offers, i.e. generating a list of possible matches.
3. Selection of a suitable server according to the dynamic server properties in the case of multiple matching offers.
4. Negotiations of agreements about service qualities to guarantee the best conformity between the parameter values requested and offered.
5. Automatic setup of an association between the service requester and the selected service provider.

As mentioned above, each of these functions, or parts of them, may be performed by the service requester and service provider themselves or may be delegated to traders, which operate on behalf of the delegating entity. Various policies have been described [10]. Moreover, multiple traders my be installed for reasons of availability and performance, e.g. to achieve load sharing and to avoid bottlenecks. All these cases result in the existence of multiple trading entities operating concurrently in the same domain, i.e. on the same set of servers.

In the design of trading functions and traders, particularly with respect to trader standardisation, it is important to know details about the performance of these functions and the entities implementing the functionality in the context considered. Important questions may be summarised under a user's and a designer's point of view. From a user's point of view the following criteria are important:

- Should the trading process be delegated to a trader, or should clients rely on two-party-trading?
- Do certain trading strategies or specific traders perform better than others?

From a designers point of view, the following criteria are important:

- Which trading strategy performs well under certain assumptions, e.g. delayed status information?
- How do concurrent traders and concurrent strategies perform?

From these performance considerations, several design principles can be derived. Particularly more precise answers to the following questions can be expected:

- Shall the use of certain information, e.g. the server status information, be restricted to traders, so that the possibilities of two-party-trading are limited?
- Shall standardisation prescribe three-party-trading so that the responsibility for effective and fair trading is delegated to specific entities?
- Shall certain trading strategies be favoured or prescribed in order to guarantee effective and fair trading?
- Shall each trader be associated with a dedicated trading domain, as proposed by [2], so that concurrency of traders is made impossible?

In order to support the evaluation of trading techniques and to find answers to these questions, we have already investigated a scenario were the load information was acquired for each service request separately, and we have demonstrated the influence of out-of-date information on the quality of the selection process elsewhere [10,11]. We now investigate a scenario where traders receive the load information periodically and where multiple traders operate concurrently.

4. The Performance of Concurrent Traders

For our investigations, we make some assumptions which constrain the broad variety of possible scenarios:

1. We concentrate on traders, i.e. the case of three-party-trading. The difference to two-party-trading is the fact that traders are able to optimise from a social point of view. That means in practice that a trader, executing the server selection process for more than a single client, can use a record of its previous assignments in addition to the server state information in order to make the best choice for all of his clients.
2. The traders considered all implement the same trading strategies. The detailed investigation of concurrent strategies is for further study.
3. The trading strategies considered are random choice, cyclic assignment, and best choice according to the dynamic server properties in two variants, one selecting from the whole set of possible offers and the other selecting from a randomly chosen subset.

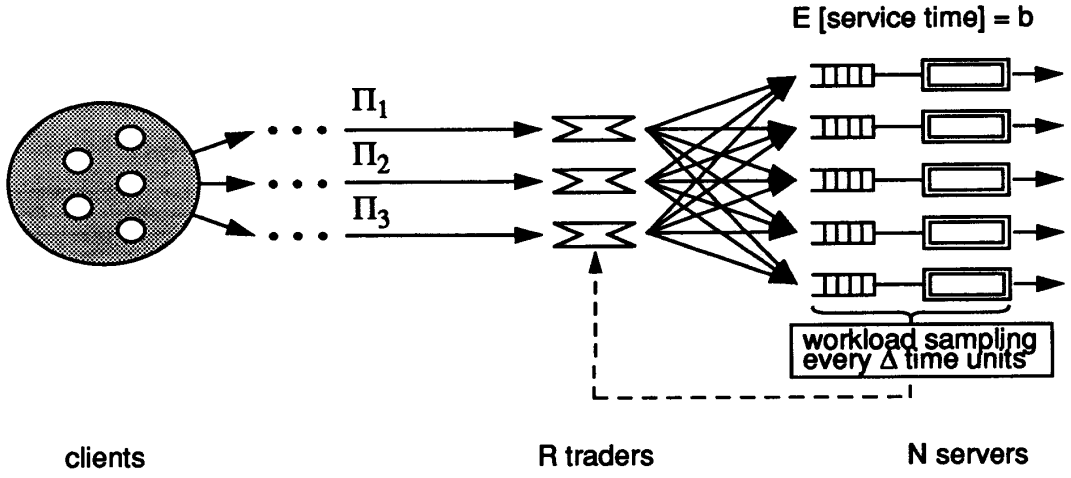

Figure 1: The generic model under study

The general schema of the system under investigation is presented in fig. 1. We assume, that the incoming service requests consult one of the R traders $T_1, T_2, ..., T_R$ according to some probabilities $\Pi_1, \Pi_2, ..., \Pi_R$ and follow the server assignment done by this trader, i.e. they join the proper queue. The traders operate independent of each other, and no trader has any information about the decisions made by other traders. We assume that all traders receive the information on the server's workload broadcasted every Δ time units, and implement the same selection strategy. In the following we assume that the time consumed by the trading process itself and the time needed to transfer the requests to the server can be neglected, i.e. we obtain the optimistic bound on the performance.

We first consider the case of selecting the server with the smallest workload. The workload of each server is estimated based on the latest server state report and the additional workload imposed by the requests assigned to this server by the considered trader since then. The selection policy is defined more precisely below.

Let us denote by t_k^0, $k = 1, 2, ...$ the moments of consecutive load samplings, $Z_i(t_k^0)$ being the sampled load of the i-th server, i= 1,2, ..., N. We shall now consider the time period between two consecutive samplings $t_k^0 < t < t^0_{k+1}$. In this period we shall distinguish moments t_k^p, $k = 1, 2, ...$; $p = 1, 2, ...$, denoting the epochs of arrival of consecutive service requests at the considered trader. Before assigning the newly arriving request, the trader computes it's *local preassignment estimate* Y_i^-, i = 1, 2, ..., N of the load of the i-th server, in the following form:

$$Y_i^- (t_k^p) = \left[Y_i(t_k^{p-1}) - (t_k^p - t_k^{p-1}) \right]$$

$$\text{with } (Y_i(t_k^0)) = Z_i(t_k^0)$$

and chooses the server with the smallest value of this estimate. Let us identify this server as the I-th one. Then, the pending request will be assigned to the I-th server, and the *local, postassignment estimate* Y_i will be computed as follows:

$$((Y_i(t_k^p)) = Y_i^- (t_k^p)) \text{ for all } (i \neq I)$$

$$(Y_I(t_k^p)) = [Y_I^- (t_k^p)]^+ + b$$

where the notation $[x]^+$ stands for x if x > 0 and 0 otherwise, while b is the service demand of the newly assigned request, i.e. additional load just assigned to server I.

The numerical data presented in fig. 2-5 have been obtained from simulation programs developed in the CSIM/SUN3 environment [8]. In order to obtain statistically significant data, additional statistical inference has been introduced. In each simulation run, the first several hundreds of events have been neglected, in order to ignore the transient period. Afterwards the total number of simulated arrivals has been dynamically increased so as to obtain the confidence interval width not exceeding 5% of the mean value, with confidence level 0.95. The confidence intervals have been estimated using the batching approach with division of the whole run into a constant number of 16

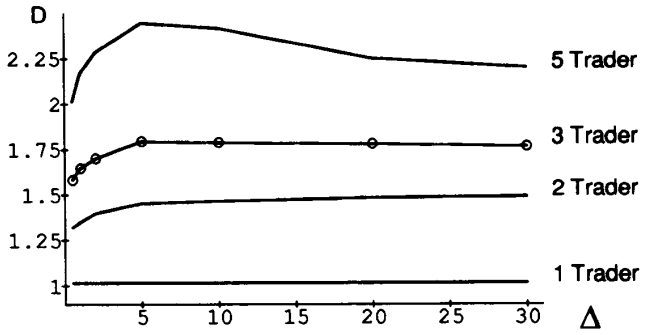

Figure 2: Choice of the server with minimal workload as computed with regard to the load generated by the trader since last measurement. N=20; $\rho = 0.7$.

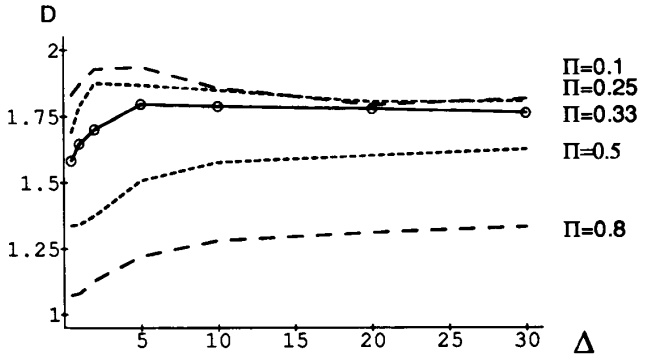

Figure 3: Different distribution of load among servers. N=20; R=3; ρ=0.7. Variant a/ solid line, b/ dashed lines, c/ dotted lines.

batches. In the figures presented, the mean values have been given, while for selected parameters circles mark the actual experiments with their confidence intervals.

Fig. 2 illustrates the results achieved with the trading policy introduced above for a server utilisation ρ=0.7. In this experiment, as well as in further examples in this section an exponential service time distribution has been assumed. The performance becomes worse with an increasing number of traders. This could have been expected, as the case of R=1 is equivalent in performance to the optimal case of a multiple server queue (the load estimates Y_i are perfect), while the concurrent operation of other traders causes the *local* estimate to be inaccurate.

In addition to that, for the case of a large number of concurrent traders, an unexpected influence of the sample interval Δ on the system performance can be observed. Certainly, the best performance can be achieved for Δ tending to zero. For medium values of Δ, however, the performance becomes worse than for the case of large Δ. Obviously, the traders tend to "adapt" themselves to the actual load situation during the longer sample intervals, in the sense that they tend to divide the load equally among the servers, while the additional information delivered by frequent information updates brings an "undesirable disorder".

In the investigations reported above, it has been assumed that the probabilities $\Pi_1, \Pi_2, ..., \Pi_R$ are equal, i.e. all traders deal with an equal share of requests. In the following we discuss the case of different shares of incoming requests being assigned to individual traders. In fig. 3 the results achieved for three variants are presented:

a/ $\Pi_1 = \Pi_2 = \Pi_3 = 0.33$

b/ $\Pi_1 = 0.5, \Pi_2 = \Pi_3 = 0.25$

c/ $\Pi_1 = 0.8, \Pi_2 = \Pi_3 = 0.1$

for the case of three traders (R=3). It can be seen, that - in absence of trading processing time - requests handled

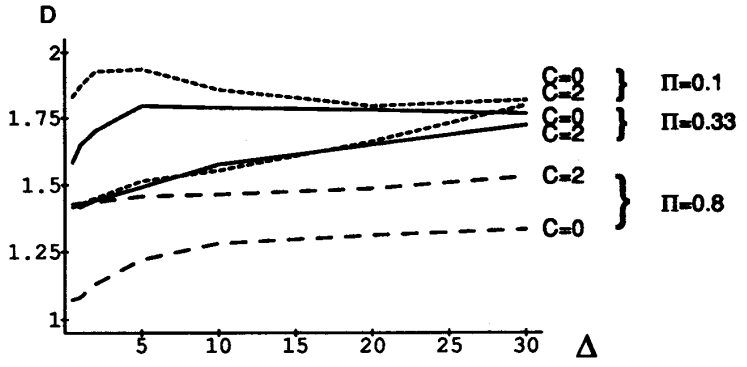

Figure 4: Comparison of no preselection strategy with the limited probing strategy for different load distribution among servers. N=20; R=3; $\rho = 0.7$. C=0: no preselection; C=2: preselection of two servers.

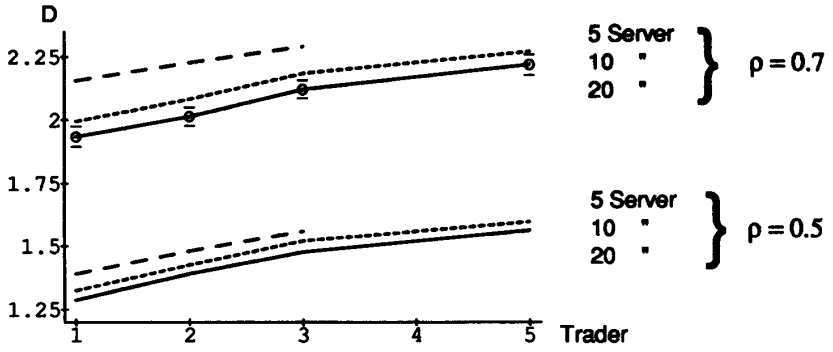

Figure 5: Cyclic selection of a server by each trader.

by the trader with the highest share experience the lowest delays.

In earlier papers [10,11] we have demonstrated, that for the server selection by individual customers, it was beneficial to use a different assignment rule: out of all N available servers some C<N are randomly preselected, and only those C are considered for the final assignment (the so called limiting probing). It has also been demonstrated, that for a wide spectrum of system parameters C=2 is the best choice.

In fig. 4 the variants a/ and c/ defined above are repeated (parameter C=0) and compared to the case, when two servers are preselected at random, while the final choice is made only among them, according to the assignment rule defined at the beginning of this section (parameter C=2).

The results are ambiguous. In the case when requests are shared equally by all traders the limiting probing performs better. If, however, the traders deal with unequal shares of requests, limiting probing is beneficial for traders serving a small share of request, but clearly adverse for the traders serving the big shares of requests.

It seemed to be interesting to what extend the use of a selection policy independent of the dynamic information on the server load could be justified. One possibility would be to use the *random selection*. Let us note, that under the assumption of Poisson arrival, the performance expressed in terms of the mean response time would, in the case of random server selection, remain uninfluenced by the number of traders! On the other hand, as each trader is allowed to use the record of it's previous assignments, we decided to additionally investigate the *cyclic assignment* policy.

In fig. 5 the influence of both the number of traders and the number of servers on the system performance for two different values of the system load ρ has been illustrated. For N=20 and ρ=0.7 the actual experiments and the exact confidence intervals (o) are given. All traders get an equal

share of requests. Let us remember (cf. fig. 1), that in the case of ρ=0.7 the mean delay under random choice was equal to $3.\overline{3}$. For any given number of servers, the performance observed for the cyclic assignment policy becomes consistently worse with an increasing number of traders. In the investigated range of parameters, however, the *cyclic assignment* policy outperforms the random assignment. This phenomenon is well known in queuing theory (cf. for example [6]) for the case of a single trader. It was interesting to see that this relation remains valid also for many concurrently operating traders. Furthermore, with a larger number of traders (e.g. five traders in our case) the *cyclic assignment* may outperform the more complicated strategies described earlier in this section.

5. Conclusions

Our investigations demonstrate that the concurrency of trading entities is critical to their performance. Generally, the performance becomes worse with an increasing number of concurrent traders. Only the random server selection strategy is not influenced by the number of concurrent traders. However, the random choice constantly performs worse than the other strategies considered.

If different shares of the number of incoming requests are assigned to the individual traders - which is the usual case in an environment of autonomous entities - the higher loaded traders make better assignments than the others, i.e. some customers are served better than others. Then, fair trading is no longer achieved. Future investigations must show whether this is compensated by the increase of service turn around time at the higher loaded traders. Cyclic assignment performs comparatively well in the concurrent environment, while the limited probing strategy which was found to be the best among those considered for single traders [10,11] does not so well in concurrency with other traders.

As a consequence a limit on the number of concurrent traders operating in the same domain must be considered in design and standardisation. At least some mechanisms of achieving an equal distribution of the incoming requests - such as a "trader trader" - should be considered.

References

[1] *ANSA Reference Manual*, Vol. C, Release 01.01, (July), Architecture Projects Management Limited, Cambridge, UK, 1990.

[2] M. Bearman and K. Raymond "Federating traders: an ODP adventure", *Proc. of the International IFIP Workshop on Open Distributed Processing*, October 8-11, 1991, Berlin, Germany.

[3] *DAF Infrastructure, CCITT Q19/VII-DAF*, (Sept.), V5-Tapiola, Finland, 1989.

[4] *Standard ECMA-XXX, SE-ODP Trading*, ECMA/ TC32-TG2/90/17, 2nd Draft, (Jan.), 1990.

[5] *ISO/IEC JTC1/SC21 N4025, WG7 Working Document on Topic 8.1* - Draft Basic Reference Model of Open Distributed Processing - Part II, (Dec.), 1989.

[6] S. S. Lavenberg (editor) "*Computer Performance Modeling Handbook*" Academic Press, Inc., N.Y. 1983

[7] R. Popescu-Zeletin, V. Tschammer and M.Tschichholz "'Y' distributed application platform" *Computer Communications*, Vol. 14, 6 (july/august), 1991, pp. 366-374.

[8] H. Schwetman "*CSIM reference manual*", Microelectronics and Computer Technology Corp., Austin, TX, 1989.

[9] V. Tschammer, K.P. Eckert and L. Henckel "Concepts for open systems cooperation in distributed CIM-structures", *Proc. of the Workshop on the Future Trends of Distributed Computing Systems in the 1990s*, (Sept.), Hong Kong, 1988, pp. 23-32.

[10] A. Wolisz and V. Tschammer "Service provider selection in an open service environment", *Proc. of the 2nd IEEE Workshop on Future Trends of Distributed Computing Systems*, (Sept./Oct.), Cairo, Egypt, 1990, pp. 229-235.

[11] A. Wolisz and V.Tschammer "Some performance aspects of trading service design", *Proc. of the IEEE INFOCOM'91*, April 9-11, 1991, Bal Harbour, FL,USA, pp. 919-928.

[12] F. Eliassen and J. Veijalainen "An s-transaction definition language and execution mechanism", Arbeitspapiere der GMD, No. 275, GMD, (Nov.), St. Augustin, Germany, 1987.

[13] Proc. of the ACM SIGOPS Workshop on Autonomy and Interdependence in Distributed Systems, (Sept.), Cambridge, UK, 1988.

[14] "ISA testbench implementation manual", Technical Report, ESPRIT Project 2267, (Aug.), 1990.

[15] "HARNESS platform basic specification extract", Document HARNESS/ PSE2/ VOL/ 101/ 0.1, ESPRIT Project 5279, (Oct.), 1991.

[16] M. Key, S. Leask, A. Oshisanwo "ROSA: object-oriented architecture for integrated broadband communication services", Proc. of the TINA '90 Workshop, (June), Lake Mohonk, NJ, 1990.