Reliability Analysis of the IP Multimedia Subsystem (IMS)

Qi Zhu, Swapna S. Gokhale and Veena B. Mendiratta[1]

Computer Science & Engineering Department, University of Connecticut, Storrs, CT 06269
[1]Chief Technology Office, Alcatel-Lucent, Naperville, IL 60566, USA
E-mail : ssg,@cse.uconn.edu ; veena@alcatel-lucent.com

ABSTRACT

The IP Multimedia Subsystem (IMS) is a Next Generation Network (NGN) architecture that provides a platform for integrated wireline, wireless and Internet services. Reliable operation of IMS is essential because it provides services that are critical to our day-to-day lives. A systematic methodology to quantitatively assess the reliability of IMS considering customer usage patterns, reliabilities of its components and its architecture is an essential ?rst step towards improving and ensuring its reliability and is the focus of this paper. The methodology provides a hierarchical two-step approach to compute IMS reliability. In the first step, the reliability of an individual IMS service is obtained from component reliabilities and IMS architecture. In the second step, the reliability of a single IMS session is obtained from service reliabilities and service distributions. In developing the methodology we draw upon our extensive recent experience in the area of architecture-based software reliability analysis. We illustrate the potential of the methodology to assess the influence of different parameters on IMS reliability with an example.

1. INTRODUCTION AND MOTIVATION

Software applications play a significant role in providing critical services that are essential to the smooth functioning of our daily lives. An example of one such application is the IP Multimedia Subsystem or the IMS [3]. The IMS is a standardized Next Generation Networking (NGN) architecture for telecom operators for providing mobile and fixed multimedia services. IMS is responsible for providing the basic call management (setup and tear down) capabilities. Telecom operators have offered these basic call management capabilities over traditional PSTN network infrastructure with near perfect reliability for several decades, and the society now takes for granted these basic capabilities. For the IMS system to be widely adopted, and in fact preferred over the traditional PSTN, it is then necessary that the IMS offers these basic capabilities with similar levels of reliability [9].

The IMS system also provides several other value-added services such as seamless hand over for calls between WiFi over IMS to GSM/UMTS over circuit-switched call when a handset leaves one of the networks; services combining IMS data session (such as best-effort video) with an existing voice session; presence services that allow a user to be informed about the reachability and availability of another user. Over time our expectations for these value-added will be similar to the reliability levels offered by the PSTN. In addition, extensive reliance on these services will mandate reliable operation of the IMS.

A most important step in ensuring reliable operation of the IMS is a systematic analysis of its reliability. Such analysis should consider customer usage patterns, reliabilities of its components and its architecture. While the analysis should undoubtedly provide an estimate of the IMS reliability, the primary objective of such analysis should be to facilitate an assessment of the sensitivity of the IMS to its components and the identification of components that are important from a reliability perspective. These components could then be targeted for reliability enhancement, so that the desired reliability targets can be achieved in a cost-effective manner.

The objective of this paper is to present a methodology to analyze the reliability of the IMS. The methodology is hierarchical; at the first level the reliability of an individual service offered by the IMS is obtained by composing the reliabilities of its components within the context of its architecture and at the second level the reliability of a single IMS session is obtained by composing the service reliabilities obtained from the first step in conjunction with the customer usage patterns or service distributions. The methodology thus considers the impact of several diverse aspects that influence IMS reliability namely, component failures, component interactions and customer usage scenarios in an integrated manner. In developing this methodology, we draw and build upon our extensive recent work in the area of architecture-based software reliability analysis [4]. We illustrate the use of the methodology to gain insights into the influence of different parameters on the IMS reliability using an example.

The remainder of the paper is organized as follows: Section 2 provides an overview of the architecture-based reliability analysis approach. Section 3 describes the IMS architecture. Section 4 presents the reliability analysis methodology. Illustrations of the methodology are in Section 5. Related research is presented in Section 6. Concluding remarks and future research directions are in Section 7.

2. ARCHITECTURE-BASED RELIABILITY ANALYSIS

In the architecture-based reliability analysis approach, the application architecture is given by its probabilistic control flow graph. Figure 1 shows the probabilistic control flow graph of an example application. In the figure, nodes represent the components of the application and the edges represent the flow of control among the components. The application begins with the execution of component 1 and terminates upon the execution of component 10. Prepresents the probability that control is transferred to component *j* upon the successful execution of component *i*. $p_{i,j}$s depend on the characteristics of the particular service being requested; for example the type of call and session and also on the operational profile of the application [13].

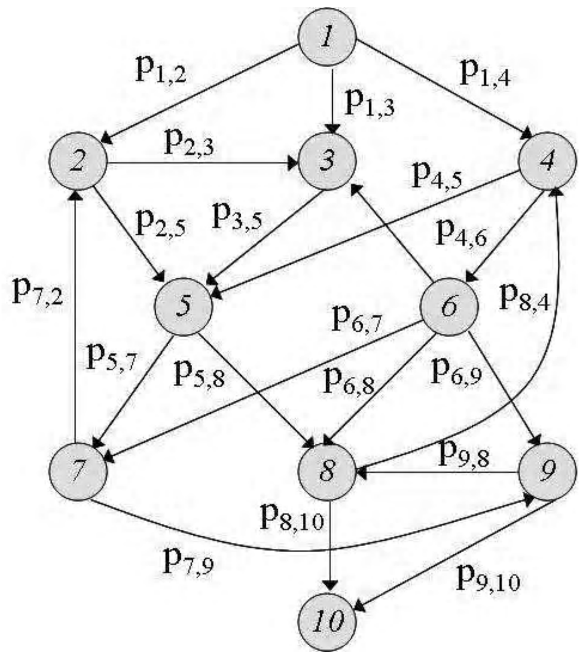

Fig. 1: Example application architecture

For the purposes of reliability analysis, the probabilistic control flow graph is mapped to a discrete time Markov chain (DTMC) [15], with a a one-to-one mapping among the components and the states. The control flow edges map to the transitions of the DTMC, and the control flow probabilities form the entries of the one-step transition probability matrix of the DTMC. The DTMC model representing the application architecture is then augmented with the component reliabilities to form a composite model. The entries of the transition probability matrix Q representing the composite model are determined as follows. Two terminal states *C* and *F* are added which respectively represent the scenarios of successful completion and application failure. For every component i, a directed branch (i, F) is created with a transition probability ($1 - R_i$), where R_i is the reliability of component i, rep- resenting application failure due to the failure of component i. The original transition probability between components i and j is modified to *Ripi,j*, which represents the transfer of control to component j from component i, conditional to the successful execution of component i. A directed branch is created from the last component to state C, with transition probability equal to the reliability of the last component, which represents successful completion of the application. Figure 2 shows the composite model for the example application in Figure 1. The composite model needs to be solved to obtain the application reliability, which is given by

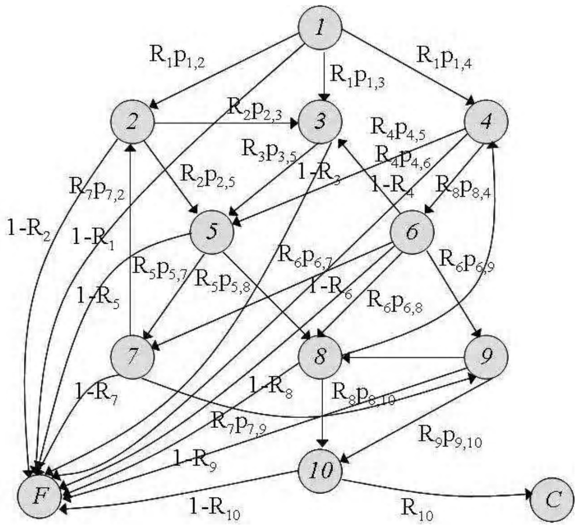

Fig. 2: Composite model of example application in Figure 1

the absorption probability in state C.

We now present the steps involved in solving the composite model to obtain the application reliability R for a generic application comprised of n components. Without loss of generality, we designate 1 to be the initial component and n to the final component.

(1) Compute matrix $\mathbf{S} = (\mathbf{I} - \mathbf{Q})^{-1}$.

(2) $R = S(1, n) \times R_n$.

Solution techniques incorporated in tools such as MAT-LAB [11] can be used to implement these steps.

3. IMS ARCHITECTURE

A brief description of the main components of the IMS architecture and their role in providing the IMS functions is provided in this section. A detailed description of the architecture can be obtained from [12].

- **IMS Terminal:** IMS Terminals (mobile phones, PDAs, computers) are used by the users to connect to a IMS network.
- **Home Subscriber Server (HSS)**: HSS is the master user database that supports IMS network entities that are actually handling the calls and sessions. It contains subscription-related information, performs authentication and authorization of the user, and can provide information about the physical location of the user.
- **Proxy-CSCF (P-CSCF):** A P-CSCF is a SIP proxy that is the first point of contact for the IMS terminal. It can be located either in the visited network or in the home network.
- **Interrogating-CSCF (I-CSCF):** A I-CSCF is a SIP proxy located at the edge of an administrative domain. The IP address of I-CSCF is published in the DNS of the domain, so that remote servers (for example, a P-CSCF in a visited domain) can find it, and use it as an entry point for all SIP packets to this domain.
- **Serving-CSCF (S-CSCF):** A S-CSCF is the central node of the signaling plane. It is a SIP server, but also performs session control. It is always located in the home network.
- **Application Server (AS):** Application servers host and execute services, and interface with the S-CSCF using SIP. An AS can be located in the home network or in an external third-party network.

The SIP servers or proxies, collectively called Call Server Control Function (CSCF) process signaling packets in the IMS. The architecture of IMS, which represents the messaging among its components is shown in Figure 3.

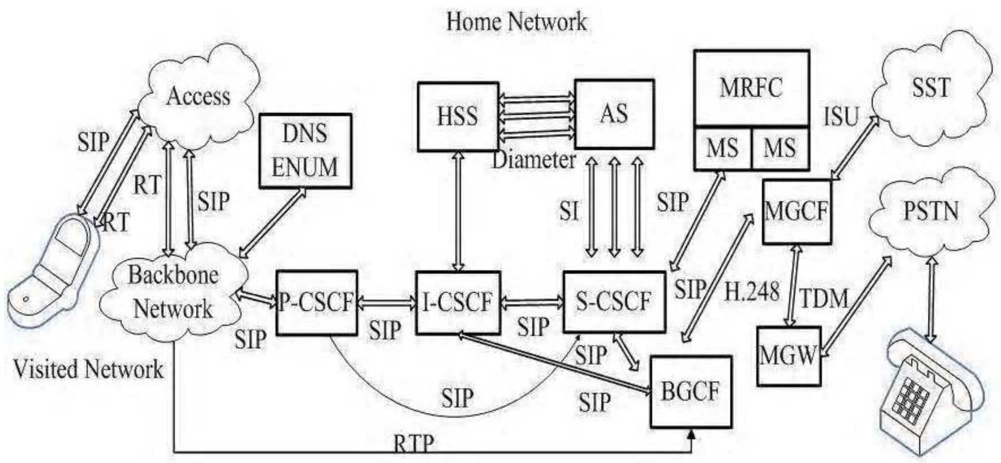

Fig. 3: IMS Architecture

4. IMS RELIABILITY ANALYSIS

The IMS provides many different types of services including basic call set up and tear down, location, presence, etc. Components of the IMS interact in a different manner for each service and these component interactions are depicted using a message flow [3], which is similar to a UML sequence diagram. Each service is thus represented by a distinct flow diagram. Furthermore, an IMS user typically requests one or more services at a time from the system, in what may be referred to as an IMS session. For example, in a session a user may first request a registration service followed by a basic call set up service. Thus, from a user's perspective the IMS is expected to function reliably for the entire session, which may potentially be composed of several several services and their corresponding flows. To capture these usage characteristics of the IMS, we now define two metrics to characterize the reliability of the IMS.

- **Service reliability:** Service reliability is the probability that the IMS functions without failure for a single service requested by the user. Service reliability must be composed from component reliabilities within the context of component interactions which are represented by the call flow of the service.
- **Session reliability:** Session reliability is the probability that the IMS functions without failure for a single user session, which may comprise of several service requests. Session reliability must be composed using service re-liabilities within the context of the service distribution which captures how the customers request these services.

In the subsequent subsections we describe a methodology to compute the above two reliability metrics.

4.1 Service Reliability

Architecture-based approach is well suited to analyze service reliability because of its dependence on component reliabilities and their interactions. The overview presented in Section II indicates that central to the architecture-based approach is a DTMC model which represents how the components of the application interact when the application provides a service. Thus, to apply this approach it is necessary to synthesize the DTMC model from the flow diagram representing the component interactions in a service.

We explain our approach to synthesize the DTMC model using an example flow diagram in Figure 4, which captures the message exchanges among the components during the registration service. The purpose of registration service is to register and authorize a user so that sessions can be established. As seen from Figure 4, the UE sends a REGISTER message to the P-CSCF, the P-CSCF sends the message to the I-CSCF to resolve the S-CSCF address,

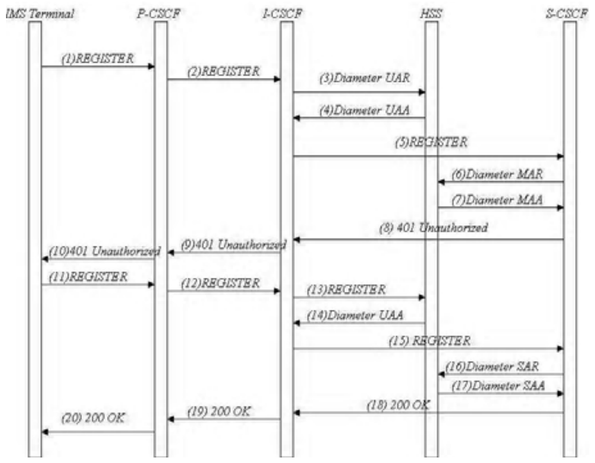

Fig. 4: Registration flow

which then routes the message to the S-CSCF. Note that two rounds of messages are involved here. The reason is that after the S- CSCF obtains authentication data, based on the user identity in the REGISTRATION message, from the HSS it sets a challenge and sends an UNAUTHORIZED message to the UE. The UE computes a response to the challenge and responds with a new REGISTER message which the S-CSCF uses for authentication. Successful authentication is finalized by a SIP OK message that the S-CSCF sends to the UE.

The three pieces of the DTMC model, namely, states, state transitions, and transition probabilities may be extracted from the flow diagram in the following manner:

- **States:** Each component, with the exception of the first component *IMS-Terminal* maps to a single state in the DTMC. The *IMS-Terminal* plays the role of two components; it is the component which initiates the service and it is also the component at which signaling or message exchanges for the service terminates. To reflect the dual role of the *IMS-Terminal* in the flow, we map the *IMS-Terminal* to two states, namely, *IMS-Terminal-Begin* and *IMS-Terminal-End*.
- **Transitions:** The transitions among the states are derived from the exchange of messages among the components. We first let **TC** denote a $m \times m$ matrix, where m denotes the number of states in the DTMC model of the flow. The entries of **TC** are initialized to zero. For a given message in the flow, let *s* and *d* denote the source and the destination of the message respectively. The $(s, d)^{th}$ entry of matrix **TC** is incremented by 1 for each message that is sent from source s to destination d. The non-zero entries of matrix **TC** then correspond to the transitions of the DTMC model. Figure 5 shows the state transition diagram of the registration flow. The state transitions are annotated with the message numbers that are exchanged among the components obtained from the flow in Figure 4. The **TC** matrix of the basic call set up flow is in Table I.

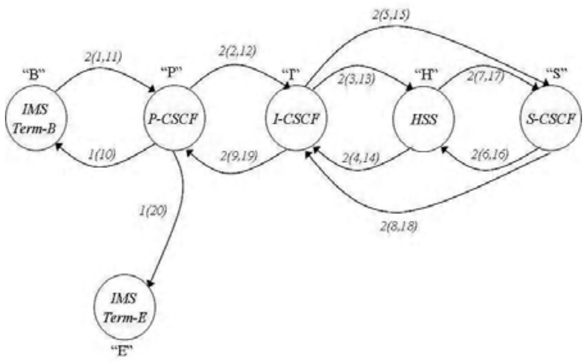

Fig. 5: State transition model of registration flow

Table 1: TC (P) matrix for Registration

	B	P	I	H	S	E
B	0	2 (1.0)	0	0	0	0
P	1 (0.25)	0	2 (0.5)	0	0	1 (0.25)
I	0	2 (0.5)	2 (0.5)	0	0	0
H	0	0	2 (0.5)	0	2 (0.5)	0
S	0	0	2 (0.5)	2 (0.5)	0	0
E	0	0	0	0	0	0

- **Transition probabilities:** Matrix **TC** obtained from the previous step provides a count of the number of messages exchanged between each pair of components, from which the transition probability matrix **P** is derived as follows.

The transition probability from a source s to a destination d, denoted *p*(*s*, *d*) is:

$$p_{s,d} = \frac{TC_{s,d}}{\Sigma_{all\ d}\ TC_{s,j}} \qquad ...(1)$$

The entries of the transition probability matrix for the registration flow are in the parentheses in Table I.

Once the transition probability matrix for a service is obtained using the above steps, service reliability may be computed by augmenting the transition matrix with component reliabilities using the method described in Section II.

4.2 Session reliability

The distribution of how different services may be requested in a session will impact session reliability. We assume that service distribution in a session may be represented using a tree-like structure, similar to the representation of the operational profile of a software system [13]. The nodes of the tree represent the services that could be requested by a user. A path in the tree, obtained by traversing from the root node to the leaf node represents a unique sequence of services that may be requested in a single session. At each node, either the user chooses a service from the possible options or decides to not request further service. Thus, the number of branches emanating from a node is one more than the total number of services that can be requested at that point in the session. The additional branch, terminating in a "null" node corresponds to the user not requesting any further service. Probabilities are assigned to these branches from each node to reflect the decision making of a user. These probabilities must sum to unity.

Based on the tree representation of the service distribution, session reliability may be computed as the weighted sum of the reliabilities of the service sequences along the paths, with the weights given by the path probabilities. Thus, to compute session reliability we need (i) path probabilities and (ii) path reliabilities. The probability of a path being chosen is computed as the product of the probabilities of the branches along that path. Path reliability is obtained as a product of the reliabilities of the services that are requested along the path. The service reliabilities used in this computation are obtained from the analysis in Section IV-A. We explain our approach with an example set of three services, namely, basic call set up (*BC*), location (*L*) and presence (*P*), with a tree-like structure representing their distribution in Figure 6. A user starts a session by requesting the basic call set up service. From this point, under the first option the user does not request any further service and in the second and third options requests location and presence services respectively. Thus there are three branches emanating from node *BC*, of which one terminates in the "null" node corresponding to no further service request. Their likelihoods are *lbc,n lbc,l* and *lbc,p* respectively. Following from node *L*, the user may then request no further service with likelihood *ll,n* or service *P* with likelihood *ll,p*. Similarly, from node *P* , the user may proceed to node "null" or to node *L* with likelihoods *lp,n* and *lp,l*.

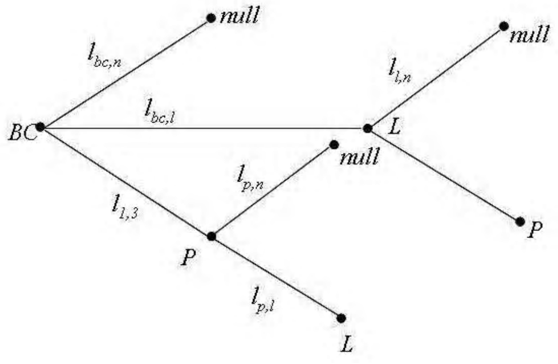

Fig. 6: Example service distribution

The above service distribution assumes that the user has to request the basic call set up service prior to requesting any other service. Furthermore, it assumes that each service can be requested only once in a session. We note, however, that both these assumptions are not limiting. Probabilistic invocation of the first service can be represented by adding a dummy initial node and branches originating from this dummy node, with the number of branches equal to the number of services that may be invoked at the beginning of the session (it may not be feasible to invoke all the possible services at the beginning of a session). Furthermore, the tree could be easily extended to allow multiple invocations of the same service during a session. These two assumptions underlying this example for the sake of illustration do not limit the applicability of our approach.

The paths through the tree along with their probabilities and reliabilities are summarized in Table II. In the table RS_i denotes the reliability of service *i*.

Table 2: Path Probabilities and Reliabilities for Figure 6

Sequence	Probability	Reliability
$BC, null$	$l_{bc,n}$	R_{bc}
$BC, L, null$	$l_{bc,l} * l_{l,n}$	$R_{bc} * R_l$
BC, L, P	$l_{bc,l} * l_{l,p}$	$R_{bc} * R_l * R_p$
$BC, P, null$	$l_{bc,p} * l_{p,n}$	$R_{bc} * R_p$
BC, P, L	$l_{bc,p} * l_{p,l}$	$R_{bc} * R_p * R_l$

Table II indicates that sessions 3 and 5 have the same set of services requested, albeit in a different order. As a result, the reliabilities of these paths will be the same. Their occurrence probabilities, however, will be different depending on the likelihood of requesting these sequences.

5. ILLUSTRATIONS

We consider the following services to illustrate our approach. A brief description of these services follows, the flows are not included due to space limitations and can be obtained from [3].

- **Registration (*R*):** IMS-level registration is the procedure where a IMS user requests IMS authorization to use the services in the IMS network. The IMS network authenticates and authorizes the user for access.
- **Basic call set up (*BC*):** This sets up a basic call between two IMS terminals. A call is set up after an IMS terminal sends a request to the IMS network and receives a response from the called terminal.
- **Messaging service (*M* S):** This service involves one Application Server (*AS*). When a request is received by S-CSCF and the trigger point conditions are met, it forwards the request to an AS, which contains the logic to provide the diversion service to the MRFC, instructing the MRFC to play a pre-recorded announcement.
- **Charging function (*CF*):** The charging function is a session establishment flow involving a roaming user when offline charging is used.
- **Presence (*W S*, *RS*):** Presence comprises of a set of IMS services that allow a user to be informed about the reachability, availability, and willingness of communication of another user. In this paper we consider two presence services, namely, Willingness Service (*W S*) and Reachability Service (*RS*) [1].

Assuming, for illustration purposes, the reliability of each component in the core IMS architecture is 0.9999, the reliabilities of the above services were computed and are summarized in Table III. Table III indicates that despite the component reliabilities of 4 9s, the reliability of each service can be achieved only at the level of 2 9s.

The path reliabilities and probabilities computed for the illustrative service distribution shown in Figure 7 are summarized in Table IV. Other service distributions, representing different customer profiles may also be developed. The session reliability, computed as the weighted average of the path reliabilities is 0.9963. Session reliability can be maintained at the level of 2 9s, at the same level as individual service reliabilities, because in the example the user requests no more than three services in a particular session.

Table 3: Service Reliabilities

Service	Reliability
Registration (R)	0.9986
Basic call (BC)	0.9993
Messaging service (MS)	0.9991
Charging #1 (CF)	0.9972
Presence #1 (WS)	0.9978
Presence #2 (RS)	0.9986

Table 4: Path Probabilities and Reliabilities for Figure 7

Sequence	Probability	Reliability
$R,\ WS,\ null$	0.1	0.9964
$R,\ WS,\ RS$	0.1	0.9950
$R,\ BC,\ null$	0.48	0.9979
$R,\ BC\ ,WS,\ null$	0.08	0.9957
$R,\ BC,\ WS,\ RS$	0.08	0.9943
$R,\ BC,\ CF,\ MS$	0.08	0.9918
$R,\ BC,\ CF,\ null$	0.08	0.9951

Next, we demonstrate the value of our approach in analyzing the sensitivity of IMS reliability to various component and service parameters. Towards this end, we set the node reliabilities to 0.90, 0.99,

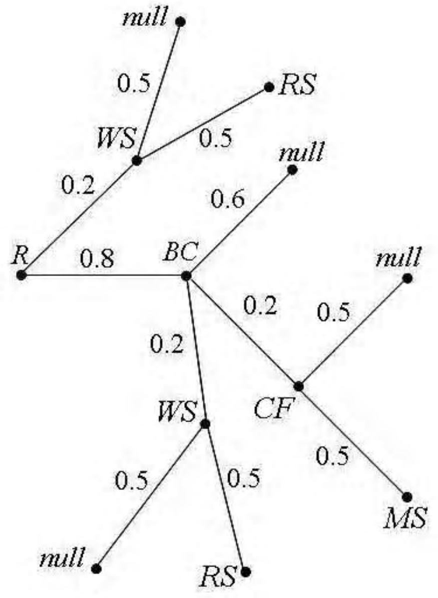

Fig. 7: IMS service distribution

0.999, 0.9999 and 0.99999 and compute the session reliability for three scenarios. These three scenarios correspond to the probability of branching from *BC* to *WS* of 0.2, 0.4 and 0.6. The branching probability from *BC* to *CF* was held at 0.2 and the probability from *BC* to *null* was adjusted accordingly. Figure 8 plots the session reliability as a function of node reliabilities for these scenarios. Note that, as expected, at low levels of node reliability the service usage profile has a greater impact on session reliability and as the node reliability increases the impact is reduced.

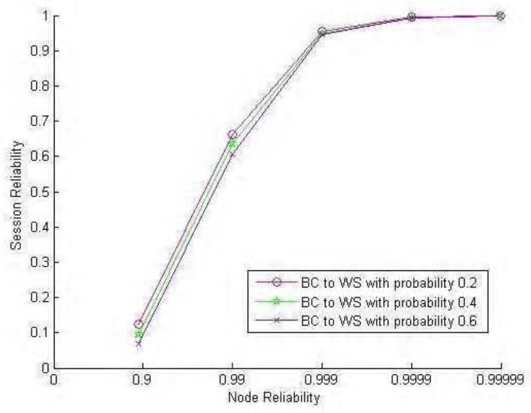

Fig. 8: Session reliability vs. node reliabilities

6. RELATED RESEARCH

In this section we summarize the related work along two dimensions, namely, (i) architecture-based reliability, and (ii) IMS reliability and place our efforts in the context.

Architecture-based software reliability analysis has been an active area of research for the past decade. A number of efforts have been devoted to the development of techniques for reliability estimation [8], [16], sensitivity analysis [10], [4], and variance quanti?cation [6], [7]. Very little work, however, has been done to apply these techniques to sizeable, real-life, industrial-strength applications. Goseva *et al.* [5] evaluate the reliability of an open-source GNU compiler using the architecture-based approach. Although the compiler constitutes a large application, it is principally a monolithic and not a component-based application. With complete access to the source code since the application is in the public-domain, and only one basic way in which it is used, the authors extract the compiler architecture from performance profiling data collected using *grof*.

By contrast, the IMS system, which is the subject of this paper is significantly different from the GNU compiler, which necessitates a fundamental adaptation of the architecture-based approach to evaluate its reliability. First, the application is closer to a component-based system, rather than a monolithic system. Second, the application provides many different services, rather than the single compilation function provided by the GNU compiler. The interactions among the components are different for each service and these interactions are given by message flows among the components. Thus, to apply the architecture-based approach to evaluate service reliability, we had to develop a method to extract the architecture of the application from the message flows for each service. Furthermore, since the different services can themselves be requested according to various distributions, we also developed a higher level composition framework to compute the application reliability from the service reliabilities.

Existing work on IMS reliability has focused on designing the system for high reliability. Bessis *et.*

al. [2] seek to improve IMS reliability by co-locating different IMS servers on the same host which reduces the number of network elements needed for the call and hence enhances the end-to-end reliability. Pant *et. al.* [14] present a methodology of the design for reliability required to address the challenges of NGN, and, in particular, of IMS design for reliability. More recent work by one of the authors presents a network level failover analysis using the IMS architecture capabilities and shows its impact on improving service reliability [12]. Despite the same focus on IMS reliability, there is a fundamental difference in the objectives of the prior work and our research. While the goal of the previous efforts was to outline design principles to improve reliability, our objective is to develop a systematic approach to quantitatively assess the reliability of different IMS configurations that may be obtained from applying these principles Our methodology could thus be used to quantitatively evaluate the impact of design principles proposed in the earlier work and decisively determine the superiority of some principles over the others in achieving IMS reliability.

7. CONCLUSIONS AND FUTURE RESEARCH

In this paper we presented a methodology for systematic quantitative assessment of the reliability of the IMS. Since the IMS offers critical services essential to our society, its reliable operation is necessary, and the reliability assessment capability provided by our methodology may provide guidance in improving the IMS reliability. The methodology considers the impact of several disparate aspects, namely, component reliabilities, IMS architecture and service distributions in an integrated manner. Finally, we illustrated the value of the methodology using a case study. Our future research consists of evaluating different design and deployment principles using our methodology.

8. ACKNOWLEDGMENTS

The research at Univ. of Connecticut is supported in part by a CAREER award from NSF (#CNS-0643971).

REFERENCES

1. T. M. Alam and Z. Wu. “Cost analysis of the IMS presence service”. In *Proc. of 1st Australian Conference on Wireless Broadband and Ultra Wideband Communication*, Sydney, Australia, March 2006.
2. T. Bessis. “Improving performance and reliability of an IMS network by co-locating IMS servers”. *Bell Labs Technical Journal*, 10(4), Winter 2006.
3. G. Camarillo and M. A. Garcia-Martin. *The 3G IP Multimedia Subsystem*. Wiley, Chester, 2004.
4. S. Gokhale. “Software reliability analysis incorporating second-order architectural statistics”. *Intl. Journal of Reliability, Quality and Safety Engineering*, 12(3):267–290, 2005.
5. K. Goseva-Popstojanova and M. Hamill. “Architecture-based software reliability: Why only a few parameters matter?”. In *Proc. of Annual Intl. Computer Software and Applications Conference*, pages 423–430, Beijing, China, 2007.
6. K. Goseva-Popstojanova and S. Kamavaram. “Assessing uncertainty in reliability of component–based software systems”. In *Proc. of Intl. Symposium on Software Reliability Engineering (ISSRE)*, pages 307–320, November 2003.
7. K. Goseva-Popstojanova and S. Kamavaram. “Software reliability estimation under uncertainty: Generalization of the method of moments”. In *Proc. of Eighth IEEE Intl. Symposium on High Assurance Systems Engineering (HASE)*, pages 209–218, 2004.
8. K. Goseva-Popstojanova and K. S. Trivedi. “Architecture–based approach to reliability assessment of software systems”. *Performance Evaluation*, 45(2-3), June 2001.
9. C. R. Johnson, Y. Kogan, Y. Levy, F. Saheban, and P. Tarapore. “VoIP reliability: A service provider’s perspective”. *IEEE Communications Magazine*, July 2004.
10. S. Kamavaram and K. Goseva-Popstojanova. “Sensitivity analysis of software usage to changes in the operational profile”. In *Proc. of IEEE/NASA Software Engineering Workshop*, pages 157–164, Greenbelt, MD, December 2003.
11. Mathworks. Matlab and Simulink for technical computing, 2007. http://www.mathworks.com/.
12. V. B. Mendiratta and H. Pant. “Reliability of IMS architecture”. In *Proc. of Australasian Telecommunications and Applications Conference*, Christchurch, New Zealand, December 2007.

13. J. D. Musa. “Operational profiles in software-reliability engineering”. *IEEE Software*, 10(2):14–32, March 1993.
14. H. Pant, C. K. Chu, S. H. Richman, A. Jrad, and G. O. O’Reilly. “Reliability of NGNs with focus on IMS architecture”. *Bell Labs Technical Journal (To Appear)*, 2008.
15. K. S. Trivedi. *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. John Wiley, 2001.
16. S. Yacoub, B. Cukic, and H. Ammar. “A scenario-based analysis for component-based software”. *IEEE Trans. on Reliability*, 53(4):465–480, 2004.