DISTRIBUTED AND COMPONENT ORIENTED TOOLS FOR COMMUNICATION NETWORKS USING WEB SERVICES

Sébastien Rumley[1], Christian Gaumier[1], Christophe Trefois[1]

[1]Ecole Polytechnique Fédérale de Lausanne (EPFL), Laboratoire de télécommunications, 1015 Lausanne, Station 11, Switzerland

sebastien.rumley@epfl.ch (Sébastien Rumley)

Abstract

Modern communication networks are reaching a high level of complexity. As their global dimensioning and performance evaluation are very hard tasks, a multi-layer approach is generally applied to divide them in smaller problems easier to tackle. Within this approach, tools only address one or two network layers. To perform an enhanced analysis encompassing more layers, several tools, not necessarily available in the same place when the problem arises, are often required. This paper shows how to allow a remote access to various existing tools using Web Services paradigm. In this way, software components modelling network elements, layers, or computing specific functions are turned into Web Services, which are available to remote users over the internet. Accessed through a simplified interface, they can be used sequentially or in parallel to solve a specific task. This dispenses the user to perform a local implementation, allows better code reuse, and offers an easy way to confront results from distinct models. A tool accessed via a web service can be beneficial for either research or educational purposes. Moreover, making tools available on the web increases credibility and visibility of their authors. Calls to Web Service require transmitting and retrieving all input and output data in a specific format. In our approach, the Multilayer Network Description (MND) format is used to describe both input data and computed results. It offers common basic structure while guaranteeing a large extensibility, and thus eases the interactions between related but different tools. The viability of this approach in the context of network planning is illustrated through two examples.

Keywords: Communication networks, Web Service, Component Oriented design, Network planning, Collaborative Work.

Presenting Author's biography

Sébastien Rumley received M.S. degree in telecommunications engineering from the Swiss Federal Institute of Technology of Lausanne (EPFL) in 2005, after undergraduate studies in Lausanne and Zurich (ETH), and graduate studies in Lausanne and Santiago de Chile (PUC). Since 2006, he is working as research assistant at the Telecommunications Laboratory of the EPFL. His current research focuses on mechanisms improving the collaborations between geographically distributed research institutes, in the context of COST actions 285 and 291.

1 Introduction

In the last decades, communication networks have enormously evolved in terms of capacity and ubiquity, but at the price of a large increase of their complexity. Nowadays global and multi-services networks are made of a multitude of composing parts. They are nevertheless operational, and as in a precision watch, each single piece is contributing, interacting with other elements, to achieve the final goal: deliver the information.

Thirty years ago, a description encompassing the whole telephone network was imaginable. In contrast, modern communications networks, due to their immense heterogeneity, are exceptionally difficult to consider entirely [1]. In consequence, a one time update of all their composing parts is too risky and too costly. To keep enhancing their performances, engineers and researchers are studying, modelling and eventually replacing elements separately.

Separation is not always straightforward to achieve as elements are rarely independent and often interacts with others in various possible ways. Tightly-coupled ones exert a permanent mutual influence and cannot be separated. They habitually have to be modelled and analysed jointly. Within loosely-coupled elements on the contrary, the mutual influence is limited. The separation is thus possible to realise, either neglecting the interactions, either reducing to some fixed constraints.

Once isolated, the next step toward element enhancement and eventual replacement is to setup a basic model. As their realisation generally implies several assumptions, models reproduce behaviours with a certain margin. In the early stages of development, basic models are enough accurate to permit valuable enhancements of the modelled parts. At a certain point however, weight of assumptions is becoming too important to permit further enhancements. In particular, as some of the assumptions are due to the negligence of the interactions, they have to be taken into account. This leads to also incorporate the models of elements concerned by these interactions.

Several approaches exist to extend a model to more than one element. One of them consists in successive and ad-hoc integration of neighbouring elements inside the model itself. However, as software grows in size and importance it also grows in complexity [2], it induces penalties in terms of development time, stability and reliability. This approach thus lacks of scalability. Another solution follows a Component Oriented development [3]. In such a design, models are still conceived independently, however their utilisation as components inside a higher level framework is assumed. They thus include interfaces permitting to send and receive signals from other models, what in turn enables to take into account interactions with other elements. The component oriented design offers the possibility to integrate a component without having to implement it. It can be used as building block for another study, as a way to confront results from distinct models of the same element (e.g. analytical and simulation model), as a way to validate other components or models, or for educational purpose [11]. In general, component oriented design favours code reuse and shortens development time [3].

Note that although the models, in their setup and analysis, are main tasks for researcher, they need some input and produce output data. Composition of the first and interpretation of the later may be difficult, especially if made by hand. Component oriented design can also be applied in this sense. Tools offering graphical interfaces (to help data edition and visualisation), providing optimisation procedures [3] or post-processing functions [4] can also be considered as components, connectable with others.

In the context of communication networks, multiple Component Oriented designs already exist. Widespread network modelling or network simulation environment like OPNET [5], OMNeT++ or NS-2 [6] can be considered as component oriented frameworks, as they permit an assembly of various devices within a common simulation [7]. There are several private initiatives leading to frameworks for component integration, as COSMOS [8] or CANPC [9]. Finally, two or more libraries, each dedicated to a particular model, can be connected using a light and ad-hoc framework which simply accesses them sequentially.

This paper proposes another approach based on the Web Services (WS) paradigm. Software components, either modelling an element, either proposing a specific function (from now the general term "tool" will be used to qualify both), are turned into Web Services accessible over the internet through simplified interfaces. The component oriented design is thus extended to a Service Oriented Architecture (SOA) [10]. Within this architecture, at the price of some limitations, the advantages the components are available, combined with others benefits. Access to tools is free. These latter are reachable by a large number of potential users across internet. The provided interfaces are clear and simple. Advantages moreover exist for the provider: a widely available tool will reinforce his credibility (important for researchers), may speed up the test phase (important for commercial entities) while the code remains hidden.

The rest of this paper is structured the following. Section II details the architecture and relates how human actors, others WS and/or other software entities interact through the WS principle. Two examples of tools are given: a wavelength assignment

algorithm to a WDM network topology (Section III) and a tool permitting an easy graphical display of network structures (Section IV). Key aspects of the architecture are summarized in Section V, which also compares it with other component based architectures. Section VI concludes this study.

2 Web Services Based Architecture

Web Services are a client-server based architecture of distributed computing. It can be seen as an evolution of the Remote Procedure Call (RPC), which suits well to a closed-world problem (awareness of the users, resources), but becomes too rigid in an open-world context. Web Services, as WebPages, should be accessible from anywhere at anytime, for anybody, and are based on purpose over mechanisms providing higher flexibility [12], in particular XML for data formatting and HTTP or SMTP [13] for transport.

The term Web Services defines only a concept. Different implementations and even different specifications exist. This paper follows the World Wide Web consortium (W3C) definition: *"a software system designed to support interoperable machine-to-machine interaction over a network, with interface described in a machine-processable format, specifically WSDL*[1]*"* [14].

2.1 Server side mechanisms

To propose a Web Service, a single process listening on a TCP port is sufficient. When a connection occurs, it parses the request, extracts the data contained in the SOAP message and runs its function using the extracted data. It then forms a new message containing the results and sends it back over HTTP (fig. 1). To comply with the WS specifications, a WSDL file describing the service should also be available some where on the internet. Messages are formatted in XML and composed according to the SOAP protocol, also specified in [14].

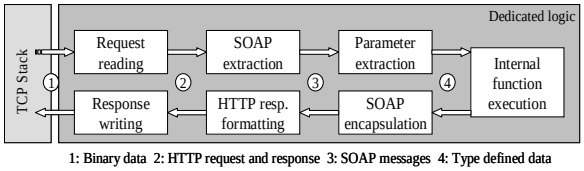

Fig. 1 All-in-one implementation of a Web Service.

This implementation is in general not used, as it implies to implement the full HTTP protocol management inside the same block. Fig. 2 illustrates another design where a Web server engine (apache server or Microsoft's IIS for instance) is used to parse HTTP request and format HTTP responses. The Web Service is associated to a certain URL and SOAP messages are extracted and transmitted to the WS when a request mentions this URL. This option permit to offer the WSDL file associated to the service on the same server, but at a different URL.

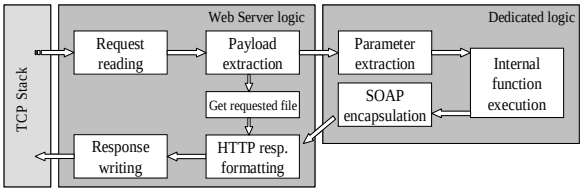

Fig. 2 Implementation using a web server engine to pre-process HTTP request and format HTTP response.

Fig. 3 presents a third evolution where all operations have been decoupled. Web server is still used but only to process HTTP requests and responses. A specialised block is responsible for SOAP messages parsing and composition. Other architectures (regrouping web server and soap processor) are possible.

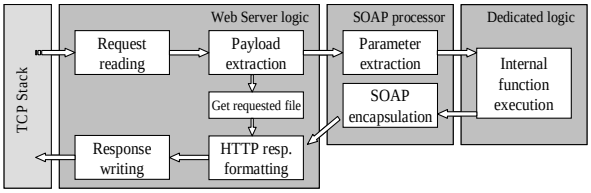

Fig. 3 maximal decoupling between function, using, besides the web server engine, a dedicated SOAP processing engine.

2.2 Client side mechanisms

The client side communication mechanism is in all point similar except that operations are done in a reverse order (starting from type defined parameter and ending with binary serialisation of an HTTP request over TCP). However, before sending any request, the user on the client side should know how to present the input data in order to be understood by the server. He also needs to know how to interpret the data he will receive. Client side should therefore implement an additional mechanism to retrieve this information.

The provider of the Web Service can release some textual information, along with examples of SOAP messages, on his web site. An interested human user may later take inspiration of these examples, replacing the parameters where it is necessary (Fig. 4a). He will then send the obtained SOAP messages (using a simple HTTP client), retrieve the SOAP response and extract the information. He can even write a simple program which automatically replaces the data inside a message SOAP skeleton and extract the response using a regular expression.

Alternatively, Web Services principle includes a much more precise mechanism, which really distinguishes it from RPC. Using the WSDL associated to the Web

[1] WSDL: Web Service Description Language

Service, which completely defines the interface and the structure of the SOAP messages, the client has all the information he needs at disposal. As the WSDL is a machine processable file, the logic responsible for parameter encoding and SOAP messages composition, for the communication with the Web Service and for the decoding of the response can be automatically generated (Fig. 4b).

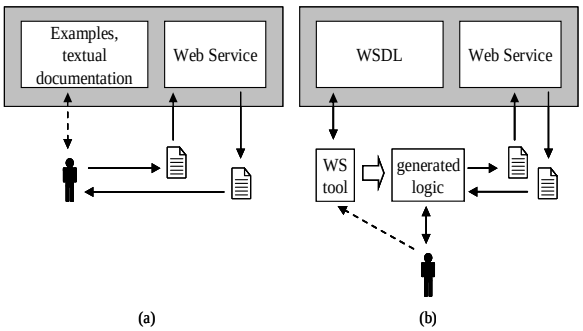

Fig. 4 The manner to access a Web Service can be learned either with examples, either using the WSDL

2.3 Integration in a component oriented design

Only atomic interactions between a user and a Web Service have been described so far. To integrate them inside a component oriented framework, additional principles have to be presented. Attention must be paid to the parameters expected by the WS and to its response. Data contained into the SOAP messages have been presented as “type defined data”. The type used plays however a key importance, as it must be supported by any actor present in the framework (user or WS).

In the context of communication networks, the tools that can be proposed as Web Services need two simultaneous inputs: data representing the addressed problem, and “configuration” data specifying the manner to address it. The result is always related to the addressed problem, except in case of failure of the algorithm (fig. 5).

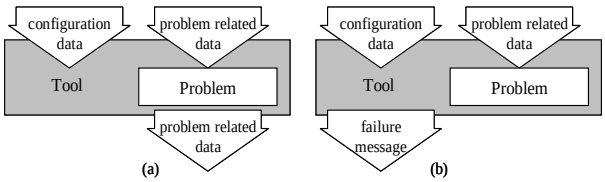

Fig. 5 a) A tool receives the input data for the problem it addresses, plus some configuration parameters. It only returns the output data of the problem. b) If tool fails during its execution, it may return a failure message instead of problem related data.

Each dataset related by a particular problem can be specific. To obtain some coherence between all different sets of data transmitted, the Multilayer Network Description (MND) has been chose as descriptor. MND offers a common basic structure while guaranteeing a large extensibility, and thus eases the interactions between related but different tools [15]. It is also based on XML and fits well inside SOAP messages. MND even permit to incorporate output data with input data inside the same document.

Besides the concrete input of the problem, a structure should also be specified for configuration data. As in this case, the data and the type of data depend largely on the acceded tool (which can be an event driven simulators, a static network planner, etc), is it far less evident to find shared properties that could be the structure base. No valid answer has been found yet, and thus the question is laid open for future conclusions. Until now a minimal structure has been used. Indeed, all configuration parameters have to be given as character strings. Figure 6 summarise the types chosen for input and output data exchange. Note that some tool may need no MND input at all (scenario generators) while others may produce human destined files (HTML report pages, JPG charts).

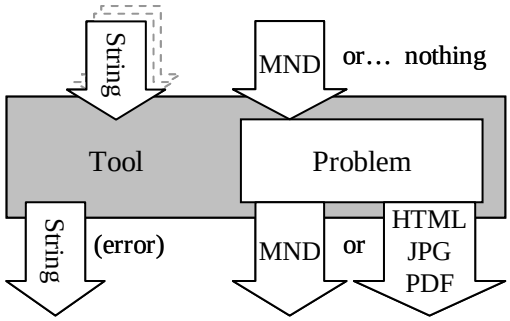

Fig 6: Summary of input and output types required of produced by a tool.

Assuming the existence of one or more WS tools able to interpret and/or produce data at the MND format, two more steps are needed to eventually obtain a real component oriented design: firstly ensure that the tools, even if they can read MND structures, will locate correctly the input data inside them. Secondly, make the tools interact in between.

MND can be seen as an XML extension for network problem definition. As it is still extensible, nothing guarantees that the MND content produced by a tool will be qualified for another. For example, a tool expects an indication of links length, but the network defined in a received MND document does not mention it.

As workaround, one may develop a pre-processor that transforms the MND document to make it qualified (for instance, adding a length of "1" to all links). This adaptation is unilateral since pre-processor does not verifies if transformation makes sense. To guarantee consistency, a provider and user must bilaterally agree on a list of required attributes. Multilateral agreements, involving several users and providers, offer of course the best guarantee, but are difficult to setup.

Regarding the interactions themselves, figure 7 presents some possible ways to implement them. a) illustrates the basic building block: the client has been generated or conceived from the textual description (see fig. 4), the WS is located at a known location over internet. In b), Web Service 1 makes itself one (or multiple) internal call to Web Service 2. In c) represents an application calling multiple services. The application can be very thin and only automate the sequential call to the WS. It may be more complex, iterating multiple time between the WS until reaching a solution. Finally, d) represents a Web Service with a Web interface front-end, permitting to the human user to launch request to the WS. Of course, these different architectures can be mixed.

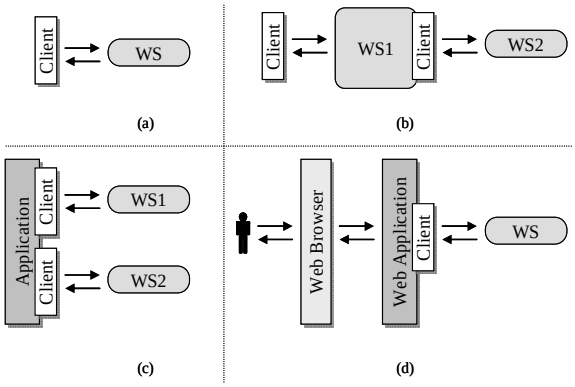

Fig 7: From the basic building block (a), different combination can be obtained, to make the Web Services interact.

3 A RWA Tool for WDM network planning

The multi-hop lightpath routing problem in Wavelength Dimension Multiplexing (WDM) networks, also called the Routing and Wavelength Assignment problem, is a great classic of network optimization and design. It plays a central role in WDM network as any further analysis on WDM (availability analysis, traffic estimation) need the RWA to be done.

The RWA problem has been for long time proved to be NP-hard, which prevents the identification of the optimal solution for large problem instances [16] but lets a great flexibility in the design of heuristics which try to approach the optimum. There are also a large number of possible criterions that can be used to determine the optimum. A great variety of RWA algorithms thus exist [17].

The RWA heuristic presented here takes as input a physical topology, where links are defined in number of fibre and number of wavelength per fibre, and logical topology (demand topology), expressed in number of wavelengths needed between pair of points in the network. Figure 8 presents and MND structure containing these elements. The output of the heuristic consists in a list of triplets (link, fibre, and wavelength) for each wavelength demand. Figure 9 presents the MND structure with the output incorporated to the input.

```
<network>
  <main_description>
    <layer id="physical" oriented="false">
     <node id="0" pos_x="623" pos_y="593"/>
     <node id="1" pos_x="494" pos_y="691"/>
     <node id="2" pos_x="683" pos_y="696"/>
     <node id="3" pos_x="527" pos_y="469"/>
     <node id="4" pos_x="756" pos_y="521"/>
     <node id="5" pos_x="797" pos_y="747"/>
     <link dest="3" orig="4" fibers="1" wavelengths="40" />
     <link dest="0" orig="4" fibers="2" wavelengths="40" />
     <link dest="2" orig="4" fibers="1" wavelengths="40" />
     <link dest="0" orig="2" fibers="2" wavelengths="40" />
     <link dest="1" orig="2" fibers="1" wavelengths="40" />
     <link dest="3" orig="1" fibers="3" wavelengths="40" />
     <link dest="1" orig="0" fibers="1" wavelengths="40" />
     <link dest="2" orig="5" fibers="1" wavelengths="40" />
    </layer>
    <layer id="logical" oriented="true">
     <link orig="0" dest="1" channels="1"/>
     <link orig="0" dest="2" channels="2"/>
     <link orig="0" dest="3" channels="1"/>
     <link orig="0" dest="4" channels="1"/>
     <link orig="0" dest="5" channels="2"/>
     <link orig="1" dest="0" channels="1"/>
          :                  :
     <link orig="5" dest="4" channels="2"/>
  </main_description>
</network>
```

Fig. 8: Input MND structure for RWA tool

```
<network>
  <main_description>
    <layer id="physical" oriented="false">
      <node id="0" pos_x="623" pos_y="593"/>
      <node id="1" pos_x="494" pos_y="691"/>
      <node id="2" pos_x="683" pos_y="696"/>
      <node id="3" pos_x="527" pos_y="469"/>
      <node id="4" pos_x="756" pos_y="521"/>
      <node id="5" pos_x="797" pos_y="747"/>
      <link dest="3" orig="4" fibers="1" wavelengths="40" />
           :              :              :
      <link dest="2" orig="5" fibers="1" wavelengths="40" />
    </layer>
    <layer id="logical" oriented="true">
      <link orig="0" dest="1" channels="1"
            mapping="[(0-1,f1,w1)]"/>
      <link orig="0" dest="2" channels="2"
            mapping="[(2-0,f1,w1)],[(2-0,f1,w2)]"/>
      <link orig="0" dest="3" channels="1"
            mapping="[(0-1,f1,w2),(1-3,f1,w2)]"/>
           :                    :
      <link orig="5" dest="4" channels="2"
            mapping="[(5-2,f1,w1),(4-2,f1,w1)],
                     [(5-2,f1,w2),(4-2,f1,w2)]" />
  </main_description>
</network>
```

Fig. 9: Output MND containing the RWA result.

The heuristic does not require any additional parameters, and the request SOAP message only contains the MND (fig. 10). Response SOAP message is sketched in figure 11.

```
<?xml version="1.0" ?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns1="rwa.tool">
  <soapenv:Body>
    <ns1:call>
      <network>
        <main_description>
                :
        </main_description>
      </network>
    </ns1:call>
  </soapenv:Body>
</soapenv:Envelope>
```

Fig. 10: SOAP request containing the MND listed in figure 8.

An RWA at disposal is interesting for many reasons. Firstly, anyone can test it, compare it to other related

heuristics, and practically evaluate its qualities. It also provides additional testing opportunities for the authors. Finally, it spares time to other development team who can directly embed it inside another study.

```
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="rwa.tool">
  <soap:Body>
    <ns1:response>
      <network>
        <main_description>
              :
        </main_description>
      </network>
    </ns1:response>
  </soap:Body>
</soap:Envelope>
```

Fig. 11: SOAP response

4 A visualisation tool with Web interface

As aforementioned and illustrated on figure 6, a Web Service may also be completely dedicated to reporting and presentation purposes. The tool presented here completely falls into this category.

One of the goals of MND is to permit the use of a unique graphical user interface (GUI) to represent any type of network or structure. Traditional GUI systems are generally platform dependant (windows native, java/swing, GTK) which is not compatible with the concept of unique GUI. Only web browser offer a graphical interface common to all platforms.

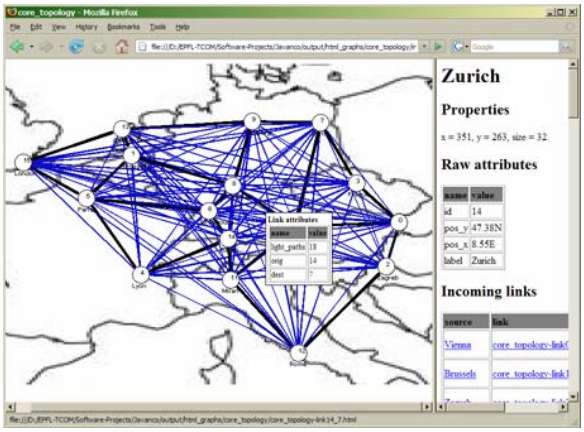

Fig. 13: The Web interface integrated inside a browser window.

Therefore, to achieve a completely platform independent GUI, a Web Service transforming an MND file into a set of browser displayable element has been realised. Using the MND core parameters [15], it builds a 2D representation of the network contained into the MND file. Link or nodes attributes can be consulted, added or removed. Some of them can be used to change the appearance. Figure 13 shows a snapshot of the GUI.

For deployment, an intermediate block must be installed between the browser and the WS, as depicted in figure 7d. Although it would be possible for a browser to post a HTTP request containing a SOAP request, the Web Service returns all its output in one pass, in a SOAP message, while the Web Browser need these files each one after each other. Figure 14 details the scenario. A Web Application (included inside a web server for instance) is thus needed in between to extract the file out the SOAP envelope (3), send the main HTML page and temporarily cache the other files (4), and then respond to the successive requests (5)-(6) for other files referenced in the HTML page.

SOAP messages structure is similar to the one used in the RWA tool.

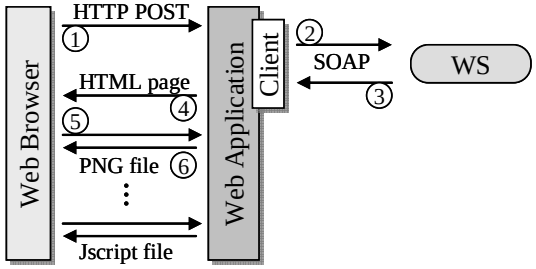

Fig 14: Organisation between the WebApplication

5 Key aspects summary and comparison

To ease descriptions and comparisons, component oriented platforms can be considered among three aspects: in terms of component awareness and quality; in terms of integration mechanisms between the components; in terms of general costs, complexity level and usability (deployment, GUI, price).

The proposed Web Services based approach is decomposed according to these aspects. A comparison with other approaches is then presented.

5.1 Component awareness and quality

In respect to the components themselves, the situation is contrasted. On one side, only two components are offered (the two presented in this paper), which is nothing compared to the multiple libraries provided by OPNET or NS-2, and absolutely not sufficient to be considered as a development environment. On the other side, it may offer in the future many components covering many domains and subjects, if the users are contributing to it.

The contrast also exists regarding the quality: various components may appear with the time, but nothing guarantees the efficiency of the reliability of each of them. However, as a WS deployment keep the code protected while permitting to broadcast the function, components implementing the newest research results may appear first as WS.

In this way, situation is the same as for Web Pages. A large quantity of information is available: some relevant, many not; some exclusive and new, many outdated. The responsibility of trusting the information (or a service) is let to the user. This one makes is choice using implicit (name of the server, of the provider) or explicit mechanisms (certificates, authentication).

5.2 Component integration and inter-component interactions

Regarding the integration of the components and the handling of their interactions, everything is inherited from the Web Services principle. The concept of "one request/one response" over SOAP and HTTP makes the Web Services simple to use and accessible to anyone.

The induced overheads (HTTP header, SOAP envelope, XML tags), which required special processing (parsing, marshalling), increase the overall response time of the call. This delay is however supposed to be minimal compared to the time needed by messages to travel between client and server (transmission time).

SOAP also imposes transmission of ASCII data instead of binary data. This again requires processing, but over all reduces offered bandwidth for data transmission [18]. If call or response contains a large quantity of data, and if additionally, the offered bandwidth between client and server is reduced, this might strongly affect the overall reaction time of the service. Remark also that WS are intrinsically stateless: no reference passing calls are possible, and all input and output data have to be transmitted each time.

These limitations exclude the use of tool needing frequent on continuous interactions with others. In particular, it dismisses many packet-level simulation components. The use of Web Services principle can therefore by applied only in presence of a clear functional decomposition [13].

Furthermore, the MND structure is not rigid enough to permit to two arbitrary WS to communicate directly without preliminary adaptations. On the side of component integration, the proposed solution presents thus more drawbacks than advantages.

5.3 Platform cost, complexity and flexibility

The benefits of the design fall in this last category. First, contrarily to commercial simulation or analysis frameworks like OPNET, this paper presents component design which is free. The price can be a criterion for small and independent research teams (who are too small to apply for OPNET academic license). It becomes the main criterion for students.

Second, the framework can almost be considered as virtual because is contains no executable, no libraries, no installation package. It is only based on a communication principle, the Web Services, and a file format, MND. There is no documentation to read, no complicated calls to the framework to perform, no platform dependent parameters to set. Once the SOAP/WS and MND mechanisms are assimilated, all combinations are theoretically possible. The required competences are reduced, which also qualify it for small team or students.

The system not either imposes specific programming language. MND, which can be considered as too flexible to model specific interactions, does not limit the field of applications. It even procures some intuition to the human reader, as it includes some meta-data mixed with the data (the XML element name, attributes names). Adaptation process between one component and another is made easier.

Finally, the design is multiplatform and distributed by essence. This two last points permits to use Web Service on different machines, on different points, which in turn permit an easy parallelization of tasks. In comparison, OPNET suffers of important restrictions regarding parallelisation [19, 20, 21]. Extensions have been developed for NS-2 but only extremely limited features have been implemented [21].

5.4 Comparison with existing frameworks

OPNET and NS-2 present rich component diversity and good component quality, but a complicated and (for OPNET) commercial framework. Library exchange is valid for punctual integration of a reduced number of components but offer no on-the-shelf components, no predefined integration mechanism nor framework. Private initiatives may provide powerful mechanisms for inter component integrations, but development costs of the components and of the framework itself is prohibitive.

The Web Service based solution greatly depends on the additional contributions in term of components. It offers reduced integration mechanisms but high framework flexibility and low cost.

The presented design proposed thus an alternative to the "big ones" (OPNET, NS-2), but specifically in two domains: first for complex but isolated function (like RWA) which is difficult to standardise. Second for users that cannot afford the time (small research teams) or the money (students) needed to use main simulation and analysis packages.

6 Conclusion

This paper proposes a component oriented approach based on Web Services. The key principle consists in restricting all interactions between elements to individual requests-responses made over internet. This

guarantees the extensibility of the concept and should keep a low complexity. Existing WS are generally available for free over internet and this should be kept in the proposed concept, leading to limited costs.

Our approach offers an alternative to the major simulation and analysis frameworks addressing dimensioning and performance analysis of communication networks. It might be especially useful for independent researchers, students, and may have interesting applications in the field of multi-machine parallelisation.

7 Acknowledgment

The authors wish to thank the Swiss Secretariat for Education and Research for supporting this work within the COST Actions 285 and 291.

8 References

[1] V. E. Paxson. Measurements and Analysis of End-To-End Internet Dynamics. Doctoral Thesis, University of California at Berkeley, 1998.

[2] B. Boehm, C. Abts, S. Chulani. Software development cost estimation approaches - A survey. *ACM Annals of Software Engineering, vol. 10, issue 1-4*, 2000.

[3] M. Lackovic, C. Bungarzeanu. A Component Approach to Optical Transmission Network Design. Modelling and Simulation Tools for Emerging Telecommunications Networks. Springer, 2006.

[4] M. Pustisek, D. Savic, F. Potorti. Packaging Simulation Results With CostGlue. Modelling and Simulation Tools for Emerging Telecommunications Networks. Springer, 2006.

[5] http://www.opnet.com/solutions/brochures/Modeler.pdf

[6] N. Kubinidze, I. Ganchev, M. O'Droma. Network Simulator NS2: Shortcomings, Potential Development and Enhancement Strategies. Modelling and Simulation Tools for Emerging Telecommunications Networks. Springer, 2006.

[7] B. P. Zeigler, S. Mittal. Final Summary Report of the Workshop on Modeling and Simulation of Ultra-Large Networks: A Framework For New Research Directions. November 2001.

[8] M. Lackovic, R. Inkret. Network Design, Optimization and Simulation Tool Cosmos. In *Proceedings of the 2nd International Workshop on All-Optical Networks*, Zagreb, Croatia, June 2001.

[9] D. Tarongì, J. Ehrensberger, S. Kessler, C. Bungarzeanu. Computer Aided Network Planning Cockpit CANPC. In *Proceedings of the 2nd International Workshop on All-Optical Networks*, Zagreb, Croatia, June 2001.

[10] J. Bih. Service oriented architecture (SOA): a new paradigm to implement dynamic e-business solutions. *ACM Ubiquity vol. 7, Issue 30*, August 2006.

[11] W. Chen. Web services - what do they mean to Web-based education? In *Proceedings of the International Conference on Computers in Education*, December 2002.

[12] http://webservices.xml.com/pub/a/ws/2002/02/06/rest.html.

[13] S. Chandrasekaran, G. Silver, J. A. Miller, J. Cardoso, A. P. Sheth. Web Service Technologies and their Synergy with Simulation. In *Proceedings of the 2002 Winter Simulation Conference*.

[14] http://www.w3.org/TR/ws-arch/

[15] S. Rumley, C. Gaumier. Multilayer Description of Large Scale Communication Networks. In *Proceedings of the COST 285 final symposium*, Mars 2007.

[16] S. Even, A. Itai, A. Shamir. On the Complexity of Timetable and Multicommodity Flow Problems. *SIAM Journal of Computing, vol. 5*, 1976.

[17] H. Zang, J. P. Jue, B. Mukherjee. A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. *Optical Networks Magazine, vol. 1, no. 1*, January 2000.

[18] K. Chiu, M. Govindaraju, R. Bramley. Investigating the Limits of SOAP Performance for Scientific Computing. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, 2002.

[19] H. Wu, R. M. Fujimoto, G. Riley. Experiences Parallelizing a Commercial Network Simulator. In *Proceedings of the 2001 Winter Simulation Conference.*

[20] M. Thoppian, H. Vu, S. Venkatesan, R. Prakash, N. Mittal, J. Anderson. Improving Performance of Parallel Simulation Kernel for Wireless Network Simulations. In *Proceedings of Milcom*, 2006.

[21] H. Ohsaki, S. Yoshida, M. Imase. On Network Nodel Division Method Based on Link-to-Link Traffic Intensity for Accelarating Parallel Distributed Simulation. In *Networking - ICN 2005, Lecture Notes in Computer Science, vol. 3420*, Springer, 2005.