Large-scale Performance Evaluation of e-Homecare Architectures Using the WS-NS Simulator

S. Van Hoecke1,2; B. Volckaert1; B. Dhoedt1; F. De Turck1

1Department of Information Technology – IBCN – IBBT, Ghent University, Gent, Belgium; 2Electronics and Information Technology Lab, University College West-Flanders, Ghent University Association, Kortrijk, Belgium

Keywords
Home care services, simulation, performance, optimization, medical informatics

Summary
Background: E-homecare creates opportunities to provide care faster, at lower cost and higher levels of convenience for patients. As e-homecare services are time-critical, stringent requirements are imposed in terms of total response time and reliability, this way requiring a characterization of their network load and usage behavior. However, it is usually hard to build testbeds on a realistic scale in order to evaluate large-scale e-homecare applications.

Objective: This paper describes the design and evaluation of the Network Simulator for Web Services (WS-NS), an NS2-based simulator capable of accurately modeling service-oriented architectures that can be used to evaluate the performance of e-homecare architectures.

Methods: WS-NS is applied to the Coplintho e-homecare use case, based on the results of the field trial prototype which targeted diabetes and multiple sclerosis patients. Network-unaware and network-aware service selection algorithms are presented and their performance is tested.

Results: The results show that when selecting a service to execute the request, suboptimal decisions can be made when selection is solely based on the service's properties and status. Taking into account the network links interconnecting the services leads to better selection strategies. Based on the results, the e-homecare broker design is optimized from a centralized design to a hierarchical region-based design, resulting in an important decrease of average response times.

Conclusions: The WS-NS simulator can be used to analyze the load and response times of large-scale e-homecare architectures. An optimization of the e-homecare architecture of the Coplintho project resulted in optimized network overhead and more than 45% lower response times.

Methods Inf Med 2010; 49: 408–419
doi: 10.3414/ME09-01-0078
received: September 1, 2009
accepted: June 7, 2010
prepublished: October 6, 2010

1. Introduction

The world’s population is aging, introducing new challenges. Since chronic diseases primarily affect older adults, the prevalence of chronic disease increases with an aging population. This aging population and a shift in the burden of illness from acute to chronic conditions drive up health costs and create a generation of people living with long-term illness and disability.

Because of growing financial pressures in healthcare and a shift in pathologies, patients are discharged earlier and care is organized at home. However, patients are only permitted to go home if the home is fitted with the appropriate technology and offer of care. Therefore e-homecare services are required that support medical purposes, care and patient integration in society.

As patients often have different care providers at the same time, a need arises for improved co-operation and integration between the different actors (e.g. patient, physician, specialist, homecare nurse, caregivers). Since the homecare field evolved to a multidisciplinary process involving more and more information of a complex nature, there is a need to provide the adequate information to the care provider at the point and time of need.

At the same time information and communication technologies are opening up to patients and homecare [1–4]. E-homecare encompasses services that are on the borderline between homecare and information technology and creates opportunities to manage and provide care faster, at lower cost and higher levels of convenience for the patients. However, since e-homecare services are built by different vendors, integration of e-homecare services is complex. Consequently, despite proven benefits, e-homecare is not yet widely used in real-life situations. Therefore, within the IBBT-Coplintho project [5], an integrated ICT environment to support the care of patients in their home environment was designed. Through this e-homecare architecture, all actors involved in the care process can set up the necessary interactions, active or passive, covering data, voice and video communication. These interaction channels need to be easy to set up, secure, user-friendly and need to support the adequate communication at the point and the time of need. The actors that were taken into account are the patient and his family and friends, and the overall care team (such as
nurses, general practitioners, hospitals, and alarm centers). Integration of e-homecare services is a complex task since these services and applications are built by different vendors, using different programming languages, data definitions and exchange standards. To enable this integration, a centralized Web service broker [6, 8] and a dynamic client environment for e-homecare (Fig. 1) were implemented using Web service technology [7] and Eclipse RCP [9], respectively. After authentication, the users (e.g., patients, nurses) are presented with a personalized list of e-homecare services. Amongst other, services such as e-scheduling, telemonitoring care pictures and emergency call are provided, illustrating the advantages of the broker architecture, enabling service integration and composition.

E-scheduling enables all these care providers and patients to schedule their visit appointments online at any time of the day or night. The e-scheduling application enables users to make appointments very flexibly and spend less time on routine scheduling tasks. Based on the user profile, a different set of schedules will be visible to consult or edit. As an add-on, integration with a planning service is enabled. This way, the most optimal schedule for the nurses is planned, taking into account their working hours, route start and ending points, patient’s preferences and urgency. The nurses then receive the schedule and the corresponding patients are alerted of the visiting time slot of the nurse. If however, due to unexpected circumstances, the schedule of the nurse changes, a new schedule is calculated and patients get notified of the changes. Patients thus no longer burden the nurse call center with questions concerning the planning of care visits.

Using telemonitoring, as an example, patients with diabetes could input their daily blood glucose level for analysis. Or, rather than relying on the manual entry of data, a range of devices can be connected via infrared or Bluetooth. This way, both the patient and a relevant healthcare professional can be alerted in case of concern; an appointment can be booked online (e-scheduling) if necessary or an ambulatory care provider can be sent automatically.

The care pictures service allows for patients or care providers to easily upload care pictures such as diabetes wound photos of foot or leg injuries in order to view the evolution of the wound over time and to consult a healthcare professional when needed.

Finally, the emergency call service is a personal alarm, which is available 24 hours a day, 7 days a week. When an emergency occurs, the patient can activate the alarm and the platform alerts the appropriate care providers (family, general practitioner or urgency care providers) based on the type of alarm.

Faced with many e-homecare services with similar functionality, service selection can grow very complex. It is no longer just a question of finding a service which can meet the functionality; non-functional Quality of Service (QoS) informations, such as service response time and cost, are also important. Currently, most techniques for selecting services are still determined at design time [10, 11], but initiatives already exist for dynamic selection at runtime [13, 14]. Amongst other initiatives, the e-homecare broker supports dynamic service selection to solve challenges such as services becoming unavailable due to service or network issues, or multiple services offering the same or similar functionality requiring to swap transparently to another service when this service better suits the client’s needs. From the user point-of-view, patients and nurses only interact with the user-friendly client application, abstracting the technical middleware and shielding users from the workflow complexity. As illustrated in Figure 2, when the user invokes an e-homecare service, the broker platform transparently selects one or more services for fulfilling the user request and aggregates the results for the client. By outsourcing requests to the broker, clients do not have to implement selection mechanisms, composition and failover themselves. This way, functionality that is difficult to achieve client-side is shifted towards the broker.

Due to the diversity of e-homecare use cases, provider characteristics, client request patterns and the effect of attrition during e-health experiments [12], testlab evaluations and field trial implementations require a lot of effort. They are not flexible enough to obtain a detailed evaluation of service selection effectiveness, leaving the performance of adopted service selection strategies somewhat speculative [8]. Only large-scale deployment can provide the true test for the broker performance and the dynamic selection of services.

Since building large testbeds is costly and time-consuming, simulation is an
important aid in the evaluation of service selection algorithms. Using simulation, realistic scenarios of network operation in complex e-homecare architectures can be analyzed, whereas analytical techniques to calculate the network flows are only adequate and effective for small topologies [13]. In addition, simulators allow for repeatable experiments under controlled circumstances. By feeding simulators with real data, simulation can be used to design new architectures and algorithms in order to explore potential effectiveness compared to existing ones.

Therefore, in this paper, we have designed and implemented a Web service simulation framework, called Network Simulator for Web Services (WS-NS), capable of evaluating different topologies of service-oriented architectures (SOAs). To the authors’ knowledge, no such simulation framework for evaluating Web service architectures and selection algorithms has been reported upon yet. Therefore a Network Simulator for Web Services (WS-NS) was designed and implemented that accurately models network traffic for service-oriented architectures on the TCP/IP level. To this end, the simulator was built on top of the widely used and well-established Network Simulator 2 (NS-2) [22]. NS-2 provides for realistic and accurate models of network links and protocols, but does not provide options to configure message payloads, nor to model service-oriented architecture (SOA) building blocks like clients, service providers, registries and brokers. WS-NS features these different resource type models and allows applying different broker service selection algorithms.

As can be seen from Figure 3, WS-NS provides a dual layer mixed Tcl/C++ interface and network packet-level simulation. The Tcl front-end gives the possibility to rapidly create new simulation scenarios by building testbeds on a realistic scale in order to evaluate the performance of large-scale e-homecare architectures.

In these cases, simulation is an important aid as analytical techniques are only adequate and effective to calculate the network flows for small topologies [15]. However, no simulation framework for evaluating Web service architectures and selection algorithms has been reported upon yet. Therefore a Network Simulator for Web Services (WS-NS) was designed and implemented that accurately models network traffic for service-oriented architectures on the TCP/IP level. To this end, the simulator was built on top of the widely used and well-established Network Simulator 2 (NS-2) [22]. NS-2 provides for realistic and accurate models of network links and protocols, but does not provide options to configure message payloads, nor to model service-oriented architecture (SOA) building blocks like clients, service providers, registries and brokers. WS-NS features these different resource type models and allows applying different broker service selection algorithms.

As can be seen from Figure 3, WS-NS provides a dual layer mixed Tcl/C++ interface and network packet-level simulation. The Tcl front-end gives the possibility to rapidly create new simulation scenarios by building testbeds on a realistic scale in order to evaluate the performance of large-scale e-homecare architectures.

In these cases, simulation is an important aid as analytical techniques are only adequate and effective to calculate the network flows for small topologies [15]. However, no simulation framework for evaluating Web service architectures and selection algorithms has been reported upon yet. Therefore a Network Simulator for Web Services (WS-NS) was designed and implemented that accurately models network traffic for service-oriented architectures on the TCP/IP level. To this end, the simulator was built on top of the widely used and well-established Network Simulator 2 (NS-2) [22]. NS-2 provides for realistic and accurate models of network links and protocols, but does not provide options to configure message payloads, nor to model service-oriented architecture (SOA) building blocks like clients, service providers, registries and brokers. WS-NS features these different resource type models and allows applying different broker service selection algorithms.

As can be seen from Figure 3, WS-NS provides a dual layer mixed Tcl/C++ interface and network packet-level simulation. The Tcl front-end gives the possibility to rapidly create new simulation scenarios by building testbeds on a realistic scale in order to evaluate the performance of large-scale e-homecare architectures.

In these cases, simulation is an important aid as analytical techniques are only adequate and effective to calculate the network flows for small topologies [15]. However, no simulation framework for evaluating Web service architectures and selection algorithms has been reported upon yet. Therefore a Network Simulator for Web Services (WS-NS) was designed and implemented that accurately models network traffic for service-oriented architectures on the TCP/IP level. To this end, the simulator was built on top of the widely used and well-established Network Simulator 2 (NS-2) [22]. NS-2 provides for realistic and accurate models of network links and protocols, but does not provide options to configure message payloads, nor to model service-oriented architecture (SOA) building blocks like clients, service providers, registries and brokers. WS-NS features these different resource type models and allows applying different broker service selection algorithms.

As can be seen from Figure 3, WS-NS provides a dual layer mixed Tcl/C++ interface and network packet-level simulation. The Tcl front-end gives the possibility to rapidly create new simulation scenarios by building testbeds on a realistic scale in order to evaluate the performance of large-scale e-homecare architectures.
defining a Web service broker supporting different selection algorithms, multiple service providers, one or more registries, multiple clients and a network topology to interconnect everything. This way, the Tcl front-end offers a high-level abstraction of the simulator entities’ implementations. The characteristics of the clients and service providers can either be configured manually in order to model complex and diverse service request patterns, or random provider characteristics and client request patterns can be generated that are uniformly distributed or follow a Poisson process.

Each component of the simulator (broker, registry, clients, service providers) is associated with a single NS-2 node. These components can be seen as models for real-life components running on the nodes they are hosted by, as the communication between components is a source of network traffic in the simulation. Messages and RPC calls exchanged between components are implemented as XML messages that can be easily transported between the C++ and Tcl layers. Objects can store properties/status information and provide methods for reading/writing from/to XML. Web service requests are commands sent over the simulated network links between source and destination. Once the requests have been received at the destination, the XML commands are deserialized and delegated to the appropriate C++ method for executing the command, this way providing the glue between the Tcl and C++ layers. As messages are encapsulated in XML structures, payload was also added to service requests to simulate and incorporate the SOAP-specific overhead of Web service messages, resulting in larger network transmission times. The extra time needed for
(un)marshalling the messages is also simulated by enlarging the server processing time according to the performance measurements of [24].

Finally, service requests can only be executed on service providers offering the requested service. In order to do so, a different set of Web services can be provided on each provider letting the broker choose which Web service he wants to use. As Web service workflows are statically composed out of different services which are each dynamically selected during the workflow execution, request dependency is required in WS-NS and directed acyclic graphs requests were implemented. A Web service composition is a collection of service requests with precedence relations existing between the service requests. Service requests can only be executed if no precedence relations are violated.

When the simulation has ended, WS-NS provides file output regarding service request lifetime (such as arrival time, launch time, end time, execution speed, selected service provider and network delay). The WS-NS output parser parses this XML output file to comma-separated value files for easy use in spreadsheet programs.

Figure 4 presents the simulation flow of WS-NS. As can be seen on the figure, simulating a Web service scenario using WS-NS roughly consists of the following steps:

First, the SOA network topology is constructed using NS-2 node and link objects. Next, WS-NS service providers are instantiated (and their configuration is parsed from XML) and associated with the appropriate NS-2 nodes. After service providers have been instantiated, Web service middleware components are constructed and configured and the broker selection algorithm is chosen. This selection algorithm can be either network-unaware or network-aware. At least one broker and one registry are required. WS-NS service providers then register themselves with the registry/registries. In a last step, clients are instantiated and configured for submitting requests to the broker. The resulting Tcl file is used as input for the WS-NS simulation. During simulation, the broker logs all of the resource allocations made. From these logs, it is possible to extract service execution times and provider utilization using the WS-NS output parser.

The time it takes for a service request to fulfill since it has been requested by the client is presented in Figure 5 and can be divided in:

- sending the service request to the broker (network delay client-broker);
- time spent in the broker’s queue and needed for the selection of appropriate providers (brokering);
- transfer time to the service providers (network delay broker-providers);
- time needed to execute the service (service execution); and
- transfer time for the service response.

The network packet-level simulation gives WS-NS a unique position compared to competitors when it comes to modeling Web service selection algorithms in realistic and diverse topologies. Moreover, the simulator covers a wide range of application cases and is not limited to evaluating the impact of service selection algorithms in e-homecare. The simulator can also be used for offline tuning of load balancing, service selection and brokering strategies in any e-health area. Example use cases are the determination of ideal locations in the network for broker and server positioning in order to roll-out health-enabling and ambient-assistive technologies on a large scale [26] or the implementation of heart rate-based measures for outpatient cardiac rehabilitation [4]. The presented simulator can also be used to evaluate service-oriented architectures in other areas outside e-health, but is especially useful to model and evaluate large-scale e-health architectures as these architectures are complex and integrate different services.

Figure 5 Steps of network-(un)aware service selection. The dashed steps querying the connection manager only occur for network aware service selection.
providers, fulfilling diverse user needs (e.g., patients, physicians, nurses, care providers) and different quality constraints (e.g., high-priority alarm messages, low-priority video-chat with family).

As a result, the simulation environment WS-NS offers the following major features:
- Large-scale evaluation of service-oriented healthcare architectures whereas field trials are limited to small-scale deployments and evaluations;
- Detailed service-oriented architecture (SOA) resource models, in particular models for Web service brokers, providers, registries, and network elements;
- Generic and flexible Web service model supporting Web service compositions and workflows;
- Possibility of accurate packet-level network simulation;
- Easy simulation scripting through high-level Tcl scripting language;
- Both network-unaware and network-aware Web service selection algorithms are built-in; new algorithms can easily be plugged in;
- And finally, extensibility in order to support evaluation of other Web service aspects as well, such as security and reliability. Note that in that case, both the payload and the server processing time have to be enlarged according to performance measurements.

3. Simulation Implementation

As all characteristics of the clients, service providers, and interconnecting network can be configured in WS-NS, the performance, scalability, and network bottlenecks of service-oriented architectures can be evaluated. Based on the results, the network topology, server positions, or selection algorithms can be adjusted. When needed, service providers can be added or e-health services can be duplicated. In order to demonstrate the capabilities of the developed WS-NS simulator, an extrapolation of a previously evaluated e-homecare field trial from the Coplintho project [30] is simulated. During the field trial, a prototype of the broker, as well as some sample e-homecare services (e-scheduling, telemonitoring server), were implemented. The prototype was then easily extended by other partners, providing additional e-homecare services such as audio-diary, video-chat, telemonitoring client, and care picture service. In order to refine the focus of the field trial, two complementary patient groups or pathologies were chosen. Based on the description of different pathologies and their prevalence in the homecare field, diabetes and multiple sclerosis were chosen as a target group for the demonstrator. Several healthcare providers from different institutions are involved in the care process of diabetes patients, respectively multiple sclerosis patients, making it an ideal case for demonstrating integration and multidisciplinarity.

The domain analysis and small-scale field trial within this project highlighted the need for integrated e-homecare services across healthcare providers, as well as the time-criticalness of e-homecare services and the need for large-scale evaluation to test the broker performance and the dynamic selection of services.

To overcome the limitations of the field trial, which was limited to three service providers and five patient users due to practical reasons, WS-NS is used to extrapolate the evaluation of the broker platform and dynamic service selection to the large-scale topology described in the next section.

3.1 Network Topology and Parameter Selection

A fixed service-oriented architecture was used for all network-unaware and network-aware simulations presented here. Within the Coplintho project, homecare providers, hospitals, and general practitioners are interconnected. The network topology and parameter selection to illustrate the possibilities of WS-NS is based on the results and project seed data of the Coplintho field trial. Figure 6 shows the network topology. Each city with at least one hospital is represented as a node, resulting in 18 geographically dispersed service sites, interconnected by 24 bidirectional links.

For the evaluation of service selection algorithms, following service distribution and synthetic request load were chosen:
- Each site has an amount of providers in accordance with the number of hospital campuses and care providers available in this site.
- The providers all can handle ten simultaneous user requests, which equals the throughput of the servers used in the field trial. Each request retrieves a fixed fraction of the service provider’s processor.
- To reflect the difference in service provider properties, not all processing capabilities of the service providers are chosen identically: the least powerful service providers have processors operating at the reference speed and service execution prices randomly chosen between 0 and 10 cost units. A second class of service providers operate at twice the speed of the least powerful providers and have service execution prices randomly chosen between 5 and 15 cost units. The third – and last – provider type operates at three times the reference speed and has prices randomly chosen between 10 and 20 cost units. The least powerful type of provider is cheap and three times as common as the most powerful, expensive one, and twice as common as the middle one. This way, on average, faster services should have higher prices but exceptions can exist due to overlap between the price intervals.
- In order to model the diversity of e-homecare applications, each of the five...
e-homecare services (tele-monitoring, care pictures, video-chat, e-scheduling and audio-diary) of the field trial was modeled four times with different functionality offerings (e.g. different input/output), resulting in 20 functionally different e-homecare services.

- Each service provider is hosting five e-homecare services, randomly chosen out of the 20 available services. As each provider has its own characteristics (load, cost, bandwidth), combined with a randomly selection of 5 out of 20 functionally different services, the complexity of available e-homecare services available in the network is a first attempt to simulate the diversity of e-homecare service offerings and quality constraints.

- The prevalence of diabetes and multiple sclerosis is 1/20, respectively 1/1000. Based on the results of the field trial, the load is chosen so that each patient in East-Flanders between 35 and 70 years uses around five e-homecare services a day, which is an overestimation compared to the Coplinito results. Based on the data from the research center of the Flemish Government [30], this results in around 33,110 patients out of 649,215 East-Flemings aging 35 to 70. By assuming that all of these patients use five e-homecare services a day, the aggregation of these 165,550 service requests allows to assume that service request interarrival times \( T \) are independent and identically distributed, following a uniform distribution over \([0, T_{\text{max}}]\), with \( T_{\text{max}} \) inverse proportional to the site's amount of providers. This way sites with a large number of hospital campuses and care providers will have a higher request rate than smaller sites, and the total request rate approaches the assumption of diabetes and multiple sclerosis patients using around five e-homecare services a day.

- Size of service requests is chosen so that execution of the request on a reference processor takes 0.5 seconds, which equals the response times measured in the testbed. The requested e-homecare service is randomly chosen out of the 20 available services.

- Three different payloads are tested, ranging from 300 bytes for a SOAP message containing limited text (such as small prescription or alarm message) over 300 kilobytes (for e.g. patient’s medical file or a history of monitoring values) to 3 megabytes for SOAP messages containing diabetes wound pictures.

As a result, all parameters for the simulation implementation are either direct representations of the Coplinito field-test values, or randomly chosen in order to eliminate their effect when averaging over large amounts of simulated service requests.

### 3.2 Service Selection Algorithms

Network-unaware service selection assumes that residual bandwidth on network links is sufficient and will select services based on the status of the service providers. Algorithms that use this kind of approach will not take into account information concerning the status of the network interconnecting these service providers. The decision which service to select for a request will be based on the information acquired from the repository.

The different network-unaware algorithms provided in WS-NS are:

1. **Fastest Service Selection (FSS)** which attempts to optimize execution time by minimizing the time a service spends on processing;
2. **Minimum Cost Service Selection (MCSS)** which attempts to minimize the cost of the selected service based on the role/profile of the client;
3. **Highest Quality Service Selection (HQSS)** which selects, for each service, the candidate that maximizes the quality score compared to other candidates by applying a Multiple Criteria Decision Making (MCDM) technique. The quality score is a weighted sum of several performance and cost parameters such as execution cost and execution duration. Fastest Service Selection and Minimum Cost Service Selection can be seen as special cases of this Highest Quality Service Selection where the weight for all the parameters except response time, respectively cost, is null.

Once the algorithm has selected the provider that has the best service offering, it executes the service request on that provider. The virtual load information in the simulator is adjusted accordingly.

Network-unaware algorithms do not take into account network topology and estimate service round-trip times (for example within FSS and HQSS balancing algorithms) purely on the execution time at the service provider. Network-aware selection algorithms, on the other hand, are able to estimate the service round-trip time and end time more accurately, taking into account the networking delay that is suffered by the individual SOAP request/response messages. In order to do so, a connection manager is used within WS-NS. Network-aware service selection algorithms will not only query the registry for service providers that adhere to the request’s requirements, but will also query the connection manager for information about the status of the network links interconnecting these resources. The connection manager will then update the broker with information about connections that can be set up to service providers. Based on this information, the broker selection algorithm is able to calculate service execution time and end time more accurately. This way, by using network-aware service selection, the network delay is taken into account, allowing for more accurate and correct service selections.

The different network-aware algorithms provided in WS-NS are:

1. **Network Fastest Service Selection (NFSS)** which is an extension of the network-unaware algorithm where the fastest service is chosen, independently of the network delay. Network Fastest Service Selection attempts to optimize the total round-trip time of a service request by minimizing the total of network time and time spent on processing;
2. **Network Highest Quality Service Selection (NHQSS)** which makes a trade-off between total roundtrip time and cost;
3. **Minimum Hop Service Selection (MHSS)** which attempts to minimize the amount of network links (hops) the request/response needs to be sent over for a request in order to sparingly use available network resources;
4. **Prefer Local Service Selection (PLSS)** which attempts to execute a service re-
request on local resources as it can be assumed that remote service providers are only used when necessary. Three reasons can make local executing impossible: the required service is not available locally, the requirements cannot be met locally, or the maximum load locally has been reached. In this case, the broker looks at the status of remote service providers and, if possible, selects the service provider meeting the service request's requirements. When selecting a remote service, the broker can use different selection algorithms in order to select that provider with the earliest service end time, or maximum quality score. The end time can also be calculated in either a network-unaware or a network-aware fashion.

4. Results

Figure 7 presents the average response time for the different service selection algorithms, both network-unaware and network-aware, for varying payload and bandwidth. The average cost for the different network-unaware and network-aware selection algorithms is presented in Figure 8.

As can be seen in Figure 8, the cost does not change over the different tests for network-unaware service selection. This is because the algorithms do not take the network into account, and the same request load is used for all test cases. This way, the tests with variable network bandwidth and message size will select the same result set of services, resulting in the same average cost. Contrary to the cost, the average response time does depend on the network overhead and does differ for varying bandwidth and payload, even in case of network-unaware service selection.

By comparing the results, one can see that the Minimum Cost Service Selection (MCSS) succeeds in finding the services with lowest cost, sacrificing service execution time. Fastest Service Selection (FSS) finds the service with lowest service execution time, not taking into account the price. Highest Quality Service Selection (HQSS) maximizes the quality score, this way finding services almost as fast as those with Fastest Service Selection but at a lower cost.

Taking into account the network results in a more accurate and correct selection of the fastest service as the network overhead is also minimized. Like for network-unaware service selection, Network Highest Quality Service Selection makes a trade-off between cost and total round-trip time and succeeds in finding services almost as fast as with Network Fastest Service Selection, but at a slightly lower cost. By tuning the weights within the (Network) Highest Quality Service Selection algorithm, a trade-off between cost and round-trip time can again be made.

From an economical viewpoint, one can assume that local services are free and remote services are only used when necessary. As can be seen in Figure 8, assuming local services have zero cost, lower the average cost 5% for Fastest Service Selection,
100% for Minimum Cost Service Selection and 45% for Highest Quality Service Selection. Since Fastest Service Selection does not take price into account when selecting a service, the cost reduction for Fastest Service Selection is solely the result of some selected fastest services being local and thus free. Minimum Cost Service Selection succeeds in finding a service with zero cost for all requests by either selecting a local service or a remote service with zero cost. Highest Quality Service Selection succeeds in finding services almost as fast as the services selected by Fastest Service Selection but at a 45% lower average cost.

When using the Prefer Local Service Selection algorithm, 60% of the requests can be serviced locally in this use case. Assuming local services to be free, as expected, the Prefer Local algorithm reduces the average service cost remarkably (compared to other service selection algorithms). The benefits of Prefer Local Service Selection are, however, cancelled out when SOAP messages are large and the bandwidth is limited. In this case, centralized brokering introduces useless network overhead in the case of locally executed service requests. It is clear that when large messages are sent to the broker and then sent back for local execution, the network overhead would better have been avoided.

The high round-trip times for bandwidths of 1 Mbps, combined with large messages containing wound pictures (3 MB) are caused by network congestion: assuming 33,110 patients using five e-homecare services a day, around 1.9 requests per second are generated. For large messages, each request results in sending 3 MB from the client to the broker, and optionally forwarding this request to a service provider different from the broker site. When the messages are 3 MB and the network has a bandwidth of 1 Mbps, this results in a minimum network time of 24 seconds for each message. Thus, at a request rate of 1.9 requests per second, the network will quickly become congested. In these cases, selecting the fastest service for executing the request makes no sense if network overhead is not taken into account. The simulator reveals the high round-trip times for relatively small bandwidth (1 Mbps or 10 Mbps), combined with large messages containing wound pictures (3 MB). The network packet level simulator allows identifying the bottleneck links causing network congestion. Based on the simulation results, network administrators can take appropriate measures.

When bandwidth is high and messages are small, taking into account the network for selecting the fastest service or service with highest QoS score, has the same results as the network-unaware version since the network overhead times can be neglected compared to the service execution times. Accordingly, the Minimum Hop Service Selection algorithm performs worse than Network Fastest Service Selection when bandwidth is high and messages are small, since the service execution largely exceeds the network time and the fastest services are not necessarily the closest.

However, when bandwidth is limited and messages are large, service execution times and brokering times are negligible compared to the network times, and as a result Network Fastest Service Selection will minimize network overhead by minimiz-
ing the hops and thus Minimum Hop Service Selection can, in that case, be used as a simplified version of the Network Fastest Service Selection algorithm, obtaining the same results.

5. Optimized e-Homecare Architecture

The simulation results showed that, for use cases involving scattered network clusters such as e-homecare, selecting the fastest service for executing the request makes no sense if network overhead is not taken into account for large SOAP messages over limited bandwidth. In that case, network-aware service selection leads to better selection strategies as the service completion time can be estimated more accurately, taking into account the speed at which SOAP request/response messages can be delivered.

Additionally, for the e-homecare use case, the centralized architecture is too restrictive to deal with the issue of network connectivity status and data-intensive service requests. Prefer Local selection algorithms result in unnecessary network overhead since the requests are first sent to the broker and then delegated back to the local service provider for execution.

As a consequence, a more efficient alternative than the centralized e-homecare architecture, presented in [4], would be to only send service localization requests to the broker in order to determine the service endpoint. The endpoint of the selected service can then be sent to the client who can use this endpoint reference and WS-Addressing [31] to call the service. This way the broker can make a decision about the endpoint reference that should be used for that request without having to send the large SOAP messages. Once the services are selected, the client communicates with the service directly without any broker intervention during the actual service process.

However, due to the dynamic nature of Web services, abrupt failure or unavailability of services needs dynamic selection of another equivalent service. The downside to this solution is that the client has to implement dynamic endpoints, aggregation and failover himself which is in contradiction with the e-homecare use case. This solution also leaves composition as the client’s responsibility.

Simulation results show that the e-homecare broker architecture benefits from a region-based design, when compared to a strictly centralized design. As can be seen in Figure 9, in a region-based architecture, service providers are clustered in logical regions. Execution within the local broker’s region is preferred, but if that proves impossible, the request is forwarded, using WS-Addressing, to a broker which resides at a hierarchical higher region and uses highest-quality score to select an appropriate service for executing the request.

When a new service provider is installed in the broker platform, it must register itself and its services to the broker of that region. The services and server capabilities are stored in the registry of the region-broker. In order to support forwarded brokering, hierarchical higher brokers store an overview of all services running in their sub-regions. This information is updated at regular intervals by the brokers of each region. At regular time intervals, an overview of the services is pushed from the region brokers to the hierarchical higher broker. When executing the highest quality score selection algorithm, the broker can request more detailed and up-to-date information about each service from its lower-region brokers.

Since for our e-homecare case 60% of the e-homecare requests can be serviced locally, this results in an important decrease of average service cost, as well as a decrease of average response times when bandwidth is limited and messages are large. Figure 10 illustrates the benefits of the Region Based Service Selection (RBSS) for our e-homecare use case. As can be seen, the Region Based Service Selection optimizes network overhead and results in at least 45% lower response times.

On the downside, implementing hierarchical broker architectures comes at a cost. First, additional brokers need to be added to the network, increasing the hardware, software and maintenance costs. Second, the network needs to be organized in regions which generate additional management overhead. In addition, there is a risk that fragmenting the network in too much regions results in long intra-broker for-

---

**Fig. 9**
Region-based e-homecare broker architecture combining Prefer Local Service Selection (PLSS) and Network Highest Quality Service Selection (NHQSS) at different hierarchical levels.
warding chains. Many parameters make positioning of services, servers and brokers in the network a complex task. Our Web service simulator can be used by network designers to manage e-homecare network complexity.

6. Conclusion

The number of retired citizens and needy elderly increases considerably, increasing costs and requiring homecare to change and provide interaction and cooperation between multiple care providers. Integration of e-homecare services and applications is a complex task since these services and applications are built by different vendors, using different data definitions and exchange standards. E-homecare brokers, based on the Web service technology, become essential to provide advanced interaction and cooperation between these multiple care providers. Since QoS requirements such as price and response time can influence the choice of service, different selection algorithms within the broker can be used.

However, the size of e-homecare platforms and the effect of attrition make it impractical to test the broker performance and selection algorithms in a real testbed since proof-of-concepts and field trials are too small scaled. Therefore the authors developed a Web service simulator, built on top of the widely-used NS-2 network simulator, and capable of accurately modeling network traffic between the different components. The simulator, called WS-NS, allows for repeatable experiments under controlled circumstances for large-scale SOA platforms. As all characteristics of the clients, service providers and interconnecting network can be configured in WS-NS, the performance, scalability and network bottlenecks of every service-oriented architecture can be evaluated in order to tune e-health architectures, position servers and brokers, and optimize selection algorithms.

Network-unaware and network-aware service selection algorithms were presented and their performance was tested. Based on the results, we may conclude that when selecting a service from the available services to execute the request, suboptimal decisions can be made when selection is solely based on the service’s properties and status. Taking into account the network links interconnecting the services can lead to better service selection strategies (especially when dealing with highly data-intensive service requests) and avoid cases where service requests are fulfilled by services which are connected by low-bandwidth connections. Clearly, for low bandwidths, not taking into account the network (when selecting the services) incurs a severe penalty; when bandwidth grows, penalize average response time: requests fulfilled at a slower local service provider are less likely to be executed on a faster one (which is the case when the fastest service provider is selected). Using Prefer Local algorithms with centralized brokering also results in unnecessary network overhead for sending the requests to the broker and delegating them back to the local service provider for executing the request.

Therefore, based on the test results, the e-homecare broker architecture was optimized from a strictly centralized design to a hierarchical region-based design where the service providers are organized in regions according to connection speed. Each region has its own broker preferring local execution. If local execution is not possible, the region-broker forwards the request using WS-Addressing to a hierarchical higher region-broker that uses the network highest quality score for selecting an appropriate service for executing the request. Since in our use case 60% of the e-homecare requests are serviced locally, this results in at least 45% decrease of average response times.

Acknowledgments

This work was partly funded by Transecare. Transecare is a project of IBBT (Interdisciplinary institute for BroadBand Technology). IBBT is a research institute founded by the Flemish Government in 2004. Sofie Van Hoecke would like to thank the IWIT (Institute for the Promotion of Innovation
References


