Traffic Conscious Distribution of Service Components

Miltiades E. Anagnostou[1] and Maria A. Lambrou[2]

[1] National Technical University of Athens, School of Electrical and Computer Engineering, GR-15780, Athens, Greece, email: miltos@central.ntua.gr
[2] University of the Aegean,Business School, Department of Shipping, Trade and Trasport, GR-82000, Chios, Greece, email: mlambrou@aegean.gr

Abstract. A service is commonly realized by a set of components distributed over different nodes. For example, Internet based applications are orchestrated across a large scale distributed computing infrastructures and underlying resource elements; Similarly, in TINA-like approaches intelligence for control and management of services and resource, in particular, is distributed among network nodes and user/terminal nodes. In fact a service can be seen as a set of interacting components with a common purpose, being it application-oriented or of a support nature . The placement of components in different machines is more or less empirically determined at the design phase, by loosely (and occasionally subconsciously) taking into account reasonable predictions of the component usage. Code mobility has added complexity to the distribution problem. The aim of this paper is to present a methodology of dealing with component distribution, to explore its limitations, and to present its effect on service and network design.

1 Introduction

The service-oriented computing paradigm considers services as the fundamental elements for constructing applications. Composite services as resulting by basic service components aggregation are utilized by service providers as commercial solutions to be offered to a diverse customer base. Thus, services may be viewed as open, self-contained software components that support efficient configuration as well quality of service composition of distributed applications. Services may be supplied by different business stakeholders and comprise a distributed computing infrastructure in support of intra- and cross-enterprise application integration and collaboration [1].

Quality of service (QoS) considerations, are seen as an integral part of the service design lifecycle, taking into account important functional and non-functional service properties, such as performance, security, reliability, transactional integrity and services overall cost. Today's dominant manifestation of service-oriented computing implementation is realized in terms of emerging web technologies [2] Current service design frameworks merely focus on service capabilities, interface and behavior models and notations, which are expressed in a

universal format, independent of a particular modeling tool and implementation platforms. In this paper, the component distribution analysis activities, in particular, are examined as an integrated phase of a performance-centered and thus quality of service-centered design process. The argument here made is that the service design philosophy and methodology can be explicitly extended to incorporate component allocation considerations in terms of combined mathematical programming and formal specifications activities.

The relationship between information mobility, which is supported by a network infrastructure, and the network itself, has so far been rather fuzzy; design has been based on empirical decisions, which aim at adapting network design to the needs it is assumed to serve. Ideally, given a set of services, a geographical distribution of users, and a demand pattern for the services requested by each user, one would design a minimum cost network, which would satisfy the quality requirements of all services, and, of course, the services themselves. Consider now that a service can be designed in a network independent manner: It consists of components, whose types and interactions are network topology independent, while only their distribution depends on the network. In this respect component based service design becomes an isolated problem, which can be separated from the aforementioned general problem. Therefore, service design (excluding the component distribution phase) becomes an input to this holistic design problem. Even after this reduction, the problem remains very ambitious, as its solution should produce both (a) the network topology, including node and channel capacities, and (b) the placement of service components over the network. The realisation of this grand objective cannot be achieved unless the relationship between the different factors of this problem becomes explicit. This paper takes us at least halfway to the solution of the problem, as it addresses subproblem (b) of the design, and partially subproblem (a) in the following sense: Although the network topology is taken as granted, node and link capacities can be determined or corrected by using the model presented in this paper; they can also be taken as given, which is plausible when new services are deployed over an existing network, and actually this is the most common situation. Therefore the main contributions of this paper are (i) that it presents a new design methodology, and (ii) it describes a number of steps, which are necessary to make the relationship between demand distribution and component distribution explicit.

2 Component distribution issues

To illustrate some of the issues, which will be explored in this paper, a simple example is useful: Assume that a new mobile device is in the design phase. A user directory application, which will of course include phone numbers, will be offered to the device user. The designer's dilemma is where to place the main component, i.e. in the device or in the network. Her choice of preference might be to make the device as light as possible, not only because this will give her less development pain, but also because it will be mass produced and it must be as cheap as possible. However, if the whole application is in the network, each

time a user asks for a person's number, a directory application interface (and proxy) in the phone must send a message to the main application component, thereby consuming communications resources, including expensive air-interface time. Driven be these thoughts she might after all decide to spend the extra effort and try to put the main application in the phone. Storage, then, is likely to become an issue, but the price of memory is constantly sinking and today's phones are equipped with a memory, which well surpasses the storage capacity of yesterday's computers. Yet, again, each time a user related piece of information changes, all phone resident copies of this information must also be updated, sooner or later. The obvious reaction of the overwhelmed designer would be to ask a colleague to prepare for her a cost-benefit analysis of the alternatives.

What is a common objective behind these and similar considerations in a designer's mind? Is it not to produce the cheapest possible product, which satisfies a set of quality criteria? In the previous example, putting the directory in the network may produce a cheaper device, but the response time in retrieving a person's number must be kept under an acceptable limit. In a cost conscious design, the total communication cost should be minimized. Short sighted design may ignore such considerations or pursue a partial cost optimization, but market competition is likely to punish such policies in the long term. In the specific example, a designer may choose a network based user directory on the grounds of creating a lower market price product. On the other hand, a misinformed product buyer will discover that she pays higher communication bills than her neighbour, just because of the directory application. A healthy market will sooner or later reject the specific phone design.

From this discussion it should have become clear that communication cost is an important factor in assigning components to *physical entities*, i.e. physical containers, such as nodes, terminals, and other devices. Other important factors in the realm of service provision over network infrastructures are development costs of devices and applications, and, more recently, the cost of acquiring and processing information. In this paper we mainly deal with the problem of component distribution over a given network infrastructure, based on communication cost minimization.

A service can be seen as a collection of concerted components, which should produce a desired outcome. Components exchange messages in order to establish cooperation and communicate results. The following observations add to the complexity of the component distribution problem:

- There can be common components between services. For example, a charging component may be used by different services.
- Certain components can be attached only to specific physical entities or specific types of physical entities (due to functional or administrative reasons). For example, a module, which captures human voice and converts it to a digital signal can only exist in the mobile phone.
- Communication volume reduction or load sharing may favor the distribution of copies of the same component over a network. For example, a user

directory database may be implemented in multiple copies, each serving the demand of a major city.

A general and loose formulation of the problems, which are explored in this paper is the following:

Problem 1. Assuming

1. a set of services,
2. a set of components for each service (including common service components),
3. an estimate of the volume of interaction between components produced by invoking a service,
4. a network topology,
5. a population of terminals attached to network nodes,
6. an estimate of service demand (per different service) created at a terminal, and
7. a charging scheme (i.e. an algorithm, which transforms traffic volume to cost),

find the assignment of components to nodes (and terminals), which minimizes the total communication cost.

3 Related Work

The problem of distributing a set of components with known mutual interaction volumes reduces to the *multiterminal cut* problem [3], when network node pairs are equidistant (or the charging scheme is flat with respect to distance). In [4] the objective is to minimize the total running time of program modules rather than communication. The problems in [4] also reduce to the multiterminal cut problem [3].

A binary program formulation of the problem can be found in [5]. The development and results of a software tool, which implemented the methodology of [5] for networks with fixed and mobile nodes has been described in [6].

Other related work concerns modeling agent mobility and performance. [7] considers the problem of optimally scheduling a single mobile agent that is assigned to perform a certain task in a computer network. The cost to be minimized is the overall response time, which consists of the time spent in the network nodes plus the time spent for the migration between nodes. [8] compares two possible implementations (static vs. mobile) of a particular service component of the TINA Service Architecture, i.e. the User Agent (UA). There are no optimization concepts involved. Mobile component optimization is explored in [9].

The problems discussed in the aforementioned papers are partly related with the popular *file allocation problem*: Individual files are allowed to replicate in order to reduce communication cost by bringing information closer to the programs that access it, but at the expense of increasing update costs. Papers [10, 11] discuss various optimization models for the distribution of files in a computer

network. The costs considered include communication and delay. Additional requirements refer to parallelism, availability and security. The problem of file migration, that is the reorganization of the file allocation scheme, is also discussed in [10]. Reference [12] discusses and compares various file migration and dynamic file allocation problems. Both adaptive and non-adaptive models are discussed for both types of problems. References [13, 14] consider the problem of reallocating a single file. For this purpose a stochastic control problem is formulated. Whereas in [13] the decision for the location of the file is made centrally, in [14] various nodes decide independently. Finally [15] presents an online algorithm for the dynamic replication of a single file.

4 Problem formulation

The core of the general problem is a mathematical problem, which will henceforth be called the *fixed component distribution problem*. This core problem has been presented and solved in papers [5, 6], where several theoretical and practical examples can be found as well (while the preconditions to use this core and its consequences are presented in this paper). To make the present paper as self-contained as possible, we outline the fixed component distribution problem, but we have omitted the examples for obvious reasons. However, a better familiarisation with certain technical aspects requires a reading of the examples.

Problem 2. Given

- a network topology graph $G(V, E)$ (where $V = \{v_1, v_2, , v_n\}$ is the set of nodes and $E = \{e_1, e_2, , e_r\}$ is the set of links),
- a set of components C,
- a collection of N service topology graphs $G(C^k, F^k)$ $(k = 1, \ldots, N)$, where $C^k = \{c_1^k, \ldots\}$ is the set of components of service k ($C^k \subseteq C$) and edges $F = \{f_1^k, f_2^k, , f_{m_k}^k\}$ represent the interaction between components according to service k, for each service a set of labels $\Lambda^k = \{\lambda_1^k, \lambda_2^k, , \lambda_{m_k}^k\}$ (that denote the traffic exchanged between components for each unit of traffic offered by a user to the service triggering component c_1^k,
- a collection of N functions $t^k : V \to \mathcal{R}$ $(k = 1, \ldots, N)$ that describe the total volume of k service demand due to users attached to a node,
- a routing scheme (i.e. a collection of paths $P = \{p_{ij}\}_{i \in V, j \in V}$, where p_{ij} is a path for each pair of nodes $(i, j)$to be used by the traffic exchanged between them) and
- certain link and node capacity constraints

find the allocation of service components to nodes that minimises a given communication cost function.

The linear program is based on initially placing copies of all components in each node. The program variables are the traffic variables $x_{ij}^{k;mn}$, where $x_{ij}^{k;mn}$ is the traffic generated by service k on edge (i, j) between a copy of component

c_m, which has been placed in node v_i, and a copy of component c_n, which has been placed in node v_j.

It is assumed that the communication cost is a (preferably linear) known function of the information traffic volumes $x_{ij} = \sum_{k,m,n} x_{ij}^{k;mn}$, where x_{ij} is the total traffic over link (v_i, v_j). The exact form and validity of this assumption depends on the charging scheme imposed by the network operator and on network protocols.

Linear programming versions of Problem 2, examples and numerical results have been presented in [5, 9]. The technique used in [5] is to place a component copy in each network node and to calculate the traffic served by each component for each service. Components producing zero traffic are finally removed. Note that the problem cannot be decomposed into independent subproblems, one for each service, because of the existence of common components between services. Additional features to the problem may include node setup costs or component installation costs. Such additions make the problem non-linear.

The single component copy version of this problem, i.e. when each component is unique in the network, is not necessarily easy in terms of complexity. This problem is easily proven to be NP-complete. Even a simplified single copy problem with only three equidistant nodes and a single service is NP-complete. Actually it can be easily shown to be equivalent to the *multiterminal cut problem* [3] (as already mentioned in the previous section), which is NP complete. Fortunately, if only two nodes exist, the problem can be solved in polynomial time by using the max-low min-cut theorem and flow maximization techniques. The mobile version of the component placement problem is treated in [9]

5 The component distribution methodology

Effectively a new importand phase in the service design methodology has evolved. After a service has been designed to the point that its components and their mutual interactions are known, an estimation of the traffic generated between components can be performed. Then by using and solving the distribution problem for sets of services with common components, components can optimally be assigned to nodes and possibly to terminals, if the latter are also in the design phase or if they are reconfigurable.

6 Factors determining component distribution

Problem 2 formulation is enlightening in the sense that it can reveal the effect of various factors on component distribution. We take up these factors one by one in the following few paragraphs.

Service demand distribution: The solution to Problem 2 is partly based on the estimation of service demand. Service demand is created at specific nodes or (groups of) terminals. Demand can follow a quite complex pattern; consequently the traffic source related literature is extensive. If the demand volume is modeled as a random process, the problem of finding the appropriate distribution for

such a process is largely open. However, in fixed network topology and capacity problems demand usually assumes the form of an average over a suitable time period, or it is a compromise between an average a peak traffic volume. In reconfigurable networks, which are by design able to respond to quasi-static traffic, i.e. traffic, which has different statistical properties in different time intervals, but its characteristics remain the same within each interval, the component distribution problem can also be formulated in a piecewise manner. In this case, the designer should cater for the transition between successive configurations.

In general, service components are "pulled" by a node with a high volume of service demand in an effort to reduce communication cost. If the capacity of such a node is not a limiting factor, if the node, component installation and maintenance costs are negligible, and if all components are movable (i.e. they are not assigned to particular nodes for particular reasons), there is a trivial solution to the distribution problem: All service components are likely to be copied to this node.

Is service demand and its distribution predictable? In general the answer is negative, and from time to time totally unpredictable and revolutionary changes may happen. The emergence of WWW is a typical example. The answer depends on the collection of services, which are likely to be offered, on customer profiles and distribution. In certain networks, e.g. in cellular mobile networks, basic services remain constant or at least predictable for a significant period of time. Often network operators try to contain change by hindering the spreading of certain technologies and services: A typical example is voice over IP. The quasi-stability assumption remains valid in special purpose networks, e.g. in the private network of a company. Internet type networks are less predictable. Obviously, the answer to the lack of totally predictable traffic patterns is the creation of reconfigurable networks. From time to time an operator should run the component distribution algorithm again and adjust the configuration of the network and its components.

Network topology and charging: The objective function of Problem 2 expresses the total communication cost in terms of traffic variables. It may contain a sum of terms of the form $d_{ij} \times x_{ij}$, where d_{ij} is the distance between nodes i and j if the communication cost is a linear function of the distance. In general d_{ij} is influenced by the physical distance between nodes, but it may also reflect the operator's charging policy. Flat charging, which depends only on volume, can be modeled by setting all distances to the same value. A quantized charging scheme will usually require a non-linear formulation. Flat charging is likely to favor component concentration in a smaller number of nodes, which offer cheaper processing and storage. Flat charging has become popular with the success of the Internet, but it is becoming increasingly attractive even in the PSTN.

To some extent charging, and the subsequent accuracy of the objective function, will also depend on packet lengths, retransmission protocols, failure recovery protocols, error rates, compression algorithms, and any other factor, which contributes to an increased number of transmitted bits for the same number of

information (i.e. service generated) bits. However, for the purposes of component distribution the influence of these factors can be seen as a second order effect.

Processing and storage capacity: In general the larger the capacity of a node, the more components it can host. Since communication cost is paid only for the interaction of pairs of components, which reside in machines separated by a physical distance, the total elimination of communication cost is ideally achieved by putting all components in the same machine, or at least in machines accommodated under the same roof. An extreme version of this idea has been realized in *computer farms*. This would be the case if users could somehow be transported in negligible time to the farms and use the services there. However, distance in our world is a factor, and a major mission of a network is to make up for distance. In other words, certain service components, which at least include a user interface, must be in the user's premises. Exactly these components pull other components towards them if the user equipment has host them in a cost effective manner. If communication becomes cheaper and cheaper, component concentration is favored again. There are some notable "exceptions": For example, a processing intensive problem can be solved by a large number of geographically separated machines, if installation and processing are also cheap. This is the SETI (Search for ExtraTerrestrial Intelligence) case and the security related decomposition of a long integer to a product of prime numbers. In both cases large numbers of machines have been volunteered by their owners, while low communication volumes and cheap Internet prices have made the communication cost negligible.

Channel capacities: Capacities have an indirect influence on the distribution problem, as they may have an impact on charges. However, a capacity constraint directly delimits the traffic, which is allowed to pass through a link. A cheap link tends to absorb more traffic. When the link is saturated, traffic is diverted to the next cheapest path.

7 Service internal traffic estimation

In Problem 2 for each service k a graph $G(C^k, F^k)$ has been defined, together with a set of labels on its edges. A label λ_{ij}^k on an edge aiming from component i to component j denotes the traffic generated on this edge for data sent from i to j for a unit traffic from a service user to the service interface component. The existence of this graph is based on the assumption that the traffic generated between pairs of service components can be evaluated or estimated. The rest of this section is devoted to the description of techniques, which allow for the estimation of service internal traffic.

The obvious solution in internal traffic estimation is to monitor the traffic between pairs of components for a certain test period. While this approach is conceptually simple and the test can run on a single machine, it requires one or more monitoring components, depending on the testing architecture. If service execution varies and depends on user input, a statistically stable result must be pursued by letting different users interact with the service interface. Also, this approach does not solve the problem of accurately determining the actual

volume of the traffic generated between two components, due to the omission of the network protocols. A more realistic approach would be to install the components in different machines, which are separated by a network similar to the target network.

Simulation is another interesting choice. Occasionally the required programing effort may be close to the service development effort. Modern popular simulation packages, like OPNET and NS2, are capable of capturing the required protocols. However, they will require extra effort in modeling the service components. The problem of the statistical stability of the results and the associated user behavior model must be taken care of by the simulation designer.

A relatively recent approach is to use formal specification techniques like SDL or UML. This approach is suitable only if the service creation process contains a formal definition phase.

7.1 SDL based estimation

The existence of formal specification techniques determine certain features of a service before its actual implementation. So far formal specification has been reserved for the functional characteristics of services. In this section a non functional usage is introduced.

SDL tools are capable of producing a simulation run, but usually this will not entail network protocol details. Also, the specification may not have modeled all those aspects of the service, which are necessary for accurate traffic estimation purposes. For example, it may not use the actual messages, which are used in communication between service components. The statistical stability of the results is also a problem.

SDL trains of events can altermatively be derived from Message Sequence Charts (MSC). MSCs do not offer any particular advantage over simulation, nevertheless they might become available before the existence of an SDL specification (which may or may not appear), e.g. as a part of a preliminary definition of the service. In this sense a collection of representative MSC (or even informal event and message exchange sequences) may be a lazy or a hurried developer's last refuge.

The observation that real messages may differ from their representation in a model, holds for MSCs as well. While the MSC user may decide to make do with the virtual message names or even with a simple message count, certain events are definitely different than simple messages in terms of volume and must be treated accordingly. Such events are packet transmissions and file tranfers in general. Note that such events are of major importance in multimedia services.

7.2 UML based estimation

UML being a standard notation for formal analysis and design of software system, offers several diagrams for separating concerns of different system views, and arguably this approach makes feasible to derive early performance models by taking into account combined data from these diagrams. In UML, a use

case diagram (UCD) provides a functional description of a system, by means of its major use cases and its external users or actors. Sequence diagrams (SD) depict a number of software components and the messages that are exchanged between them in a given scenario (generally a single use case can be described by a set of scenarios, i.e., a set of sequence diagrams).Thus, sequence diagrams provide specific information about the order in which events occur and the interactions required for each event. Consequently, estimates on component traffic generation can be obtained similar to to SDL-MSC based estimates that were treated in the previous section. A deployment diagram (DD) is a graph of computing nodes connected by communication links. Nodes may contain component instances (indicating that the component lives on the node) so it shows the mapping of components on processing nodes. It is apparent that, for example, SDs alone can directly support the traffic conscious specifications of service systems, since they depict in a straightforward manner the timed sequence and generated traffic of service events. Also, DD where the mapping of software components to hardware nodes is described can support the modeling of service distribution. Nevertheless, the level of modeling detail acquired does not directly stem from the set of diagrams and their refined semantics and constructs adopted to describe the hardware/software system; it rather depends on the depth of system knowledge and the designers intuition. Besides, extracting combined information from other UML diagrams would be helpful in order to keep into a performance model relevant characteristics of the system that are not explicitly captured from the SD and DD diagrams being considered. In [16] a comprehensive survey on using UML diagrams for performance modeling and a particular performance estimation algorithm are given.

8 Conclusions

In this paper ample evidence and methodological directions are given in support of service design activities, which will deal with component distribution in such a way, as to minimize communication cost. Communication cost has been considered the main factor of component distribution in this work. Other important and indicative factors, as security and reliability, have not been considered, and can be explored in future research on component distribution. The overall design philosophy should tend to an integrated and QoS aware service-oriented design methodology that explicitly incorporates non functional requirements modeling such as cost, performance, security, availability and reliability in the established service analysis and design methodology phases.

References

1. Papazoglou M.P. et al. “Service-Oriented Computing”, *Communications of the ACM*, Vol.46, No3, pp. 25-28, October 2003.
2. Yian Yang. “Web Service Componentization”, *Communications of the ACM*, Vol.46, No3, pp. 35-40,October 2003.

3. E. Dahlhaus et al. "The Complexity of Multiterminal Cuts", *SIAM J. Comput.*, Vol. 23 No. 4, pp. 864-894, August 1994.
4. H. S. Stone, " Multiprocessor Scheduling with the Aid of Network Flow Algorithms", IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, Jan. 1977, pp. 85-93.
5. M. Anagnostou, "Optimal Distribution of Service Components", *Lecture Notes in Comp. Science*, No. 1430, pp. 17-30, Springer, 1998.
6. M.Anagnostou, A.Rouskas, S.Trigila, "The DOLMEN Component Distribution Methodology and Tool", *IS&N Book 1999: On the Way to the Information Society - 5 Years of European ACTS IS&N Research*, IOS Press, Amsterdam, Netherlands, 2000.
7. K. Moizumi and G. Cybenko, "The Travelling Agent Problem", *Mathematics of Control, Signals and Systems*, Jan. 1998.
8. A. Kuepper and A. S. Park, "Stationary vs. Mobile User Agents in Future Mobile Telecommunication Networks", *Proceedings of the Second International Workshop MA '98*, Stuttgart, Germany, Sept. 1998, pp. 112-123.
9. I. Avramopoulos , M. Anagnostou , "Optimal Component Configuration and Component Routing," *IEEE Transactions on Mobile Computing*, Vol. 1, No. 4, pp. 303-312, 2002.
10. B. W. Wah, "File Placement on Distributed Computer Systems", *IEEE Computer Magazine*, Jan. 1984, pp. 23-30.
11. L. W. Dowdy and D. V. Foster, "Comparative Models of the File Assignment Problem", *Computing Surveys*, Vol. 14, No. 2, June 1982, pp. 287-313.
12. B. Gavish and O. R. L. Sheng, "Dynamic File Migration in Distributed Computer Systems", *Communications of the ACM*, Vol.33, No. 2, Feb. 1990, pp. 177-189.
13. A. Segall, "Dynamic File Assignment in a Computer Network", *IEEE Transactions on Automatic Control*, Vol. AC-21, No. 2, April 1976, pp. 161-173.
14. A. Segall and N. R. Sandell, "Dynamic File Assignment in a Computer Network - Part 2: Decentralized Control", *IEEE Transactions on Automatic Control*, Vol. AC-24, No.5, Oct. 1979, pp. 709-715.
15. O. Wolfson et al., "An Adaptive Data copytion Algorithm", *ACM Transactions on Database Systems*, Vol. 22, No. 2, June 1997, pp. 255-314.
16. Cortelessa V. et al, "PRIMA-UML: a performance validation incremental methodology on early UML diagrams", *Science of Computer Programming* No. 44, pp 101-129, 2002.