FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 17, April 2004, 33-40

Evaluation of a Fault-Tolerant Call Control System

Marjan Bozinovski, Liljana Gavrilovska, Ramjee Prasad, and Hans-Peter Schwefel

Abstract: The session initiation protocol (SIP) is the main signaling protocol in the 3GPP IP multimedia subsystem (IMS). The SIP sessions in IMS have to be highly reliable. The developed fault-tolerant SIP call control concept includes state-sharing mechanism, failure-detection and fail-over management. The state-sharing mechanism representing the core entity in the overall system has been developed to meet the specific SIP functional features and requirements for reliability of SIP services. Theoretical analysis and measurements in a prototype implementation showed that TFTP outperforms FTP.

Keywords: Call control, fault-tolerance, session initiation protocol, reliability.

1 Introduction

SIP [1] is an application layer signaling protocol, which establishes, modifies and terminates multimedia sessions in the 3GPP IMS [2]. The SIP call control services must be highly resilient to server failures. Therefore, a SIP fault-tolerant system, deploying a state-sharing mechanism is required [3], [4], [5], [6]. This mechanism is matched with a failure-detection and fail-over management. Their task is to keep the SIP sessions alive, in presence of the potential server or network failures. In a replicated SIP-based session control system, call (session) states are shared among the servers. Session reliability, state consistency and system response time are important parameters. Reliability and consistency have to be kept sufficiently high,

Manuscript received February 12, 2004. An earlier version of this paper was presented at the 6th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services, October 1-3, 2003, Nish, Serbia and Montenegro.

M. Bozinovski, R. Prasad and H.-P. Schwefel are with the Center for Teleinfrastructure, Aalborg University, Niels Jernes Vej 12, 9000 Aalborg, Denmark (e-mails: marjanb@kom.auc.dk, prasad@kom.auc.dk, and hps@kom.auc.dk). L. Gavrilovska is with the Faculty of Electrical Engineering, Skopje, Macedonia, and part-time with the Center for Teleinfrastructure, Aalborg University, Denmark (e-mail: liljana@etf.ukim.edu.mk).

while minimizing system response time. In a typical session setup/termination scenario such as basic telephony call, SIP call states are driven by a binary call state machine [7]. In such a scenario a SIP session consists of only two transactions: INVITE (call setup transaction) and BYE (call termination transaction). The INVITE transaction creates the session state and the BYE transaction updates it, forwards it on to the non-SIP post-processing entity (e.g., billing system) and deletes it. This implies that either the state (if already created by an INVITE transaction) is found at a given server by the BYE transaction or not at all. In such a special case, the state consistency turns into the probability of having the call state at a given server. On the other hand, reliability defined as the probability of successful call completion (i.e., successful BYE transaction in this particular scenario) is proportional to the probability of finding the call state at a server, i.e., proportional to state consistency. Thus, reliability linearly depends on state consistency. This reasoning suggests that a state has to be replicated as fast as possible in order to have high reliability. The transporting mechanism is the crucial functionality of the state-sharing mechanism, which has to provide fast, reliable state replication in the set of replica servers. Thus, deployment of concurrency and commitment protocols is not always justified particularly in this call scenario. Using an efficient dissemination utility is a good compromise between the requirement for fast call control and high reliability at low design complexity. Thus, reliability and availability are the only relevant parameters that need to be investigated as consistency is hidden in the definition of reliability.

In this paper, an overall fault-tolerant system integrated into a SIP proxy server is proposed, which increases the overall reliability of the SIP signaling services. The paper is organized as follows. Section 2 presents the proposed state-sharing mechanism, and Section 3 explains the failure-detection and fail-over mechanisms. Section 4 describes the testbed, while the analytical and experimental evaluation is given in Section 5. Conclusions are presented in Section 6.

2 Proposed State-Sharing Algorithm

The state-sharing algorithm has a rather simple design. A text file is used as a medium for transporting SIP state update messages (SUM). The interface with the SIP level is independent of the file transport mechanism applied underneath. Therefore, any reliable file transfer utility can be supported. We employ the file transfer protocol (FTP) and trivial file transfer protocol (TFTP). The block scheme of a SIP system based on the proposed state-sharing mechanism is illustrated in Fig. 1. The algorithm consists of three components, implemented on each host:

- SUM-to-txt converter (STC);

- file transfer script (FTS);
- txt-to-SUM converter (TSC).

The STC and FTS are running in the server that generates/updates a state (source, i.e., host 1). TSC is running in the server(s) (destination, i.e., host 2) that receive the text file containing the state update message (SUM). The SUM is created in the message handler (MH) and added to the input queue of the state manager (SM). The STC takes a SUM from the MH as an input and creates a text file as an output. The output text file is an input for the FTS component. The FTS is a simple script, which has to execute the transfer of the text file. TSC is a monitoring process, running in a destination server. It converts the contents of the text file into a SUM, whose structure is of the same type as the one on the source side. Further, the TSC adds the output SUM to the input queue of the destinations SM. The state is correspondingly updated according to the state machine of the particular call to which the SUM belongs. Thus, the final goal is achieved: servers in a state-sharing pool keep an identical image of ongoing calls states.

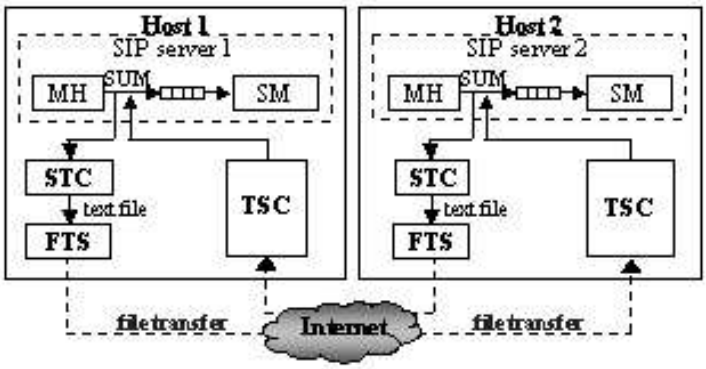

Fig. 1. The state-sharing block scheme

3 Failure-Detection and Fail-Over Subsystem

A SIP client uses its timer system for failure detection. Failure is detected when a timer per request (T_1) fires. The value of the timeout can vary. Failure detection triggers the client fail-over mechanism. A new server for serving the next SIP requests is chosen by applying a server selection policy (SSP). There are static and dynamic schemes for maintaining the list of available servers. The static mechanism uses a predefined list of servers, whereas the dynamic solution is based on the reliable server pooling (RSerPool) protocol suite [8]. RSerPool defines architecture and protocols for server pool management and client access mechanisms

yielding a highly distributed, flexible, scalable and cheap fault-tolerant solution. Both mechanisms have been developed and implemented in the testbed.

4 Testbed

An experimental testbed has been set up in order to evaluate the functionality and performance of the above state-sharing algorithm. The testbed is presented in Fig. 2. An auxiliary evaluation system has been implemented and integrated into the testbed. A server activity follows an ON/OFF process with exponential server failure and repair time distribution. The mean value of each random variable is MTTF (mean time-to-failure) and MTTR (mean time-to-repair), respectively. The Dummynet software network emulator package is employed to emulate a real Internet environment with adjustable parameters. Thus, each virtual link between two hosts is assigned a tuple of relevant network parameters, link delay (D) and link bandwidth (B).

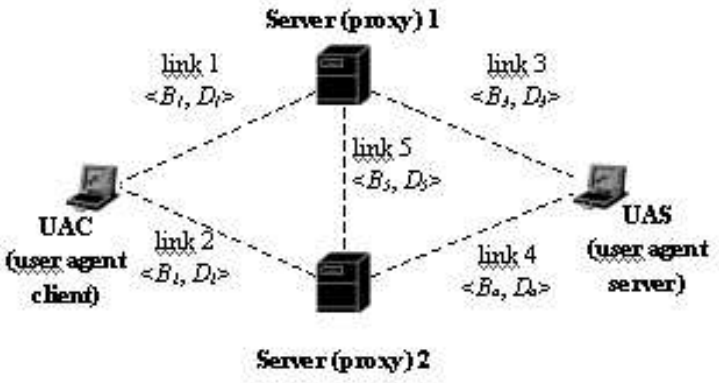

Fig. 2. The logical testbed topology

5 Analytical and Experimental Evaluation

The most important performance parameter is the session reliability [5], [6]. It is defined as the probability that an established SIP session is successfully maintained until completion. Since a BYE transaction terminates a session, reliability is the probability that the server pool successfully processes the BYE transaction. A BYE transaction must read (find) the call state in order to be processed. In the case of fail-over, the BYE request is retransmitted to another available server in the pool. The condition under which the fail-over is unsuccessful due to a missing state is as

follows (refer to Fig. 3):

$$t_{11} - t_6 < 0 \tag{1}$$

where t_{11} is the moment when server 2 receives the retransmitted BYE, and t_6 is the moment when server 2 receives the SUM from server 1. After applying a simple derivation procedure, Eq. 1 turns into:

$$T_1 + \text{callduration} < T_p \tag{2}$$

where T_1 is the failure-detection timeout per request, callduration is the time interval between the moment of sending the ACK request of the INVITE transaction and the moment of sending the BYE request of the BYE transaction at the SIP client. T_p denotes the time needed for propagating a SUM file from one server to another. This inequality unambiguously shows that if a fail-over has to be committed, then the probability of a missing state replica increases as call duration decreases and T_p (SUM propagation time) increases. Thus, for given callduration and timeout value T_1, T_p has to be minimized in order to have higher reliability. It turns out that the session reliability is a function of the parameters of the link between the servers, and the utility used for file transfer.

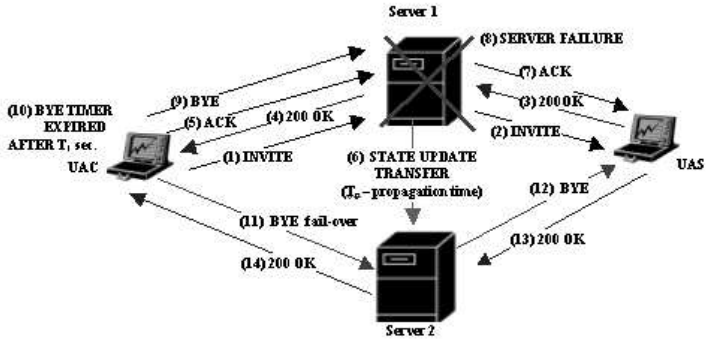

Fig. 3. A typical failure and fail-over scenario

Provided that the state-sharing mechanism is ideal (a state is reliably updated in zero time), maximum reliability is determined as follows:

$$R_{max} = R\big|_{P_r(\text{callstatefound})=1, \quad \text{at each server}} \tag{3}$$

The expression for the probability of having the call state found $P_r(callstatefound)$ can be derived using Eq. 2. It is the probability of the event derived from inequality Eq. 2:

$$P_r(\text{callstatefound}) = P_r(\text{callduration} > T_p - T_1) \tag{4}$$

T_p varies depending on the dissemination protocol and it is a function of the delay and bandwidth of the link between the servers. For the considered topology in Fig. 2, and the corresponding state-sharing mechanism, the general (approximate) expression for T_p is the following:

$$T_p = a_1 D_5 + \frac{a_2}{B_5} + a_3 \tag{5}$$

The coefficient a_1 estimates the total number of packets exchanged during the SUM transfer, a_2 is interpreted as the total number of bits exchanged and a_3 is proportional to the total message processing time at both endpoints. The coefficients a_1, a_2 and a_3 are direct measures of the communication overhead being produced by the corresponding state-sharing dissemination protocol when a state update transfer takes place. The higher these parameters are, the larger the communication overhead is between the peer servers. Assuming exponential distribution of call duration with mean $1/\mu$, the expression for the probability of having the call state found is the following:

$$P_r(\text{callstatefound}) = \begin{cases} 1, & T_p \leq T_1 \\ e^{-(T_p - T_1)\mu}, & T_p \geq T_1 \end{cases} \tag{6}$$

By observing Eq. 5 and Eq. 6, it turns out that there exist maximum values of D_5 upon which reliability starts to decrease. These are referred to as break points - D_5^{bp}. Precisely, reliability is constant and maximum in the interval [0, D_5^{bp}], where D_5^{bp} is the D_5 breakpoint with respect to a particular dissemination protocol for a given B_5. Values of D_5 greater than D_5^{bp} cause monotonous decrease in reliability. Using Eq. 5, D_5^{bp} can be found as the maximum D_5 for which the following equation is satisfied:

$$T_p\big|_{D_5 = D_5^{bp}} = T_1 \tag{7}$$

By plugging Eq. 7 into Eq. 5, the expression for D_5^{bp} is obtained as follows:

$$D_5^{bp} = \frac{(T_1 - \frac{a_2}{B_5} - a_3)}{a_1} \tag{8}$$

In Table 1, the particular values for a_1, a_2 and a_3 are given for the two dissemination protocols FTP and TFTP. They have been obtained by measuring T_p for different values of D_5, B_5 and solving linear matrix equations. It is seen that FTP causes the largest overhead exchanging around 17 messages until a state update is

Table 1. Coefficients of different state-sharing mechanisms

	FTP	TFTP
a_1[number of packets]	16.8	3
a_2[bits]	9878	2500
a_3[ms]	44	6.9

committed, whereas TFTP uses 3 packets for state update dissemination thereby causing significantly smaller communication overhead.

The D_5 breakpoints are analytically and experimentally obtained as functions of the link 5 bandwidth - B_5, and for the fixed set of system parameters given in Table 2.

Table 2. System Parameters

MTTR	MTTF	T_1	$1/\mu$
0.5s	49.5s	1s	0.5s

The D_5 breakpoint values for the two transfer protocols are given in Table teftab3.

Table 3. Delay breakpoint values for FTP and TFTP

		FTP	FTP	TFTP	TFTP
		B_5=64kbit/s	B_5=800kbit/s	B_5=64kbit/s	B_5=800kbit/s
D_5^{bp}[ms]	Analyt.	48	56	318	330
D_5^{bp}[ms]	Exper.	50	63	375	390

It is straightforward that TFTP outperforms FTP due to the much larger overhead that FTP produces. Generally, overhead slows down the data transfer. TFTP avoids large communication overhead and enables faster state update transfer while providing the required transport reliability in case of packet losses due to network congestion or low quality of wireless links. It should be kept in mind that call duration lesser than 1s is unrealistic and in real systems $1/\mu$ is much larger. Thus, the worst case has been considered here. The analytical breakpoints are slightly different from the experimental ones. This mismatch is natural as the coefficients a_1, a_2 and a_3 are obtained by measurements that are not likely to have been ideally carried out. As B_5 increases so does a breakpoint as seen from Eq. 8. That is, the higher the bandwidth, the shorter the SUM propagation time.

6 Conclusions

A SIP-specific state-sharing mechanism was proposed in this paper. It was implemented into SIP servers, which have to provide highly reliable SIP sessions in

UMTS. The main goal of this mechanism is to synchronize the state machines of all ongoing calls. Consequently, upon failure detection, the fail-over transparently resumes an ongoing service, by increasing its reliability. Failure-detection and fail-over mechanisms were also developed and along with the state-sharing mechanism implemented in a testbed. The overall fault-tolerant system has been evaluated through mathematical analysis and experiments. Results show that TFTP is more efficient dissemination protocol than FTP regarding the maximum supported delay range of the inter-server link. TFTP provides the maximum reliability in the range of up to approximately 400ms, while FTP does the same only up to approximately 60ms link delay. Both protocols show the robustness of the state-sharing mechanism and its applicability in a heterogeneous Internet environment.

Acknowledgments

The authors would like to thank Mr. Robert Seidl, Siemens, Munich, for his valuable comments. This work has been carried out within the joint Siemens-AAU research project.

References

[1] J. Rosenberg, "SIP: Session initiation protocol," Internet Engineering Task Force," RFC 3261, June 2002.

[2] 3GPP, "IP multimedia (IM) subsystem - stage 2," 3GPP," 3GPP TS 23.228, June 2001.

[3] A. Helal, A. Heddaya, and B. Bhargava, *Replication Techniques in Distributed Systems*. Kluwer Academic Publishers, 1996.

[4] G. Coulouris and T. K. J. Dollimore, *Distributed Systems: Concepts and Design, 3rd Edition*. Addison Wesley, 2001.

[5] M. Bozinovski, L. Gavrilovska, and R. Prasad, "Performance evaluation of a sip-based state-sharing mechanism," in *Proc. IEEE VTS 56th Vehicular Technology Conference (VTC) 2002*, Vancouver, Canada, 2002.

[6] ——, "Fault-tolerant SIP-based call control system," *IEE Electronics Letters*, vol. 39, pp. 254–256, Jan. 2003.

[7] C. Ross, P. Conrad, A. Jungmaier, W.-C. Sim, and M. Tuexen, "Reliable IP telephony applications with SIP using RSerPool," in *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)*, Orlando, USA, July 2002, pp. 352–356.

[8] M. Tuexen, Q. Xie, R. Stewart, M. Shore, and J. Loughney, "Architecture for reliable server pooling," Internet Engineering Task Force," draft-ietf-rserpool-arch-07.txt, Oct. 2003.