User-Perceived Instantaneous Service Availability Evaluation

Rafael Rezende, Andreas Dittrich and Miroslaw Malek
Advanced Learning and Research Institute (ALaRI)
Università della Svizzera italiana (USI)
Lugano, Switzerland
Email: {ribeiror,andreas.dittrich,malekm}@usi.ch

***Abstract*—Today's businesses rely ever more on dependable service provision deployed on *information and communications technology* (ICT) infrastructures. Service dependability is highly influenced by the properties of individual infrastructure components. Combining these properties for consistent dependability analysis is challenging as every service requester might use a different set of components during service usage, constituting the user-perceived view on a service.**

This paper presents a methodology to evaluate user-perceived instantaneous service availability. It uses three input models: (1) The ICT infrastructure, with failure rates, repair rates and deployment times of all components, (2) an abstract description of complex hierarchical services, (3) a mapping that contains concrete ICT components for the service pair requester and provider, as well as existing replicas, and a duration of usage.

The presented methodology sets up the basis for automatic generation of availability models from those parts of the ICT infrastructure needed during provision for the specified pair. To calculate instantaneous availability, the age of the ICT components, the order and time of their usage during service provision are taken into account. The methodology supports generation of different availability models, exemplarily providing reliability block diagrams and fault-trees. We demonstrate the feasibility of the proposed approach by applying it to parts of the network infrastructure of University of Lugano, Switzerland.

***Keywords*-Distributed computing; Client-server systems; Availability; Fault tolerance; Modeling**

I. Introduction

In the last decade, *service-oriented architecture* (SOA) has emerged as a formalism to improve system design by focusing on modern business requirements, which everyday rely more on successful service provision. SOA proposes services as the basic building elements of a system. [1] Services are mostly deployed on *information and communications technology* (ICT) infrastructures. Every service is highly dependent on the properties of the network where it is deployed as, for instance, dependability. Therefore, to estimate the probability of a successful service delivery, it is necessary to extract and analyze the dependability properties of the relevant ICT components, in order to reflect the properties of the infrastructure layer to the service under evaluation. The complexity of the services along with this dependency imposes challenges to the evaluation of dependability, especially facing the growth of functional and non-functional requirements.

Services can be requested and provided from different locations within a network. Depending on these locations, only a fraction of the network is needed during service provision, constituting the *user-perceived* view on a service. For consistent dependability analysis, only that fraction should be taken into account. Redundant paths and redundant service providers are also common approaches to increase the probability of a successful service delivery. SOA also proposes hierarchical composition of services that can be accessed with a single interface. The criteria for service divisibility is a business decision, so that the smallest service elements – called *atomic services* – have a unique functionality and can be reused within different *composite services*. For instance, an *email* composite service can be divided into atomic services *authenticate*, *send emails* and *receive emails*, which can be reused by different composite services.

One of the most common criteria when evaluating the quality of service providers is their interval availability, the uptime of a system over a reference period. As motivated above, for every service and every pair of requester and provider, a different set of components is required and accessed at specific instants. As availability decreases over time, the last maintenance of components will also impact the overall service availability. Moreover, the longer the service is expected to run, more relevant is the possibility of eventual failure during its execution.

In addressing these issues, we propose the evaluation of user-perceived *instantaneous* service availability, the probability of a service to be available at a specific point in time. It takes an ICT infrastructure model, an abstract service description and a mapping between them as inputs. The infrastructure model includes all ICT components, their failure and repair rates and deployment times. The abstract service model describes hierarchical services as composition of atomic services. The mapping contains concrete ICT components for the service requester and provider, including possibly redundant components and their expected duration of usage. Using these models, the methodology automatically generates an availability model from those parts of the ICT infrastructure needed during service provision for

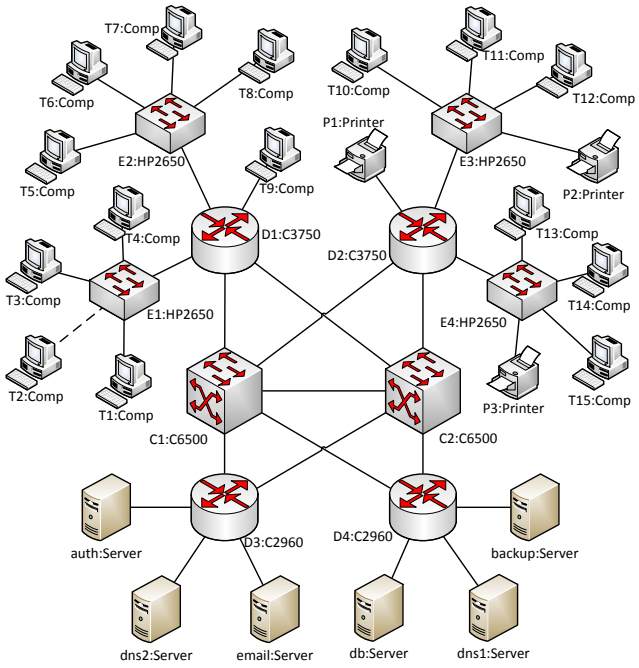

Figure 1. Network infrastructure based on university campus network.

the specified user-perceived view. The methodology supports the generation of different availability models, as we demonstrate by providing either *reliability block diagrams* (RBDs) or *fault-trees* (FTs). We demonstrate the feasibility of the methodology by applying it to parts of the network infrastructure of University of Lugano, Switzerland. This infrastructure is depicted in Figure 1. It consists of six interconnected routers and switches in its core with servers directly connected to it, and tree-like peripheral networks composed of clients and printers. A more detailed description can be found in Section VI.

The remainder of the paper is organized as follows. Section II provides an overview of availability metrics. Related work is introduced in Section III. We state the scientific problem in Section IV. The methodology to evaluate user-perceived instantaneous service availability is presented in Section V and demonstrated in Section VI. Section VII concludes this work.

II. Background

Empirical analyses have shown that components are more prone to failure at the beginning and at the end of their life-cycles. In this sense, the failure rate (λ) over time results in the so-called bathtub failure curve. Vendors usually try to mitigate the effects of infant mortality by intensively testing the hardware before dispatching it so that customers receive the product at the stage of lowest failure probability, in which the failure rate is nearly constant until the wear-out stage, that delimits the end of product's life-cycle. This stage of constant failure rate corresponds to the major part of the life time of an electronic component. The failure *probability density function* (PDF) $f(t)$ gives the relative frequency of failures at any given time t. For a constant failure rate, it can be approximated as an exponential function (see Equation 1). The *cumulative distribution function* (CDF) $F(t)$ represents the probability of a failure occurring before time t, and is given by the Equation 2. The complement of the CDF is therefore the probability of a component to perform its functions for a desired period of time without failure, better known as the reliability $R(t)$ of a component (Equation 3).

$$f(t) = \lambda \cdot e^{-\lambda \cdot t} \tag{1}$$

$$F(t) = \int_0^t f(t) \cdot dt = 1 - e^{-\lambda \cdot t} \tag{2}$$

$$R(t) = 1 - F(t) = \int_t^\infty f(t) \cdot dt = e^{-\lambda \cdot t} \tag{3}$$

For repairable systems, the probability of a component to be alive at time t is given by the probability that no failure has occurred before t – reliability $R(t)$ itself – plus the probability that, after the last failure, the component was repaired at time x, with $0 < x < t$, and has worked properly since then – $R(t-x)$. This probability is called availability $A(t)$ or more specifically, instantaneous availability. It can be expressed in terms of $R(t)$ and the probability of repair at instant x, given by $m(x) \cdot dx$ (see Equation 4). For constant failure rate λ and repair rate μ, the availability $A(t)$ can be expressed as in Equation 5. As can be noticed, for a repair rate tending to zero, $A(t)$ tends to reliability $R(t)$.

$$A(t) = R(t) + \int_0^t R(t-x) \cdot m(x) \cdot dx \tag{4}$$

$$A(t) = \frac{\mu}{\lambda+\mu} + \frac{\lambda}{\lambda+\mu} \cdot e^{-(\lambda+\mu)\cdot t} \tag{5}$$

The interval availability $A_I(t)$ is the probability that a system is operational during a period of time (Equation 6). In this paper, the function $A(t_1, t_2)$ is used interchangeably for interval availability, where t_1 and t_2 denote the start and end times of the evaluated interval. The steady-state availability is the probability that a system is operational when $t \to \infty$. As seen in Equation 7, it depends only on the failure and repair rates of components.

$$A_I(t) = A(0,t) = \frac{1}{t-0} \cdot \int_0^t A(\tau) d\tau \tag{6}$$

$$A = \frac{\mu}{\lambda+\mu} \tag{7}$$

Hardware vendors usually provide the *mean-time-to-failure* (MTTF) of their products. For repairable systems, the *mean-time-to-repair* (MTTR) depends on the implemented maintenance processes. Constant failure and repair rates can be obtained by calculating the multiplicative inverse of the MTTF or MTTR, respectively.

III. Related Work

SOA [1] proposes services as basic building elements modeled apart of the infrastructure layer. The *service availability forum* (SAF) [2] points to a different direction in its *availability management framework* (AMF) by stating that AMF components – pieces of hardware or software – are intrinsically service providers. This paper adopts the service definition as proposed by Milanovic et al. in [3]:

Definition 1: Service is an abstraction of the infrastructure, application or business level functionality. It consists of a contract, interface, and implementation. [...] The service interface provides means for clients to connect to the service, possibly but not mandatory via network.

Their methodology proposes the automatic evaluation of steady-state service availability based on network monitoring data [3], [4]. Since they rely on a configuration management database system to gather topology information at run-time, prior evaluation during design time is not contemplated. In their methodology, they propose the usage of a *depth-first search* (DFS) algorithm [5] to discover all possible paths between service requester and provider, generate a boolean equation from each path and then merge and simplify them into a single equation using an external boolean solver application. The resulting equation is then converted to an RBD model which is, again, evaluated with an external application.

The authors of [6] also describe a model to evaluate user-perceived service availability. However, their approach has a different service model in mind. Services are perceived as available by a user, if their specific resources are available upon request by that user. The model presented in the paper at hand does not take into account the quality during service usage but focuses on the availability of the network infrastructure used during service communication between arbitrary pairs requester and provider. The user-perceived scope is thus defined by the network subgraph of such a pair and not by the quality requirements of a specific user.

A comprehensive collection of foundations, models, methods and tools that can be used for service availability assessment can be found in [7]. More recently, the CHESS project [8] was funded by a joint of public-private partners of the European Union to support the development of an automated tool for dependability analysis. This project focuses on embedded systems and provides an Eclipse-based tool with its own modeling language.

Few works have been proposed in which dependability properties are fixed by means of a Unified Modeling Language (UML) profile. Basically, UML profiles are mechanisms to customize the existing UML models by allowing the addition of permanent attributes according to the nature of the target model. Profiles can be composed of *stereotypes* and its *stereotypes attributes*. Stereotypes are applied to existing UML elements, which automatically inherit the respective stereotype attributes. Following this direction, the *dependability analysis modeling* (DAM) project [9] proposes a UML profile to model dependability, and extends the existing MARTE profile [10] to better represent non-functional properties. The project provides an extensive set of models with detailed attributes, but it is focused on modeling only and does not provide a transformation to reliability models for further evaluation.

Finally, [11] proposes a methodology for evaluation of user-perceived service properties, in which the ICT infrastructure and services are modeled independently using UML object and activity diagrams, respectively. Then, a mechanism is used to project the properties of ICT components to services through an XML mapping that correlates their respective models. The work in [12] complements this methodology with a UML availability profile that supports steady-state availability analysis. Since the paper at hand extends [11] and [12] with a time dimension for instantaneous availability evaluation, their details are better described in Section V.

IV. Problem Statement

A consistent evaluation of user-perceived instantaneous service availability presents diverse challenges. A methodology is needed to merge the four dimensions – infrastructure, service, user and time – into a consistent model-driven availability evaluation. The methodology should include:

1) A model to describe the ICT infrastructure with availability properties (failure and repair rate, deployment time or time after last maintenance action).
2) A model to describe services hierarchically as a composition of atomic services.
3) A mapping between service elements and ICT infrastructure elements based on the user perspective, defining concrete service requesters and providers and their redundant instances if available.
4) A mechanism to evaluate the user-perceived service availability at a given point in time and for a given user-perceived view on a service according to the availability properties of the provided infrastructure.

Items 1, 2 and 3 should facilitate updates, as the infrastructure, its properties, the service description and user perspective will eventually change for different analyses. Item 4 should be automated and the complete methodology should be preferably implemented using mature standards and open-source tools. Failure rates of ICT components are assumed to be given by hardware vendors or estimated using monitoring data, implying also software failures of service providers. Repair rates depend on the implemented maintenance strategy. Obtaining these values is out of the scope of this work. Modeling and predicting other external factors like network load is also not considered.

V. METHODOLOGY

User-perceived service availability is highly dependent on the underlying ICT infrastructure. Since availability is an intrinsic property of the ICT layer, a mechanism is needed to reflect this property on a service availability model. Following Definition 1, we assume a service to be available if all network components needed for interface connection and communication during service provision are available. The set of ICT components deployed for service provision differs for each pair of service requester and provider within the network, and with it, the respective availability varies as well. We simplify the fault model by taking only constant failure and repair rates of ICT components into account. This means we assume that all faults that happen after a defined deployment time are combined in the failure and repair rates of individual ICT components.

In order to evaluate user-perceived *instantaneous* service availability, the *time* dimension must be added to the problem. As said, [12] is dealing only with steady-state availability, where time $t \to \infty$. In such scenarios, every component is known to have reached a constant and stable availability, so that composite service availability can be evaluated as if composing atomic services were invoked at the same time $t \to \infty$. While steady-state availability has its applications, it cannot capture the behavior of services over time as it cannot consider different execution times of atomic services and the age of their providing components. For example, if after failure a hardware component is replaced with an identical, but new unit, steady-state analysis would result in the same service availability as before the replacement. When evaluating the service availability at time $t_x < \infty$, it is important to know the estimated execution time of each atomic service, since they may be invoked at different instants and the availability of ICT components varies over time. Moreover, every component may have been deployed at different points in time. Some components may have already reached a steady-state condition, while others are in transient state.

A. Input models

ICT infrastructures are able to support a variety of services, while a single service description can be similarly applied to a diversity of networks. For this reason, the presented methodology supports independent modeling of infrastructure and services, relying on a third model to provide a relation between them. This approach implies also that changes on the network topology or service description should be reflected only in their respective models.

1) Infrastructure model: In practice, a network topology can be represented as a bidirectional graph, in which network devices and their links are respectively characterized by nodes and edges. Since networks are composed of heterogeneous nodes, in opposition to graphs, a separate model would better represent the individual characteristics of network components. This approach has been proposed in [11]: ICT components are modeled using UML class diagrams, while the network topology is represented in a UML object diagram. A simple UML profile is applied to the elements of the UML class diagram to guarantee that ICT components contain a uniform set of dependability properties compliant with the evaluation. Following this approach, we also provide an availability profile, which is depicted in Figure 2. However, since this work is focused on instantaneous availability, as opposed to steady-state, the profile includes different stereotype attributes: (1) failure rate, (2) repair rate and (3) component deployment time, at which a component is expected to have its maximum availability. Although any date and time format could be applied, epoch time has been chosen for deployment time to simplify subsequent steps. ICT components are divided into devices and links and are represented in the UML class diagram in Figure 2 as classes and associations, respectively.

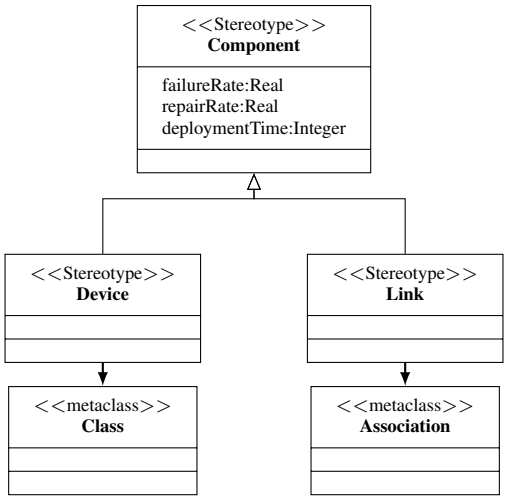

Figure 2. Availability profile used to represent availability properties.

2) Service model: Services are modeled using UML Activity Diagrams, in which every *action* element represents an atomic service, as proposed in [11]. In this methodology, the availability is measured by the probability to traverse the activity diagram from start to end nodes. To accomplish that, every composing atomic service must be successfully executed, that is, there must be at least one path between a service requester and one of its providers in which all network components are available. As an example, the composite service model in Figure 3 shows that atomic services A and B are executed in parallel, C is executed right after both of them completed, D follows after C. The represented composite service is successful if and only if all its composing atomic services are successful.

3) Mapping model: To obtain a set of potentially required ICT components for each atomic service, the service model is projected to the ICT infrastructure by a separate mapping,

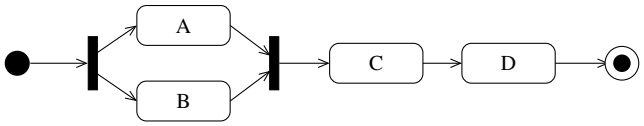

Figure 3. Example of composite service model as UML Activity Diagram.

```
<atomicservice id="C" requester="req1" timeout="10">
  <provider="prov1" duration="3" priority="0">
  <provider="prov2" duration="4" priority="1">
  <provider="prov3" duration="8" priority="1">
</atomicservice>
```

Figure 4. Mapping example of single atomic service in XML code.

represented in an XML file. The mapping denotes which node is requesting a specific atomic service and which nodes are providing that service. Contrarily to the *service mapping pair* proposed in [11], this methodology introduces a mapping model that allows multiple service providers per atomic service, which enables the modeling of redundant parts (i.e. multiple DNS servers) located in different areas of the network. This feature requires an additional annotation to describe the priority of access, as the redundant parts can be accessed in parallel or in series after an eventual failure. For the latter case, further annotations are needed to define serial access times of redundant components. Additionally, this methodology supports redundancy modeling using independent components with possibly different properties, instead of identical replicas as in [11]. Service and infrastructure descriptions are time-independent and, alone, are not able to provide an estimated execution time for atomic services. This becomes instead a new parameter of the mapping. Having such information, it is possible to estimate for which interval each atomic service availability should be evaluated. The service model plays an important role by describing which atomic services are executed in parallel or in series.

This methodology uses a mapping model as exemplified in Figure 4. Multiple providers within an atomic service description are always considered to be redundant, that is, at least one of the listed instances must be available for the requester to achieve a successful service provision. Every instance may be able to provide the atomic service with a different estimated duration in seconds, and the priority (0 = highest) plays a role for the definition of start and end execution times for each instance access.

In the example of Figure 4, three providers *prov1, prov2, prov3* are able to deliver a specified atomic service *C* to the requester *req1*. Supposing this atomic service is invoked at time $t = 0s$, the example models the following behavior: component *req1* requests a service from component *prov1*, which has the highest priority. If service provision fails, *req1* requests the same service from components *prov2* and *prov3* simultaneously – both have priority 1 – after the timeout. Therefore, next requests are invoked at time $t = 10s$ and take 4 and 8 seconds, respectively. The atomic service is considered finished only after every provider has finished. The duration of atomic service *C* is then 18 seconds, that is the latest estimated end time minus the earliest estimated start time of its redundant providers. This way, the next atomic service will be invoked at time $t = 18s$.

Although there is a well-defined serial and parallel order of execution in the system behavior, availability analysis will always consider them as parallel since they represent redundancy. Details are described in Section V-C, where the example of Figure 4 is evaluated.

4) Input model considerations: As input models, this work adopts the Unified Modeling Language (UML) for infrastructure and service descriptions as it is standardized and widely used, especially for design purposes. For the mapping model, XML is chosen due to its versatility. The decisions for UML and XML guarantee that the models remain human-readable and visualization was a relevant factor driving those decisions. However, the main contribution of the proposed methodology lies in the evaluation of user-perceived instantaneous service availability, given that the ICT infrastructure is accordingly described with availability properties. Therefore, those inputs can also be provided using different formalisms, keeping intact the main purpose of the methodology but improving, for instance, the scalability of its application. One possible improvement has already been proposed in [13], where the topology information is gathered directly from the routing layer to be used in the responsiveness evaluation of service discovery in wireless mesh networks. The routing layer is also able to provide the quality of links, as it keeps statistical data about successful packet transmission among nodes.

B. Path discovery algorithm

In order to identify the ICT components potentially required for service provision, all possible paths between the service requester and provider must be traced. Multiple paths significantly increase the availability of an atomic service, as they provide redundancy. The path discovery algorithm used in this methodology was first described in [11] and is based on the DFS algorithm [5], with a path tracking mechanism to avoid live-locks in cycles. It takes a pair of service requester and provider from the mapping model, identifies both in the graph representation of the network – obtained from the UML object diagram in Section V-A1 – and traces the paths between them to be subsequently merged into a subgraph. This process is repeated for every provider described in the mapping model.

Consider the graph in Figure 5 and the atomic service mapping of Figure 4. The path discovery is executed three times, once per provider. Taking provider *prov1* as example, the algorithm is able to discover two paths starting from component *req1* (Figure 6(a)). In order to merge all

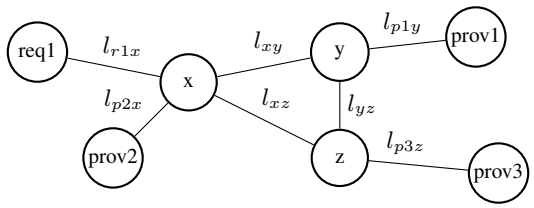

Figure 5. Simple network topology to demonstrate path discovery.

(a) req1 $\longrightarrow l_{r1x} \rightarrow$ x $\rightarrow l_{xy} \rightarrow$ y $\rightarrow l_{p1y} \rightarrow$ prov1
req1 $\longrightarrow l_{r1x} \rightarrow$ x $\rightarrow l_{xz} \rightarrow$ z $\rightarrow l_{zy} \rightarrow$ y $\rightarrow l_{p1y} \longrightarrow$ prov1

(b) req1 $\longrightarrow l_{r1x} \rightarrow$ x $\rightarrow l_{xy} \rightarrow$ y $\rightarrow l_{p1y} \longrightarrow$ prov1
x $\rightarrow l_{xz} \rightarrow$ z $\longrightarrow l_{zy}$

Figure 6. Paths between *req1* and *prov1* (a) and merged subgraph (b).

discovered paths, identical vertices and edges are merged. The resulting subgraph is shown in Figure 6(b), and can be directly transformed into an availability model, as described in the Section V-C.

The complexity of a DFS algorithm is prohibitive in networks with a high degree of connectivity, such as wireless mesh networks. For that reason, the authors of [13] propose a probabilistic path discovery to reduce the complexity of regular DFS. Although it does not guarantee every possible path, the probabilistic algorithm ensures that the most relevant ones will be found. If a subsequent analysis can use such probabilistic information, this is a valid solution to achieve better scalability. The evaluation of the methodology presented in the paper at hand is able to use the output of probabilistic path discovery. But since complexity poses no problem due to the low connectivity of the network under analysis, it was chosen to do full DFS instead which will result in a higher accuracy of the results. This assumption is valid for most cable-based networks.

C. Evaluation of user-perceived service availability

In this step, the resulting subgraphs from Section V-B are transformed into individual availability models and connected to a composite model that calculates the user-perceived instantaneous service availability. A key contribution of this methodology is that the availability of individual components is shifted in time according to their deployment time and to the estimated atomic service duration. This approach provides a realistic and consistent evaluation for transient scenarios, and can be divided into two steps: The *Model generation* step generates the service availability model, which is composed of the availability of individual components arranged according to their roles in the service provision. The *Access time definition* step complements the service availability model by identifying the exact instant at which each component is invoked within its life-cycle.

1) Model generation: Existing methodologies for steady-state availability analysis mostly use RBDs to represent their output models. If the focus of modeling is on failure instead of success conditions, FTs are used. Both models provide comparable analysis for different points of view and are supported by the proposed methodology.

The user-perceived service availability model, represented in the FT in Figure 7, is composed of three main stages:

1) According to the service model in Figure 3, the condition for the composite service *S* to fail is that at least one of the atomic services fails. Therefore, the first stage of this model can be represented in the rightmost part of the tree by a single OR logic gate, where the number of input ports corresponds to the number of atomic services.
2) In Stage 2, every atomic service is represented by an AND logic gate with every provider connected to an input port. This logic gate represents a condition in which all providers must fail to result in an atomic service failure. This information is provided by the mapping model in Figure 4, which contains three providers $prov1$, $prov2$ and $prov3$.
3) Stage 3 in Figure 7 does not have a fixed pattern. Its logical circuits depend exclusively on the subgraphs from Section V-B. Components essential for service provision of a specific provider are connected to OR logic gates, while redundant components are connected to AND logic gates. The third stage shows the logic circuits of the resulting subgraph in Figure 6(b), corresponding to communication between requester *req1* and provider *prov1* in Figure 4.

The same problem can be modeled using an RBD, depicted in Figure 8. The layers 1, 2 and 3 correspond to the equivalent stages in Figure 7: Composite service S, atomic service C and component $req1$ using component $prov1$ when requesting C. According to the RBD formalism, logical AND gates represent parallel blocks while logical OR gates represent serial blocks. The models in Figures 7 and 8, in addition to failure and repair rates of individual ICT components, are sufficient to evaluate the user-perceived steady-state service availability. In the next section, the access time definition is explained that allows these models to evaluate instantaneous availability of composite services.

2) Access time definition: In a non-steady-state scenario, we want to evaluate how availability decreases over time. The deployment time defines the instant at which individual components were fully available. Furthermore, using the additional information proposed in the mapping model – duration and priority – it is possible to calculate when every single atomic service provision is expected to start and finish. The assumption that every service provision starts at the same time is only valid for steady-state evaluation, where components have reached a stable availability – a proposition hardly realistic in dynamic and heterogeneous networks.

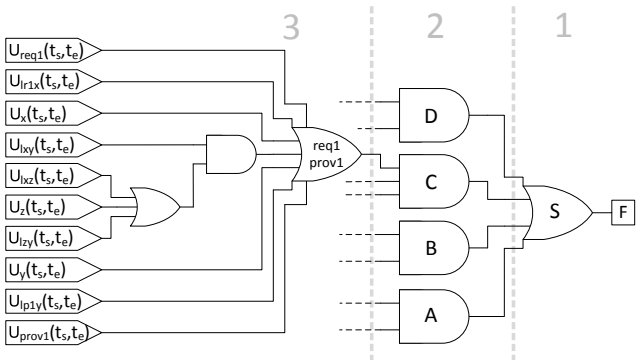

Figure 7. Partial *fault-tree* (FT) of the provided example.

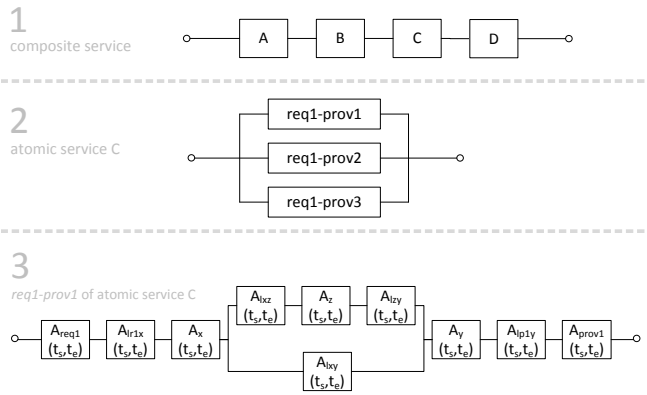

Figure 8. Partial *reliability block diagram* (RBD) of the provided example.

As seen in Section V-C1, the user-perceived service availability depends on the interval availability of many components, according to their roles in their respective atomic services. Another option would be to use the instantaneous availability of those components at a specific instant within this interval. Picking the correct instant is not trivial, however. Using the start of the interval would lead to an overly optimistic estimation while using the end time might be too pessimistic, especially for longer running atomic services. Picking any specific instant within the interval would need a sophisticated atomic service model with access durations for every component and dependencies among them. It is unclear what such a model would look like, knowing that components will change for every user-perspective, and how it could be reasonably validated to justify its usage. This is why this methodology proposes to use interval availability, the average availability over the whole duration of an atomic service. The exact range of the interval availability evaluation, however, is estimated according to the instant each component is invoked, taking its deployment time as reference. Initial references can be defined as follows: t_0 of an individual component is independently defined as its own deployment time, t_{a0} of a composite service is defined as t_0 of the newest component. Furthermore, every atomic service is invoked at different instants, with t_{a0} as reference. Their initial times are set according to the service and mapping models, which denote the serial/parallel configuration of each atomic service, and provide their estimated duration. As mentioned in Section V-A3, the estimated duration of each atomic service is given by the latest estimated end time minus the earliest estimated start time of its redundant providers.

As an example, consider the FT model in Figure 7. The relevant network components, required for the provision of atomic service C from provider *prov1* to requester *req1*, were deployed at different instants. In this example, components *prov1* and l_{p1y} are assumed the youngest and their deployment time is therefore taken as reference time t_{a0}. The composite service S will be invoked at $t = t_{a0} + 3600$, that is, one hour after the youngest component was deployed. The atomic service C, mapped in Figure 4, has an estimated duration of 18 seconds, as already described in Section V-A3. Using similar analysis, the estimated duration of atomic services A and B are set to 10 and 20. This way, it is possible to identify the time intervals of each atomic service relative to the reference t_{a0}:

$$t_{invocation\_start,A} = t_{a0} + 3600\ seconds$$
$$t_{invocation\_end,A} = t_{a0} + 3600 + 10\ seconds$$
$$t_{invocation\_start,B} = t_{invocation\_start,A}$$
$$t_{invocation\_end,B} = t_{a0} + 3600 + 20\ seconds$$
$$t_{invocation\_start,C} = t_{invocation\_end,B}$$
$$t_{invocation\_end,C} = t_{a0} + 3600 + 38\ seconds$$

The invocation start time of C is equal to the latest invocation end time of A and B, as it will be invoked only after both A and B have finished. The same applies to each service provision within atomic services. Absolute start and end times of the availability intervals of individual components i necessary for service provision p are given by:

$$t_{s,i} = t_{invocation\_start,p} - t_{deployment,i}$$
$$t_{e,i} = t_{invocation\_end,p} - t_{deployment,i}$$

Let the deployment time of component x be ten days before t_{a0}. This way $t_{s,x}$ and $t_{e,x}$ in the service provision *req1-prov1* within atomic service C are obtained as follows:

$$t_{s,x} = t_{a0} + 3600 + 20 - (t_{a0} - 864000) = 867620$$
$$t_{e,x} = t_{a0} + 3600 + 38 - (t_{a0} - 864000) = 867638$$

The interval unavailability $U_x(t_s, t_e) = 1 - A_x(t_{s,x}, t_{e,x})$ for component *x* in the example of Figure 7 should be calculated using Equation 6. For constant failure and repair rates, $A_x(\tau)$ is given by Equation 5. A different instantaneous availability equation could be derived for non-constant rates and used without any impact on the proposed methodology. Still, a constant failure rate is a reasonable approximation after the wear-in period of components, in which faults become arbitrary events along most of their lifetime.

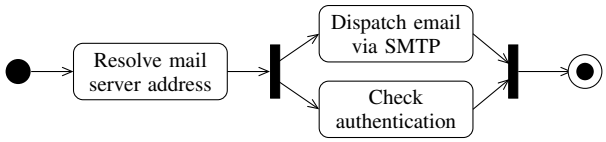

Figure 9. Send mail service represented in UML activity diagram.

VI. Case Study

This section demonstrates the proposed methodology applied to an illustrative *Send email* service that requires a *mail exchanger* (MX), a *domain name system* (DNS) server and an authentication server. A client first uses the DNS to resolve the MX address, then connects to the MX using that address. It will then send an email message over that MX by means of the common *simple mail transfer protocol* (SMTP). During SMTP communication, the MX will check authentication credentials provided by the client using an external authentication server. Three atomic services compose the *Send e-mail* service, as seen in Figure 9.

Send email is deployed on a network based on the real network infrastructure of the University of Lugano, Switzerland. The network (see Figure 1) consists of six interconnected routers and switches in its core, and tree-like peripheral networks composed of clients and printers. Servers are directly connected to the bottommost switches. Every component has a unique ID and a specified type in the format $id : type$. For better visualization, links lack labels. They are, however, referenced throughout this section as the concatenated IDs of the devices they connect. Links in this network are categorized either as wired or wireless, respectively represented by full or dashed lines. Availability properties are set according to their types (Table I). Deployment times are provided individually for each component (Table II) and are represented in epoch time. For each link, this corresponds to deployment time of the youngest component connected to its edges. Reasonable values that could reflect a real world example were chosen for all components. In an actual network infrastructure, both tables could be updated at run-time using monitoring information and a configuration management database.

The steady-state availability of *Send email* has also been evaluated in [12]. However, the network in Figure 1 purposely presents redundant DNS servers in different locations to emphasize a parallel contribution of the current paper, the possibility to assign multiple providers for a single atomic service. In [11], [12], redundant instances are always at the same location and have identical failure and repair rates.

Two distinct scenarios will now be evaluated to demonstrate the variance of instantaneous availability from different user perspectives (Section VI-A) and when changing the age or number of components (Section VI-B). The resulting RBDs and FTs are too complex to be reasonably presented in a paper and are left out, their results shown instead.

Table I
Specification of ICT component type properties.

Type	Failures/hour λ	Repairs/hour μ
C2960	0.00000545	0.5
C6500	0.00001631	0.5
C3750	0.0000053	0.5
HP2650	0.00000503	0.5
Server	0.00001667	0.25
Comp	0.00033333	0.04167
Printer	0.00138889	2.0
Wired link	0.00000769	0.25
Wireless link	0.08333333	33.33

Table II
Deployment times of individual ICT components.

ID	Deployment time (s)	ID	Deployment time (s)
t1	1366043100	e3	1366016400
t2	1366024200	e4	1367073300
t3	1368788400	d1	1366038000
t4	1368896400	d2	1346511000
t5	1366013700	d3	1366027200
t6	1366459200	d4	1346511000
t7	1366545600	c1	1346511000
t8	1367778000	c2	1346511000
t9	1369040400	p1	1366027200
t10	1366016400	p2	1366099200
t11	1366026600	p3	1368361800
t12	1366026600	backup	1368446400
t13	1367073300	db	1365850800
t14	1367073900	dns2	1366027200
t15	1367247600	auth	1366108200
e1	1366038000	email	1366050000
e2	1366013700	dns1	1355572800

Table III
Mapping model of client *t1* for *Send email*.

Atomic Service	Requester	Timeout	Provider	Prio	Durat.
Resolve address	t1	10s	dns1	0	2 sec
			dns2	1	2 sec
Dispatch email	t1	10s	email	0	5 sec
Check auth.	email	2s	auth	0	2 sec

A. Scenario – Send email from different clients

In this scenario, three different clients *t1*, *t2* and printer *p2* are requesting *Send email*. Component *t1* is connected to *e1* via a wired link (see Figure 1). The mapping model of this scenario is shown in Table III. As opposed to *t1*, *t2* is connected to *e1* via a wireless link. The rest of the infrastructure remains unchanged, so the only difference from client *t1* is a less reliable link. The third client, printer *p2*, connects to the network from a completely different position. The change of user perspective for *t2* and *p2* is achieved with only minor modifications to the mapping model, changing the requester component of the atomic services *Resolve address* and *Dispatch email* in Table III. The methodology then automatically generates different availability models.

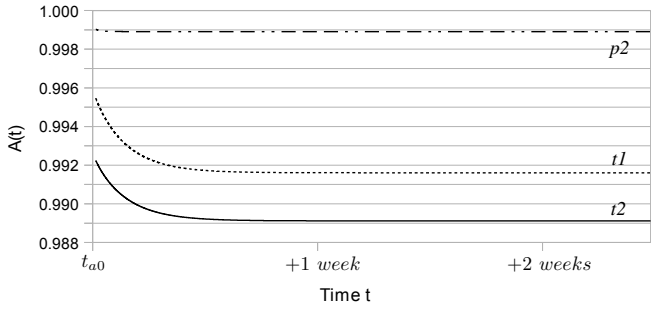

Figure 10. *Send email* service availability for different clients.

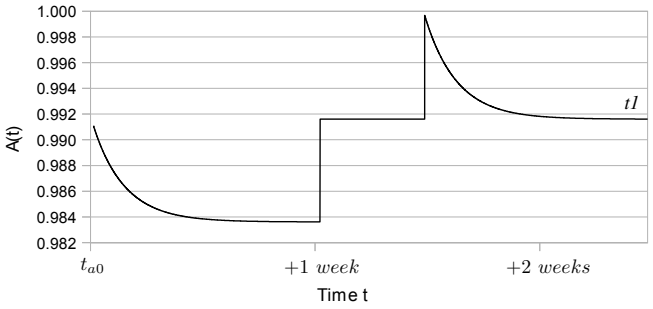

Figure 11. *Send email* service availability for *t1* when changing equipment.

Table IV
AVAILABILITY OF *Send email* FOR DIFFERENT USER PERSPECTIVES.

Requester	$A(t_{a0})$	A	$A(t_{a0}) - A$	
t1	0.9955	0.9916	0.0039	, 5.616 min/day
t2	0.9922	0.9891	0.0031	, 4.464 min/day
p2	0.9990	0.9989	0.0001	, 0.175 min/day

Table V
AVAILABILITY OF *Send email* FOR *t1* WHEN CHANGING COMPONENTS.

Requester	$A(t_{a0})$	A	$A(t_{a0}) - A$
only *dns1*	0.991091	0.983619	0.007472 , 10.76 min/day
dns1, *dns2*	0.991609	0.991605	0.000004 , 0.3 sec/day
new lab	0.999674	0.991612	0.008063 , 11.61 min/day

The reference time t_0 of this evaluation corresponds to the deployment time of the newest component potentially required during service provision, component *auth* at epoch time 1366108200 for all three clients. Evaluation of instantaneous availability $A(t)$ is then performed over time until it reaches a steady-state condition. Although this network contains younger components, these were not identified by the path discovery as potentially required during service provision and have no impact on this analysis. The resulting curves for the instantaneous availability of *Send email* when invoked at time t are presented in Figure 10. *Send email* is not fully available at t_0 because not all components were deployed at that exact time. Over time, the availability decreases until also the most recently deployed components, in this case *auth* and *d3_auth*, reach their individual steady-state availability. Figure 10 shows a comparison of the instantaneous availability of *Send email* when invoked by clients *t1* (dotted line), *t2* (full line) and *p2* (dashed line). Some corner values are shown in Table IV with the instantaneous availability $A(t_{a0})$ at the reference time, the steady-state availability A and difference of the two.

The steady-state availability for *t1* is 0.9916 and that of *t2* is 0.9891. This translates to 3 days of downtime per year for *Send email* when requested from *t1* versus almost 4 days when requested from *t2*. In both scenarios, the links between clients *t1* and *t2* and the device *e1* have a time redundancy when the DNS service is requested. If the first request to *dns1* fails, the links will be tried again when requesting *dns2*. So with respect to *t1_e1* and *t2_e1*, the DNS service will be successful when they are available either during the first or the second request. This can be modeled as a system of parallel availability blocks. During a subsequent request to the *email* server, the links are again accessed, which can be modeled as a block in series to the previous parallel system.

The evaluation of *t1_e1* and *t2_e1* according to their access order in this system of two parallel blocks in series with a single block, results in the steady-state availability of 0.99997 and 0.9975, equivalent to the one for wired and wireless links, respectively. The ratio of these values resembles the ratio of the composite service steady-state availabilities for *t1* and *t2*, since the only difference between them is the link from the clients to *e1*. $A(t)$ of *p2* is much higher than the one of *t1* and *t2*. It is also notably more stable: While the difference between $A(t_{a0})$ and A is minor for the printer, it sums up to a few minutes per day for the two client computers (see Table IV). The scenario shows that the methodology is able to capture different availabilities of the same service, depending on which client is using it and also, that availabilities vary diversely over time.

B. Scenario – Adding and replacing equipment

The second scenario evaluates how $A(t)$ changes when a set of network components is added or replaced. Evaluation is done from the perspective of *t1* with reference time t_{a0}. This time, *dns2* is absent during the first week. In the mapping model, this fact is reflected by having no redundant provider for the first atomic service. The next event is a renewal of equipment in the lab room where components *t1, t2, t3, t4* and *e1* were located (see Figure 1).

In Figure 11, the availability with a single DNS server reaches a lower steady-state availability of 0.984 during the first week, against 0.9916 with redundant DNS providers. The addition of *dns2* after one week alone does not cause significant overshoot in instantaneous availability, as all but one component (link *dns2_d3*) necessary to reach *dns2* are also required to access components *dns1* and *email*, and have already reached a steady-state condition. $A(t)$ for *Send email* minimally decreases over a few days to steady-state availability before the lab room equipment is exchanged.

Following the replacement, the $A(t)$ reaches 0.9997, as new components are known to be fully available. The individual component availability then decreases until they again reach a steady-state condition, bringing the overall service availability to the same level as before. Results are summed up in Table V, steady-state values for A represent the lowest values within the evaluated period.

When exactly the availability of a service will reach steady-state depends on the individual characteristics of the deployed components. Usually, the instantaneous availability will tend to steady-state availability after a duration in the order of weeks without changes in the network. In a regular network with a reasonable amount of components and dynamism, it is very rare to have weeks without any changes to the ICT infrastructure. This means that at any time, there will be at least some user-perspectives in a transient state, which justifies the decision to evaluate instantaneous availability.

VII. Conclusions and Outlook

Service availability is highly dependent on the properties of the underlying ICT infrastructure, which may vary considerably within a network. Thus, different users will also have divergent perceptions of service availability according to their location. The main contribution of this paper is a methodology for the evaluation of user-perceived instantaneous availability in service networks. It uses three input models: An ICT infrastructure model, a service description and a mapping model to correlate them. It considers the individual availability of every component that would potentially be required during service provision for a given pair requester and provider. Moreover, the exact component access times during service provision are taken into account in the evaluation, which results in a more accurate estimation, especially for longer execution times.

As input models, the Unified Modeling Language (UML) was adopted for infrastructure and service descriptions as it is standardized and widely used, especially for design purposes. For the mapping model, XML was chosen due to its versatility. Visualization has been an important factor and these decisions guarantee that the models remain human-readable. The methodology relies on a model transformation, aided by a path discovery algorithm to identify potentially required components. It generates a model (reliability block diagram or fault-tree) that allows evaluation of instantaneous service availability. This model can be evaluated for each instant in a time range of interest.

To demonstrate the feasibility of the methodology, an exemplary email service was analyzed in two scenarios: (1) instantaneous service availability from different user perspectives and (2) the impact on user-perceived service availability when modifying network components. The results show availability as a dynamic property and indicate how the methodology can improve network design by estimating the impact of new component deployment.

Future work will focus on improving the accuracy of the resulting availability estimation especially with respect to short term effects on the infrastructure. Extensions to support variable failure and repair rates will be examined. Furthermore, optimization of the particular methodology steps will be evaluated to improve scalability.

References

[1] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, 1st ed., ser. The Prentice Hall Service Technology Series from Thomas Erl. Prentice Hall PTR, Aug. 2005.

[2] Service Availability Forum, "Application interface specification," Aug. 2013. [Online]. Available: http://www.saforum.org

[3] N. Milanovic and B. Milic, "Automatic generation of service availability models," *IEEE Transactions on Services Computing*, vol. 4, no. 1, pp. 56–69, Jan. 2011.

[4] M. Malek, B. Milic, and N. Milanovic, "Analytical availability assessment of IT services," in *Service Availability*, ser. Lecture Notes in Computer Science, T. Nanya, F. Maruyama, A. Pataricza, and M. Malek, Eds. Springer Berlin Heidelberg, 2008, vol. 5017, pp. 207–224.

[5] S. Even, *Graph Algorithms*, 2nd ed. Cambridge University Press, Nov. 2011.

[6] D. Wang and K. S. Trivedi, "Modeling user-perceived service availability," in *Service Availability*, ser. Lecture Notes in Computer Science, M. Malek, E. Nett, and N. Suri, Eds. Springer Berlin Heidelberg, 2005, vol. 3694, pp. 107–122.

[7] N. Milanovic, "Models, methods and tools for availability assessment of it-services and business processes," Habilitation, Technische Universität Berlin, Jun. 2010.

[8] ARTEMIS Embedded Computing Systems Initiative, "CHESS project," Jul. 2012. [Online]. Available: http://www.chess-project.org

[9] S. Bernardi, J. Merseguer, and D. Petriu, "An UML profile for dependability analysis and modeling of software systems," University of Zaragoza, Tech. Rep. RR-08-05, May 2008.

[10] Object Management Group, "MARTE profile," Feb. 2011. [Online]. Available: http://www.omgmarte.org

[11] A. Dittrich, I. Kaitovic, C. Murillo, and R. Rezende, "A model for evaluation of user-perceived service properties," in *International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW)*. IEEE Computer Society, May 2013, pp. 1508–1517.

[12] A. Dittrich and R. Rezende, "Model-driven evaluation of user-perceived service availability," in *Dependable Computing*, ser. Lecture Notes in Computer Science, M. Vieira and J. C. Cunha, Eds. Springer Berlin Heidelberg, May 2013, vol. 7869, pp. 39–53.

[13] A. Dittrich, B. Lichtblau, R. Rezende, and M. Malek, "Modeling responsiveness of decentralized service discovery in wireless mesh networks," Aug. 2013, unpublished manuscript, available on request.