CZECH TECHNICAL UNIVERSITY IN PRAGUE

DOCTORAL THESIS STATEMENT

Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Telecommunication Engineering

Ing. Jiří Hlaváček

Robustness of VoIP systems

Ph.D. Programme: Electrical Engineering and Information Technology

Branch of study: Telecommunication Engineering

Doctoral thesis statement for obtaining the academic title of “Doctor”, abbreviated to “Ph.D.”

Prague, september 2014

The doctoral thesis was produced in combined manner Ph.D. study at the Department of Telecommunication Engineering of the Faculty of Eletrical Engineering of the CTU in Prague.

Candidat:	Ing. Jiří Hlaváček Department of Telecommunication Engineering Faculty of Eletrotechnical Engineering, CTU Technická 2, 166 27, Prague 6, Czech Republic
Supervisor:	Ing. Robert Bešťák, Ph.D. Department of Telecommunication Engineering Faculty of Eletrotechnical Engineering, CTU Technická 2, 166 27, Prague 6, Czech Republic
Opponents:	....

The doctoral thesis statement was distributed on: ....

The defence of the doctoral thesis will be held on .... at .... a.m./p.m. before the Board for the Defence of the Doctoral Thesis in the branch of study *Telecommunication Engineering* in the meeting room No. .... of the Faculty of Electrical Engineering of the CTU in Prague.

Those interested may get acquainted with the doctoral thesis concerned at the Dean Office of the Faculty of Electrical Engineering of the CTU in Prague, at the Department for Science and Research, Technická 2, Praha 6.

....

Chairman of the Board for the Defence of the Doctoral Thesis
in the branch of study Telecommunication Engineering
Faculty of Electrical Engineering of the CTU in Prague
Technická 2, 166 27 Prague 6.

Table of Contents

List of Figures

Index of Tables

1 Current Situation of the Studied Problem

The use of the Internet Protocol as a support for telecommunication systems brought many advantages, but also a few problems. The availability engagement in the Service Level Agreement of the Voice over Internet Protocol (VoIP) phone line proposed currently by the telecommunication operators is usually about three nines. The standard of the traditional telephony services is five nines. It is reached thanks to a specific hardware and software equipments. VoIP systems are popular because of cost efficiency reached using standard hardware and software, but there's an important impact on system robustness.

Current private branch exchanges (PBX) can be integrated with other information technology systems and bring the possibility to implement many advanced services, for example, a web callback. Contrary to the traditional PBX, which is often deployed as a standalone system, Internet protocol PBX (IPBX) is deployed in a complex environment with many interconnections. New challenges like the cloud and distributed environments are faced. Network availability can be ensured using redundant paths and appropriate configuration protocols. A high availability hardware with redundant components is also widely available. A typical mean time between failures (MTBF) of a server is about 6.5 year [1], but usual OS's MTBF is about 60 days [2]. IPBX software enables advanced services and integrations, but its availability is one of the VoIP system's drawbacks.

An IPBX providing advanced services and integration with other services requires more complex software. New software architectures are needed to be able to provide the same availability while increasing software complexity. Software architecture should be fault tolerant and allow context replication or restore on failure in order to preserve service continuity. The software should also be able to overcome hardware malfunctions. It should enable reliability modeling and estimation of the availability characteristics like the MTBF. Two active research fields are identified. The first one is focused on software architectures in the cloud computing and the second on the distributed software applications [3] [4] [5].

1.1 Availability of VoIP Servers

SIP proxy servers are critical components of SIP networks. SIP proxies are relaying SIP messages between terminals; each server therefore represents a single point of failure. SIP servers are software applications usually running on high availability server hardware.

Modules used in circuit switched systems have a MTBF about 1 000 000 hours (approx. 100 years, based on MTBF of German Electronic World Switch Digital (EWSD) modules). Although today's servers may have multiple processors, redundant network cards, several hard-disks working in a RAID and redundant power sources; theirs MTBF doesn't reach MTBF of circuit switched system specific hardware (MTBF of servers is usually between 50 000 and 300 000 hours). Exceptions are routers and specialized router cards with MTBF about 1 000 000 hours (Cisco router modules). Large legacy systems like the German EWSD system implement software recovery controllers and hardware monitors [6]. Software recovery controllers are special programs, which aim to detect, solve and report errors. Hardware monitors test periodically each component of the system. Each anomaly is logged up and reported. As far as I know, software implementations of SIP servers aren't that robust, interfaces for software recovery controllers aren't proposed. Hardware monitoring of servers should be done by some specialized hardware monitor with problem reporting. Actual VoIP platforms are represented by a standard redundant hardware together with an operating system configured to provide high availability services. Software architectures of VoIP servers usually do not implement recovery and fault tolerance, therefore they are not as robust as the software of the legacy systems. As an example I can cite open source VoIP servers Asterisk and Mobicents. The Asterisk PBX is a multi-threaded C application [7], therefore sensitive to deadlock and with possible segmentations faults. The Mobicents suite runs in the JBoss application server [8] [9], it's error-recovery features are also limited.

A document from Cisco [10] compares the availability of the legacy PBX to the Cisco VoIP systems. It is shown that both approaches can achieve five nines availability (5.26 minutes of downtime per year) by replication of components. This is a special case, because Cisco SIP servers are not running on a standardized server but directly by the Cisco router. Also, the features of the application programming interface (API) exposed by the presented Cisco system are very limited. A generic VoIP system can use switches and routers with the MTBF values close to routers or circuit switched systems. Anyhow, the server running SIP applications doesn't reach the MTBF value of the router. A server redundancy must be introduced in order to achieve five nines availability using standard servers.

Based on whether the call and associated services are lost or preserved following a failure, two types of recovery exist: fault recovery without context preservation and fault tolerance with context replication.

1.1.1 Fault Recovery without Context Preservation

The simplest implementation of the server redundancy is to introduce an address of a backup proxy/registrar server. This solution is only static and doesn't scale well. It's useful only as a cheap solution for small local networks. Also, if a registrar/proxy fails, established calls are lost and the endpoint is not reachable before it re-registers with the new active registrar/proxy server. One of the recommended methods to ensure service continuity in case of server failure is the use of the Domain Name System Service Record (DNS SRV) mechanism [11] [12]. This solution, discussed in [10], is called “Intelligence in previous hop”. The idea is quite simple: a special DNS request is issued to get an IP address of the SIP proxy/registrar. This implies that SIP components must implement DNS SRV lookups. The abbreviation DNS SRV means an IP address of the server that proposes the required service. The response from the DNS server is a DNS SRV record, which is a list of IP addresses; each address is associated a weight and a priority. This solution is standardized for the SIP core network, but most terminals doesn't implement DNS SRV mechanism. Furthermore, the use of DNS requests increases call establishment time, in particular in the case of primary DNS failure. In this method, a single point of failure is the DNS server itself. A secondary DNS server is usually present but the timeout and takeover adds some seconds to the call establishment. Also, the context replication issue is not addressed and thus this solution is suitable only for stateless servers.

The paper [13] explains a solution that allows using of standard DNS requests to redirect SIP terminals to one of multiple SIP servers. The idea is to introduce a domain based load balancer (DN-LB). The load balancer uses a heart-beat probe mechanism to check availability of SIP servers and in function of result redirects SIP clients to one of the pool of SIP servers. SIP clients must address DNS lookup requests to the domain based load balancer. This solution is quite easy to implement using existing software products. The main inconvenient of this solution is that the domain based load balancer represents a single point of failure. This difficulty can't be solved by using two domain based load balancers, because the DNS clients wait for some seconds before sending the request to the second DNS server (DN-LB here). This is hardly acceptable in a highly available system. Another weak point is that SIP terminals are required to do a DNS lookup before sending a SIP request. This feature is not proposed by standard SIP terminals.

It’s possible to let the media data pass directly between terminals, without passing by the proxy. Such a configuration allows to maintain calls even when a proxy failure occurs, but as the call context is lost, additional services can’t be offered.

1.1.2 Fault Tolerance with Context Replication

A common point of the solutions described below is the use of the Internet Protocol (IP) failover mechanism. This mechanism ensures that the failover is transparent from the client's point of view by migrating the IP address of the failed server to the backup server [14].

The passage of the IP address from the failed server to the backup one is fundamental for the VoIP service continuity. However, ensuring a reliable delivery of SIP messages to the operating SIP server isn't enough to ensure a successful failover without lost calls. The new active server doesn't know current calls nor registered terminals. Consequently, as the new active server doesn't know any endpoints, any incoming calls can't be processed before a new terminal registration. SIP allows use of stateless proxies, but stateful proxies are needed for billing and other services. New active server must therefore restore contexts of the established calls and registered terminals in order to ensure the continuity of services. Call establishment, call transfers, charging and other complementary services then continue to be provided by the new active server. This mechanism can work only when UDP [15] is used. TCP [16] is connection oriented and when the connection fails the terminal must open a new connection with the server. If this condition is fulfilled, described mechanism can be used even using TCP or TLS [17]. A way to ensure failover of the TCP connection is standardized in [18].

The call context is saved by the application. There are two types of fault tolerance: solutions affecting applications and solutions transparent for applications.

1.1.2.1 Solutions Impacting Applications

An application level replication is described by A. Gorti in [19]. It is using the OpenAIS AMF (Open Application Interface Specification Availability Management Framework) for the call context replication. It doesn't consider the replication of registrations. Call contexts are replicated using checkpoints. A checkpoint is defined as a current call state. The aim of checkpoints is to simplify context replication. In order to define appropriate number of checkpoints a functional analysis of SIP protocol should be done. The article doesn't explain how checkpoints were chosen. The focus is on the implementation and its complexity. Following areas of application design were identified as keys to the high availability application simplicity: event-driven model, checkpoint data identification and process roles. It was shown that using discussed solution the availability of a VoIP service can be improved from 99,999 % to 99, 9999 % without considering hardware faults. The presented solution can be considered as a proof of concept, because its real performances are very low. The proposed solution requires a specific software development, which is expensive and long. Replication enabled applications based on software frameworks for high availability (e.g., Terracota) are hard to configure and maintain. Furthermore, there are several requirements on the software architecture such as thread safety [20].

G. Kambourakis et al. propose a database-based state sharing mechanism [21]. Contexts are saved in a database and the replication is done by the database engine. This solution is relatively easy to implement as only a database connector needs to be developed instead of a complex replicated system preserving data consistency. Nevertheless, the architecture remains complex and hard to integrate with existing applications.

The paper [22] uses the OpenAIS framework to supervise SIP proxies by the dispatcher. It proposes an OpenAIS-based SIP load balancing strategy. Proxy health conditions and CPU load are monitored by the dispatchers using OpenAIS functions. This solution brings down the time needed to find out a failed proxy and improves the balancing thanks to the knowledge of the charge of each proxy. The main disadvantage of this solution is the need of some custom software development.

A common problem to all application dependent solutions is the breakdown of TCP and Transport Layer Security (TLS) connections when taking over the IP address. These connections can't be migrated without specialized operating systems or replication aware clients.

1.1.2.2 Solutions Transparent for Applications

Solutions transparent for application usually require more resources, but it is compensated by an easier development and maintenance of applications.

For special cases of Java applications, Aspect-oriented programming frameworks with real-time byte-code instrumentation for context replication can be used [23]. The cited work discusses drawbacks of this solution. Main issues are complexity of use and performances.

Hardware virtualization is another possibility discussed below.

2 Aims of the Doctoral Thesis

With consideration of the actual situation of the studied problem, the aims of the thesis were defined as follows:

- Analyze drawbacks of existing solutions and architectures for high available VoIP systems with a focus on private branch exchanges. An important axis of the analysis is the way the implementation is dealing with software failures.
- Find a software architecture suitable for implementation of highly available VoIP systems. The emphasis is on the call and service continuity, the call context must be restored or replicated following a failure. The architecture should be applicable to the distributed and cloud environments.
- Propositions should be validated on a prototype in order to validate its feasibility.
- The availability should be validated either by measurements, or by simulation.

3 Working Methods

The thesis is organized in three main parts. The first part provides a survey of the current situation, the second analyze the high available virtualization solutions and the third use of the actor model in a high available software applications.

3.1 Current Situation

This part presents a survey and analysis of existing solutions and proposals. It contains results of research, synthesis and classification of existing works. It is completed by an evaluation of the current situation in different contexts and propositions of possible enhancements and combinations improving the overall system availability.

3.2 Virtualization

The second part is focused on hardware virtualization. By analysis of the previous research works [24] [25] [26] was deduced, that the virtualization is suitable for highly available systems. Based on details explained in [27] [28], it is also expected that the continuous live migration mechanism degrades network performances. A testbed without virtualization is installed and initial measurements of network characteristics are performed. These measurements are repeated on a testbed with virtualized servers. The continuous live migration mechanism is configured and a new set of measurements is executed.

Expectations are confirmed and the problem is analyzed. The checkpointing mechanism together with the output commit problem [27] generates an important jitter and packet bursts. An improvement consisting in a fine packet classification and different routing path for the real-time packets is proposed and implemented. This modification is then verified by further measurements. An analysis of the modified method is presented.

3.3 Actor model

The third part deals with the actor model. In order to examine advantages of the actor model for the software availability, following methods are used. A prototype SIP server is designed and implemented. Its software architecture reflects capacities and fault tolerance features of the actor model presented in [29] [30] [31]. A new availability model based on stochastic colored Petri Nets suitable for actor systems is then proposed. This model is elaborated using methodologies for availability analysis presented mainly in [32] [33] [34]. The failure rate of a single actor is evaluated based on results presented in [33].

The proposed model is simulated using a simulation software [35]. Using the proposed model, the availability of the new fault tolerant implementation is estimated and compared with the availability of the corresponding standard implementation. Further analysis evaluating the impact of the failure rate on the availability is realized. This analysis is performed by simulations.

4 Results

The thesis brings a detailed survey of existing solutions and architectures for highly available VoIP systems. The software is identified as the weak point of the VoIP systems and new software architectures improving the availability are proposed.

4.1 Virtualization

The section dealing with the hardware virtualization analyses the impact of the virtualization on the VoIP system network characteristics. At first, it is shown that the virtualization introduces a small jitter. The jitter values without virtualization are about 100µs, whereas values up to 8ms are observed on virtualized servers. Thereafter, the continuous live migration mechanism is focused. This mechanism enables a hot standby high availability configuration. The work demonstrates that the actual implementation doesn't meet requirements of real-time systems like VoIP servers. The impact of the continuous live migration on the network characteristics is shown in Fig. 1. There's an important jitter and packets are released in bursts each 400ms (packets are sent each 20ms).

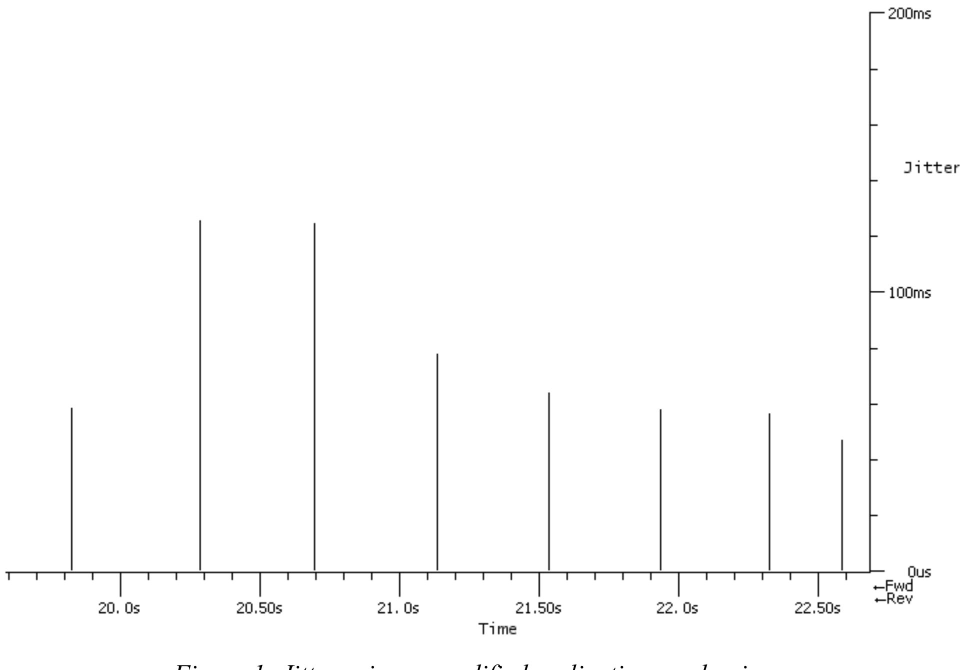

Figure 1: Jitter using unmodified replication mechanism

An improvement in the continuous live migration is described, implemented and validated. It resolves the problem of the jitter identified in the current implementation. The jitter using the improved mechanism is depicted in Fig. 2. The remaining jitter observed in Fig. 2 is due to the checkpointing required by the principle of the continuous live migration and is emphasized by the limited performances of the used testbed. A low performance testbed is used to validate the function in an environment with limited resources.

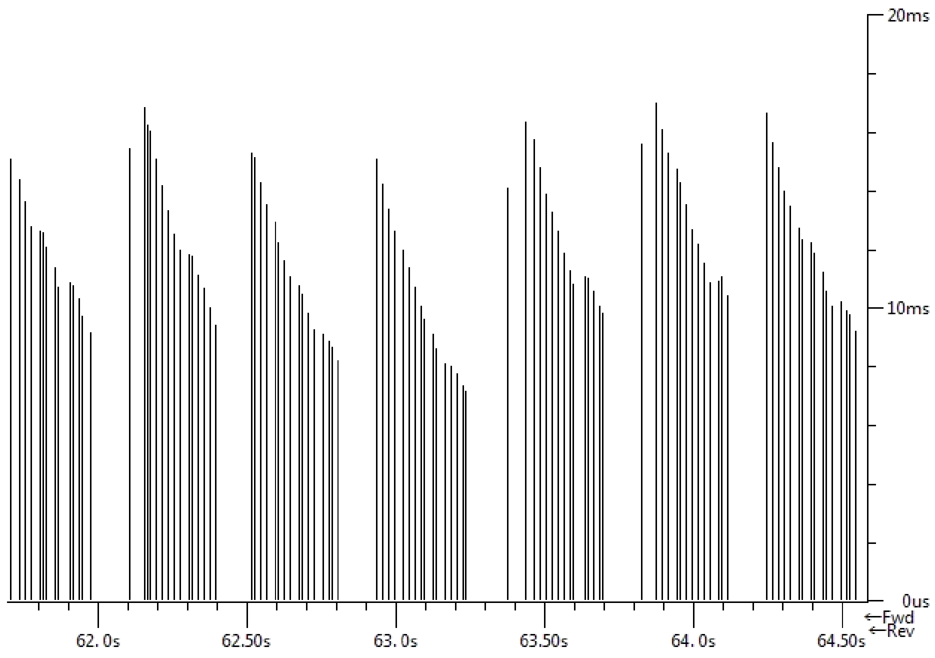

Figure 2: Jitter using modified replication mechanism

The behavior during the crash of the primary machine is also analyzed. The results show that the hot standby machine takes over and all established calls are preserved with a short call drop-out.

4.2 Actor model

The thesis also study the actor model in the context of high availability software architectures. The actor model is a hierarchical model, actor create, supervise and destroy children [36]. An actor can also supervise other actors. Each actor can be addressed by its reference and communicates with others by messages.

A highly available implementation of a SIP proxy is proposed. High availability is reached thanks to the mechanisms enabling fault-tolerance. These are also described.

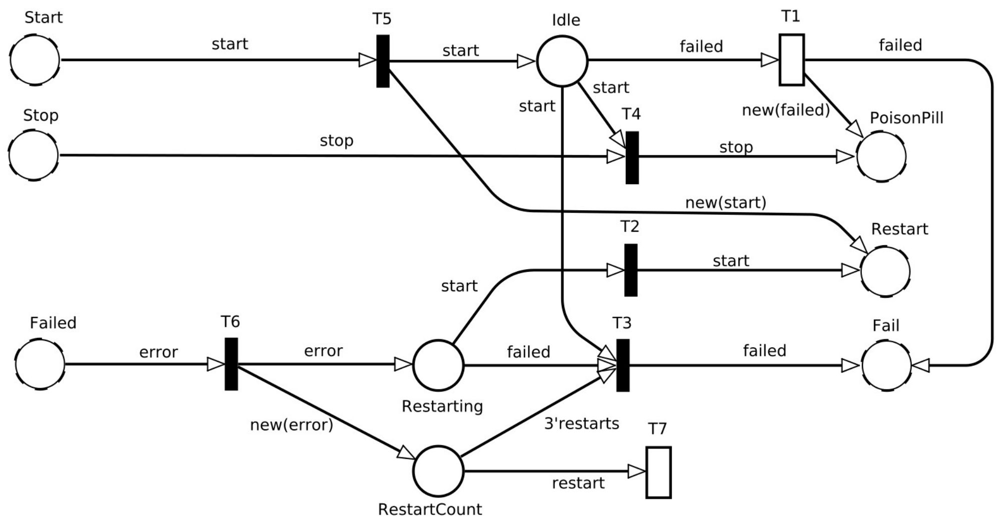

Figure 3: Single actor model

A new availability model for actor systems is proposed. It is based on the stochastic colored Petri nets. The model is hierarchical. A single actor model is defined and reused to build the model of the whole system. The single actor model presented in Fig. 3. It represents an actor controlling one child. If the child fails three times in a specified period, the actor stops itself to escalate the problem. The model contains six places serving as connection points when the model is used as a submodel. The first two places Start and Stop are life controlling places. Then there are two places for child actor lifecycle control named Restart and Poison Pill and finally two places for fail notifications, Failed for a notification from the child actor and Fail for upwards failure notification.

4.2.1 High-level model

In this section, an example model of the SIP server's transport layer is described to illustrate the use of the proposed availability model. A complete model is composed of modules representing the single actor model presented above. The Fig. 4 presents the module interface. The module is configurable; hence, it is possible to reuse the same model with a customized parameters of failure distribution.

A SIP transport layer implementation is modeled. An actor tree of the considered implementation is depicted in Fig. 5. Actors are represented as circles, supervision by arrows and message exchanges by lines between actors.

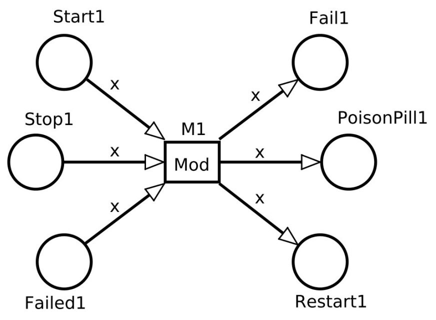

Figure 4: Single actor module interface

The presented hierarchy is relatively simple as there are only two supervision levels. The transport supervisor is an actor dedicated to fault tolerance. Its work is to deal with failure notifications. Actors UDP sender, UDP receiver are stateless and can be restarted on failure without losing any saved state. Restart is preferred to resume as it’s important to reopen the socket in case of reconfiguration such as an IP address update. SIP encoding and transport actor is stateless and is resumed on failure. It is considered that the parser error doesn't require actor restart. The transaction router maintains the directory of transactions and associated actors. The actor named Transaction 1 is its child representing a SIP transaction. These transactions are stopped when the transaction router is restarted.

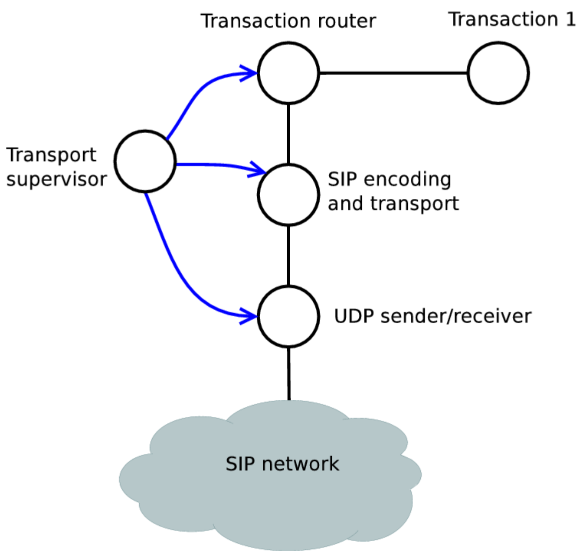

Figure 5: Actor tree of the SIP transport layer implementation

The availability model of the transport layer composed of the presented module is shown in Fig 6. Each module instance is represented by a filled rectangle named I0-I4. The module was adapted to expose the right number of child supervision interfaces. One transaction children actor is used (represented by the module I1).

The model is suitable for actor based systems. It is used to estimate the availability of the proposed implementation. A reference model for a traditional implementation is also designed.

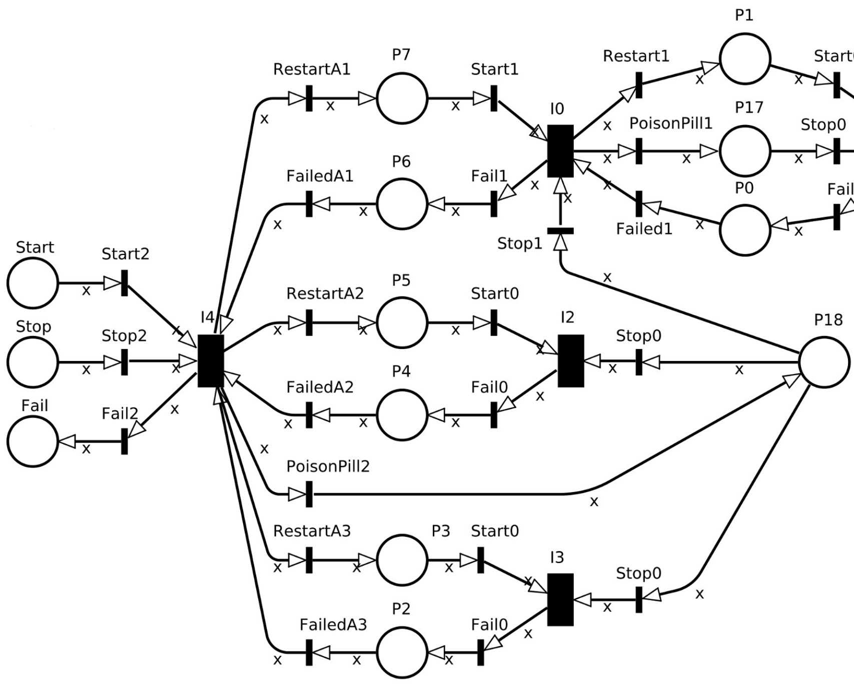

Figure 6: Transport layer availability model

Using the reference model, the availability of the actor based implementation is compared with the availability of the traditional implementation.

4.2.2 Availability Analysis

This section describes how to calculate the system's availability using the model presented in the previous section.

A method similar to the one used in [32] is used. It consists in state qualification in a failure state set and an operational state set. Each state in the model is marked as failure or operational and the mean time spent in failure and operational states set is used to determine failure and repair rates. In the presented case, failure states are Restarting and Restart states of each actor. The probability of the failure state can be written as:

$$P_{Failure}=\sum_{i=1}^{n} P_{i,Restarting}+P_{i,Restart} \qquad (1)$$

Where i determines the number of actors in the system and goes from 1 to the total number of actors in the system, $P_{i,\ Restarting}$ and $P_{i,\ Restart}$ are probabilities of each failure state by actor instance. Likewise, operational states are $P_{i,\ 1}$, $P_{i,\ 2}$ and P_{Start} and the probability the operational state can be expressed as:

$$P_{Operational}=\sum_{i=1}^{n} P_{i,1}+P_{i,2}+P_{i,Start} \qquad (2)$$

A basic model for the system's availability is a Markov chain with two states: UP and DOWN. The system gets from the state UP to the state DOWN with a failure probability λ and back to the UP state with a repair probability μ. This model is depicted in Fig. 7.

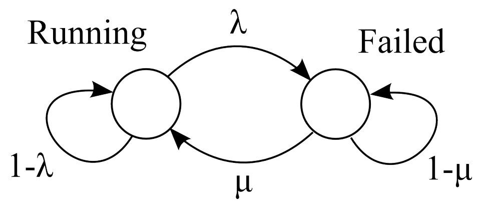

Figure 7: Two-state Markov chain

Let's define the steady-state availability as follows:

$$A=\frac{Uptime}{Uptime+Downtime} \qquad (3)$$

Using the failure and repair rates the equation (3) can be rewritten:

$$A=\frac{\mu}{\lambda+\mu} \qquad (4)$$

The considered model is limited to two states, therefore, it can be stated that:

$$P_{Operational}+P_{Failure}=1 \qquad (5)$$

The reciprocal of $P_{Failure}$ can thus be used as the failure rate λ and the reciprocal of $P_{Operational}$ as the repair rate μ. The steady-state availability can be rewritten using (5) and (2):

$$A=\frac{P_{Operational}}{P_{Operational}+P_{Failure}}=P_{Operational}=\sum_{i=1}^{n} P_{i,1}+P_{i,2}+P_{i,Start} \qquad (6)$$

4.2.3 Numerical results

According to results presented in [37] and [38], the time to failure of software application is lognormally distributed. Thus, it is assumed that the actor's time to failure represented by the transaction T1 in Fig. 4 is also lognormally distributed. Due to the utilization of multiple general transitions (many actors in an active state), a simulation must be performed to get numerical results. Simulations were performed by the TimeNet software tool developed at the Technische Universität Berlin [35].

System	**Availability**	**Downtime/year**
Actor based implementation	0.9993507	5 hours 41 minutes
Standard implementation	0.9872581	4 days 15 hours

Table 1: Availability comparison between an actor based system and a standard one

Results of performed simulations are reported in Table 1. The actor model can improve software availability by a magnitude of two orders.

5 Conclusion

The thesis analyses the robustness of VoIP systems with focus on the software architectures. The aim of the thesis is to find a software architecture allowing to reach the five nines availability standard for telephony systems. The work is divided into three parts: a survey of the highly available VoIP systems, a study of high availability solutions based on the virtualization techniques and an analysis of the actor model in the context of the fault tolerant software development.

An analysis of the highly available VoIP systems is presented first. Requirements on the network architecture, network services and hardware components are detailed with references to standards and related articles. Advantages and drawbacks of each solution are underlined. The conclusion of these investigations is that there are industrialized solutions for the network and hardware infrastructure, but only partial solutions for the software. Therefore, the second and third parts focus on suitable software architectures.

The second part study the hardware virtualization in the context of high availability systems. The hot standby configuration called continuous live migration is analyzed and improved. This type of replication works as follows: a virtualized running server is backed-up by a continuous real-time replication on another physical machine. The backup virtual machine takes the place of the master in the case of failure. Measurements show that the current implementations of continuous real-time replication are not suitable for the real-time data. It degrades the call quality by an important jitter. A modification of the replication mechanism is proposed and realized. Measurements performed with the improved version confirm that the modified hypervisor is suitable for real-time systems. As the whole machine state is replicated, all calls with associated contexts are preserved when a failure occurs. This solution is application transparent and is appropriate for VoIP systems where the high availability requirements were not considered in the initial design.

The actor model is studied in the third part. It is an interesting alternative to the object-oriented development as it permits to separate the application's logic from the error processing and parallelization. A highly-available SIP server prototype is implemented and employed fault-tolerant techniques are described. An availability model for actor systems is proposed. The model is built on the stochastic colored Petri nets and is hierarchical. A generic module representing one actor is proposed. The whole system is then modeled as a hierarchical composition of modules. The availability of the actor based systems is then compared with the availability of a standard software implementation. Results indicate that the actor model can improve software availability by a magnitude of two orders. The actor model is worth considering when a new VoIP application is designed.

Present work proposes two ways to implement high-available VoIP systems. As each one suits a particular context, these two propositions are complementary.

5.1 Future Work

A research work focusing on the efficiency of the virtual machine state replication is still in progress. Replication requires a lot of bandwidth and processing power, these are topics to be addressed by further studies. The modification proposed in this work is applicable for other applications than VoIP systems. The implementation should be extended to support other real-time data flows.

A further study of the proposed availability model would allow to derive design principles improving the overall availability. Furthermore, using the data obtained by an in-depth monitoring of actor systems the availability model prediction capacities can be improved. More research work is needed to confirm and describe possibilities of the prediction.

6 References

[1] Intel Server Calculated MTBF Estimates, rev. 1, 2009, http://download.intel.com/support/motherboards/server/sb/s3420gpmtbfcalculationrev10.pdf, [retrieved: June 2014].

[2] Kim, D. S., Machida, F., Trivedi, K.S., "Availability Modeling and Analysis of a Virtualized System," Dependable Computing, 2009. PRDC '09. 15th IEEE Pacific Rim International Symposium on , vol., no., pp.365,371, 16-18 Nov. 2009.

[3] Kanso, A., Lemieux, Y., "Achieving High Availability at the Application Level in the Cloud," 2013 IEEE Sixth International Conference on Cloud Computing, 2013, pp. 778-785.

[4] Havemose, A., Ngan, C. Y. P., "Method and system for providing high availability to distributed computer applications," Google Patents, 2013. US Patent 8,433,951.

[5] Beyersdorf, C. F., Wermser, D., Hartmann, D., Cao, X., "Virtualization of VoIP Application Servers for Implementation of Private Unified Communication Services via LTE," In Mobilkommunikation (ITG-FB 242) Technologien und Anwendungen, Vorträge der 18. ITG-Fachtagung vom 15. bis 16. Mai 2013 in Osnabrück.

[6] Botsch, D., Eberding, H., "A Real-Time Communication System with High-Level Language Software," IEEE TRANSACTIONS ON COMMUNICATIONS. 1982, 30 (6), pp. 1337–1342.

[7] Asterisk project homepage, http://www.asterisk.org/, [retrieved: June 2014].

[8] Mobicents project homepage, http://www.mobicents.org/, [retrieved: June 2014].

[9] JBoss project homepage, http://www.jboss.org/, [retrieved: June 2014].

[10] Cisco Systems: High-Availability Solutions for SIP Enabled Voice-over-IP Networks, 2002. White paper. http://www.cisco.com/en/US/tech/tk652/tk701/technologies_white_paper09186a00800a9818.shtml, [retrieved: June 2014].

[11] Rosenberg, J., Schulzrinne, H., "Session Initiation Protocol (SIP): Locating SIP servers," RFC3263, Internet Engineering Task Force, June 2002.

[12] Gulbrandsen, A., Vixie, P., Esibov, L., "A DNS RR for specifying the location of services (DNS SRV)," RFC2782, Internet Engineering Task Force, February 2000.

[13] Leu, J.; Hsieh, H.; Chen, Y.; Chi, Y. Design and Implementation of a Low Cost DNS-based Load Balancing Solution for the SIP-based VoIP service. In Proceedings

of the IEEE Asia-Pacific Services Computing Conference, IEEE Asia-Pacific Services Computing Conference. 2008, pp 310–314.

[14] Singh, K., Schulzrinne, H., "Failover, load sharing and server architecture in SIP telephony," Comput. Commun. 30, 5, March 2007, pp. 927-942.

[15] IETF RFC 768: User Datagram Protocol, http://www.ietf.org/rfc/rfc768.txt [online], August 1980.

[16] IETF RFC 793: TRANSMISSION CONTROL PROTOCOL, http://www.ietf.org/rfc/rfc793.txt [online], September 1981.

[17] IETF RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2, http://www.ietf.org/rfc/rfc5246.txt [online], August 2008.

[18] IETF RFC 5626: Managing Client-Initiated Connections in the Session Initiation Protocol (SIP). http://www.ietf.org/rfc/rfc5626.txt [online]. October 2009.

[19] Gorti, A., "A fault tolerant VoIP implementation based on open standards," In Proceedings of the IEEE Dependable Computing Conference, Dependable Computing Conference, 2006, pp. 35–38.

[20] Hlaváček, J., Bešťák, R., "Software Architectures for High Available Telecommunication Service Platforms," In Knowledge in Telecommunication Technologies and Optics - KTTO 2010. Ostrava: VŠB - TUO, FEI, Katedra elektroniky a telekomunikační techniky, 2010, pp. 92-96.

[21] Kambourakis G., et al., "High availability for SIP: Solutions and real-time measurement performance evaluation," International Journal of Disaster Recovery and Business Continuity, vol. 1, no. 1, 2010, pp. 11-30.

[22] Cheng, Y., Wang, K., Jan, R., Chen, Ch., Huang, Ch., "Efficient Failover and Load Balancing for dependable SIP proxy servers." In Proceedings of the IEEE Symposium on Computers and Communications 2008, IEEE Symposium on Computers and Communications. 2008, pp. 1153–1158.

[23] Dahlstedt, J., Vasseur, A., Bonér, J., "Java Virtual Machine support for Aspect-Oriented Programming." 5th International Conference on Aspect-Oriented Software Development (AOSD'2006) Bonn, Germany. March 20-24, 2006.

[24] Lee, M., Krishnakumar, A. S., Krishnan, P., Singh, N., Yajnik, S., "XenTune: Detecting Xen Scheduling Bottlenecks for Media Applications," IEEE Globecom 2010 (Communications Software, Services and Multimedia Applications Symposium), Miami, FL, Dec 6-10, 2010, pp. 1-6.

[25] Clark, Ch., et al., “Live migration of virtual machines,” In Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, vol. 2, USENIX Association, Berkeley, CA, USA, 2005, pp. 273-286.

[26] Patnaik, D., Bijlani, A., Singh, V. K., “Towards high-availability for IP telephony using virtual machines,” IEEE 4th International Conference on Internet Multimedia Services Architecture and Application (IMSAA), December 2010, pp. 1-6

[27] Cully, B., et al., “Remus: High availability via asynchronous virtual machine replication,” In Proc. Symp. Network Systems Design and Implementation (NSDI), 2008, pp. 161–174.

[28] Lee, M., Krishnakumar, A. S., Krishnan, P., Singh, N., Yajnik, S., “Supporting soft real-time tasks in the Xen hypervisor,” VEE 2010, Pittsburgh, PA, March, 2010, pp. 97-108.

[29] Vinoski, S., "Reliability with Erlang", Internet Computing, IEEE, vol. 11, no. 6, 2007, pp. 79.

[30] Chechina, N., Trinder, P., Ghaffari A., Green, R., Lundin, K. and Virding. R., “The Design of Scalable Distributed Erlang,” In the Draft Proceedings of the Symposium on Implementation and Application of Functional Languages 2012 (IFL'12), Oxford, UK, 2012.

[31] Erlang-an experimental telephony programming language, JL Armstrong, SR Virding - XIII International Switching Symposium, 1990.

[32] Jian, S., Shaoping, W., Yaoxing, S., "Petri-nets Based Availability Model of Fault-Tolerant Server System," Robotics, Automation and Mechatronics, 2008 IEEE Conference on, pp. 444.

[33] Salfner, F., Wolter, K., "A Petri net model for service availability in redundant computing systems,” Simulation Conference (WSC), Proceedings of the 2009 Winter, 2009, pp. 819.

[34] Vinayak, R., Dharmaraja, S., “Semi-Markov Modeling Approach for Deteriorating Systems with Preventive Maintenance,” International Journal of Performability Engineering, Volume 8, Number 5, September 2012, Paper 6, pp. 515-526.

[35] TimeNet 4.0 homepage, http://www.tu-ilmenau.de/sse/timenet/, [retrieved June 2014].

[36] Hewitt, C., Bishop, P., Steiger, R., “A Universal Modular Actor Formalism for Artificial Intelligence,” IJCAI, 1973.

[37] Mullen, R.E., "The lognormal distribution of software failure rates: origin and evidence," Software Reliability Engineering, 1998. Proceedings. The Ninth International Symposium on, pp. 124.

[38] Rahmani, C., Srinivasan, S.M. and Azadmanesh, A., "Exploratory failure analysis of open source software," Software Technology and Engineering (ICSTE), 2010 2nd International Conference on, 2010, pp. V1-51.

7 Publications Related to the Thesis

7.1 Publications in Impact Journals

Hlaváček, J. - Bešťák, R.: Fault tolerant VoIP server based on the actor model. Submitted to *Annals of Telecommunications.*
authorship share: Hlaváček 50%, Bešťák 50%

7.2 Publications in Reviewed Journals

Hlaváček, J. - Bešťák, R.: Quality of Service Management in the Voice over IP Systems. *Access server* [online]. 2010, roč. 8., č. 01, s. 00003. Internet: http://access.feld.cvut.cz/view.php?cisloclanku=2010010003. ISSN 1214-9675. (in Czech).
authorship share: Hlaváček 50%, Bešťák 50%

Hlaváček, J. - Bešťák, R.: Availability Model for Virtualized Platforms. *Advances in Electrical and Electronic Engineering*. 2013, vol. 11, no. 5, p. 316-320. ISSN 1336-1376.
authorship share: Hlaváček 50%, Bešťák 50%

7.3 Publications Excerpt in Web of Science

Hlaváček, J. - Bešťák, R.: Configuration of Live Migration for VoIP Applications. In *Proceedings of 15th Mechatronika 2012*. Praha: Czech Technical University in Prague, 2012, p. 187-190. ISBN 978-80-01-04987-7.
authorship share: Hlaváček 50%, Bešťák 50%

7.4 Other Publications

Hlaváček, J. - Bešťák, R.: Live Replication of Virtualized VoIP Servers. In *The Eighth International Multi-Conference on Computing in the Global Information Technology*. Silicon Valley: International Academy, Research and Industry Association (IARIA), 2013, p. 277-282. ISBN 978-1-61208-283-7.
authorship share: Hlaváček 50%, Bešťák 50%

Michaux, J. - Buu, E. - Hlavacek, J. and Najm, E. An open-source platform for converged services. In *Proceedings of Principles, Systems and Applications on IP Telecommunications* (IPTComm '13). 2013, ACM, New York, NY, USA, p. 1-8. ISBN: 978-1-4503-2672-8
authorship share: Michaux 25%, Buu 25%, Hlavacek 25%, Najm 25%

Hlaváček, J. - Bešťák, R.: Software Architectures for High Available Telecommunication Service Platforms. In *Knowledge in Telecommunication Technologies and Optics - KTTO 2010*. Ostrava: VŠB - TUO, FEI, Katedra elektroniky a telekomunikační techniky, 2010, p. 92-96. ISBN 978-80-248-2330-0.
authorship share: Hlaváček 50%, Bešťák 50%

Hlaváček, J. - Bešťák, R.: Improvements in the Availability of SIP Networks. In *Proceedings of the 2010 Networking and Electronic Commerce Research Conference*. Dallas, TX: American Telecommunications Systems Management Association Inc., 2010, p. 109-117. ISBN 978-0-9820958-3-6.
authorship share: Hlaváček 50%, Bešťák 50%

Hlaváček, J. - Bešťák, R.: Improvements in the Reliability of the Computer-Based Telecommunications Systems. In *Proceedings CD-ROM of Digital Technologies 2007*. Žilina: Slovenská elektrotechnická společnost, 2007. ISBN 978-80-8070-786-6.
authorship share: Hlaváček 50%, Bešťák 50%

Resumé

Telefonní hovor je součástí každodenního života, může zachránit život v případě nouze nebo pomoci překlenout vzdálenost mezi lidmi. Telefonní spojení je tradičně vnímáno jako bezporuchová služba s vysokou dostupností. Internetová telefonie umožňuje implementaci pokročilých služeb a snížení nákladů, ale přináší také komplexnost. Bohužel, systémy internetové telefonie jsou v dnešní době oproti tradičním telefonním systémům stále méně spolehlivé.

Tato práce se zaměřuje na robustnost systémů internetové telefonie (VoIP). V první části je prezentován přehled existujících návrhů a řešení. Pro každé řešení jsou rozvedeny jeho výhody a nevýhody. Z přehledu vyplývá, že existuje kompletní řešení pro síťové zařízení a služby, jakož i pro hardwarové zařízení. Nicméně, optimální architektury programového vybavení jsou stále studovány, zejména řešení pro distribuované a takzvané cloud, tedy na Internetu založené systémy.

Druhá část se zabývá počítačovou virtualizací a možnostmi replikace virtuálních strojů jako platformy pro vysoce dostupné programové vybavení. Nejprve je studován vliv virtualizace na aplikace pracující v reálném čase, jejichž příkladem jsou právě systémy internetové telefonie. Následně je studován mechanismus kontinuální replikace v reálném čase. Z výsledků práce vyplývá, že kontinuální replikace v reálném čase vytváří značné fázové chvění. Toto chvění vylučuje použití tohoto typu replikace pro systémy pracující v reálném čase. V další části je mechanismus migrace upraven tak, aby bylo chvění potlačeno a jeho vlastnosti jsou ověřeny měřením na laboratorní konfiguraci. Virtualizace spolu s kontinuální replikací v reálném čase je zajímavé řešení pro existující programové vybavení bez podpory vysoké dostupnosti.

Třetí část zkoumá aktorový model (actor model) a jeho aplikace v rámci vysoce dostupných systémů. Nejprve je proveden návrh implementace proxy serveru protokolu SIP s použitím aktorového modelu. Mechanismy umožňující odolnost proti chybám jsou v práci podrobně popsány. Dále je prezentována koncepce modelu dostupnosti vhodného pro programy založené na modelu aktorů. Tento model je hierarchický a umožňuje za pomoci modulů konstrukci komplexnějších modelů. Je založen na barevných stochastických Petriho sítích s logaritmicko-normálním rozložením četností poruch. V závěru této části jsou prezentovány simulace porovnávající dostupnost systému založeného na modelu aktorů se standardním systémem. Tyto ukazují, že aktorový model může zvýšit dostupnost služeb proxy serveru až o dva řády.

Tato práce představuje dvě možné implementace pro vysoce dostupné systémy. Virtualizace spolu s kontinuální replikací v reálném čase je vhodné řešení pro stávající aplikace bez podpory vysoké dostupnosti. Z výsledků práce vyplývá, že při vývoji nové aplikace by naopak mělo být zváženo použití modelu aktorů.

Summary

Telephony is used in our everyday life, it can save lives in emergency cases or overcome the distance between people. Traditionally, it is expected to work with practically no interruption. Voice over Internet Protocol (VoIP) technology brings new blended services and cost efficiency, but also more complexity. Unfortunately, VoIP systems are nowadays still less reliable than the traditional switched systems.

The thesis deals with the robustness of VoIP systems with focus on private branch exchanges. In the first part, a survey of existing solutions and propositions is presented and advantages and drawbacks of each possibility are analyzed. It is shown that there are complete solutions for the high availability network hardware and services, as well as for fault tolerant hardware equipments. However, optimal software architectures are still an active research field, in particular for the cloud and distributed environments.

The second part addresses virtualization and its replication capabilities as a platform for high availability software. The impact of virtualization on network characteristics of real-time systems like VoIP servers is measured first. Results show that the virtualization generates a small jitter. Thereafter, a continuous live migration mechanism is studied. This work shows that continuous live migration generates an important jitter and packet bursts. The jitter is prohibitive for use of this type of migration in the real-time systems. The migration mechanism is improved and its behavior is verified by new measurements on a testbed. The proposed enhancement eliminate the problem: the jitter observed is comparable with the jitter without continuous live migration. Virtualization is transparent at the application level. A virtualized server with the improved continuous live migration is thus an interesting solution for existing software without high availability support.

The third part studies the actor model and its applications to high available systems. A highly available implementation of a SIP proxy is proposed. It is based on the actor model and is structured to deal with software faults. Mechanisms enabling fault tolerance are explained in detail. A modular availability model suitable for actor based applications is conceived. The model is hierarchical and scales well. It is based on stochastic color Petri nets with lognormal distribution of failure rates. Simulations were performed to compare the availability of the actor based system with a standard one. Results show that the actor model can improve the availability by a magnitude of two orders.

The work presents two possible implementations of a high availability system. A virtualization based one, which is suitable for existing applications without high availability support. And an actor model based one, which should be considered when developing a new application.