BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 60, No. 1, 2012
DOI: 10.2478/v10175-012-0014-9

The model of end-to-end call setup time calculation for Session Initiation Protocol

P.S. GUTKOWSKI and S. KACZMAREK*

Faculty of Electronics, Telecommunications and Informatics, Department of Teleinformation Networks, Gdansk University of Technology, 11/12 Gabriela Narutowicza St., 80-233 Gdansk, Poland

Abstract. End-to-end call setup delay is one of the most important grade of service (GoS) parameters for VoIP networks with Session Initiation Protocol (SIP). A typical subscriber wants to have a connection established as soon as possible. From the operator's perspective call setup time is also crucial because he needs to know how probability of successful packet transmission on SIP links is related to a call setup delay. Then he can make calculations how many resources are needed to guarantee such a transmission. In this paper we proposed the model for end-to-end call setup time calculation. The fundamentals of the model are to check all or almost all (highly probable) possible situations during the call setup. Formulas for calculation are included and results of numerical calculation are presented. Comparison against the values obtained from the simulation shows that it is possible to make calculations of end-to-end call setup time and prepare the discrete cumulative distribution function for a trapezoid model in reasonable time using the proposed model.

Key words: SIP, VoIP, model, GoS.

1. Introduction

SIP (Session Initiation Protocol) is a well-known protocol. The second version dates back to June 2002 and was standardized by Internet Engineering Task Force in RFC 3261 [1]. According to that document, SIP is an application layer protocol for creating, modifying and terminating sessions with one or more participants. These sessions include Internet telephone calls, multimedia distribution and multimedia conferences. It is mainly used for call control and signaling purposes, SIP is still gathering the interest of telecommunication equipment vendors, operators and academic researchers.

There are many workable applications implementing SIP elements functionality such as user agents and proxy servers. Most of them, for example *osip* [2] licensed on GPL [3], are free software and it is also the reason why SIP gains popularity. SIP architecture is as open as it can be and it could be easily extended. Moreover, there are many SIP-related drafts and RFC documents connected with specific areas of a protocol like NAT traversal, audio-video media transport, security issues, etc. It leads to tens of RFCs but they still do not answer to all of the needs. One of the important protocol areas which needs more attention is SIP traffic engineering. So far there were few approaches to that problem, for example SIP-Q [4]. Also, some models of SIP network behavior were developed [5–10].

In addition, there were research studies in European grants, for example EuQoS [11]. Problems such as the influence of transport layer protocol for SIP were also studied [12]. One of the main goals of the research was to describe GoS and QoS parameters.

For SIP protocol we can define such GoS parameters as: end-to-end call setup delay, call termination delay, INVITE message mean delay, etc. In addition, we can specify QoS parameters such as SIP packet/call loss percentage and mean SIP message delay between two nodes. From the caller's perspective the most important parameter is a call setup delay because a typical subscriber wants to establish the connection with the called party as soon as possible. From the operator's perspective the setup time is also crucial, as he needs to know how the probability of successful packet transmission on SIP links is related to call setup delay. It enables him to make calculations how much resources are needed to guarantee such a transmission. This is the reason why we focus on that problem in this paper. The remaining part of the paper is organized as follows: in Sec. 2 we present several SIP protocol features which are crucial assumptions for creating the traffic model (transaction mechanism), in Sec. 3 we describe the model which can be used for calculation of end-to-end call setup time. Section 4 covers numerical results for the proposed model, which are later compared to simulation results. Section 5 is the conclusion.

2. Sip protocol architecture

2.1. Transaction mechanism. SIP is a transactional protocol. It means that message flows are gathered in series called transactions. Each request and one or few responses to that request form a transaction. There is also one more term related to the message flow called a dialog. Transaction is the idea that is narrower than the dialog because dialog contains few transactions. A typical dialog consists of two transactions (one for call setup and another one for call termination).

*e-mail: kasyl@eti.pg.gda.pl

The most important fact is that the transaction mechanism controls retransmission of undelivered packets.

Because of limited space in this paper we do not present detailed information about the transaction mechanism. It is well covered in RFC 3261 [1].

From our point of view the most important elements are two timers related to INVITE message retransmissions and Invite Client Transaction:

Timer A – which controls request retransmissions (TA – it starts from default value $T1 = 0.5$ sec. and is doubled on every retransmission),

Timer B – which controls transaction timeouts (TB – the transaction is destroyed after TB = $64T1$ and the call is lost).

Moreover, there are OK message retransmissions generated by the transaction user (TU) called party side when ACK is not received by the server side. They are controlled by Timer G which initial value is set to $T1$.

Call termination time goes beyond the scope of this paper.

2.2. Trapezoid model. The most typical SIP scenario for call setup is trapezoid model involving four actors: two SIP user agents and two SIP proxy statefull servers between them. The scenario is presented in Fig. 1.

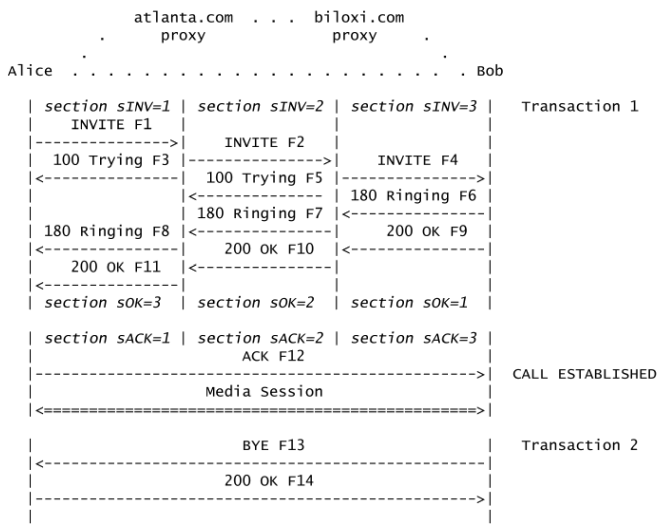

Fig. 1. Typical SIP scenario – trapezoid model

We take into consideration only INVITE transactions and ACK messages associated with them. Then we have in our study a three-way handshaking problem. The whole network is divided into *sections*: the first section is Alice ↔ atlanta.com, the second is atlanta.com ↔ biloxi.com, and the third is biloxi.com ↔ Bob (*sections*=3 for trapezoid model). For each section we have INVITE, OK and ACK messages.

In our example for successful call setup INVITE message must traverse through *sINV* sections 1-3 towards Bob, while in the reverse direction OK message must pass *sOK* sections 1-3 to reach Alice. Also ACK messages must pass from *sACK* sections 1-3 to reach Bob. It is crucial that end-to-end call set-up time does not depend on provisional responses (100 TRYING and 180 RINGING responses) because they belong to 1xx class responses, which are informational responses. Their main effect is a change from Calling to Proceeding state on ICT diagram. When the client transaction fails to receive any 1xx message in Timer A time, it will perform the retransmission. However, it will not exert any influence on call setup time, as the request have reached the destination, yet the information about that fact was discarded. When 1xx is lost it generates additional traffic of the messages but the proposed model does not operate on link load, instead of that it will use probability of successful transmission.

We also make an assumption that the call is answered immediately and there is no delay between INVITE and OK on the called party side (we assume an immediate answer).

In each section there could be retransmission of INVITE messages and it depends on probability of successful transmission in section *sINV* denoted as ps_{sINV}. Retransmissions of OK messages are generated by Bob's terminal, they happen when ACK message is not received.

Since the response was a 2xx, the ACK is not considered to be part of the transaction (SIP protocol architecture assumption [1]). It would be necessary to check if ACK reaches the destination, otherwise the call is lost. Retransmissions of ACK messages are triggered by OK message retransmissions received by Alice's terminal – if there is ACK retransmission a corresponding OK retransmission must occur. That is because nodes do not retransmit ACK messages themselves. ACK retransmission is triggered by OK message retransmission and it can happen several times.

3. The model for numerical calculations of end-to-end call setup time

3.1. Possible situations during the call setup and associated limits on CPU operations. The fundamentals of the model are to check all or almost all possible situations during call setup (those with highest probability, quantity depends on CPU power applied for calculations). If there are no retransmissions, the calculation is trivial. Otherwise, it is more complex because there are many more situations to check.

During call setup several non-overlapping situations may occur. They are shown in Fig. 2, where n is the number of retransmissions from an end-to-end perspective. According to Fig. 2, we have four areas of events:

I – there is no retransmission ($n = 0$, calculating probability of that situation is trivial, as it is the multiple of successful transmission probability in each section),

II – there are n retransmissions ($n \geq 1$, retransmissions could be in any section; for example, we have 4 retransmissions from end-to-end: 2 in section *sINV*=1, 1 in section *sINV*=3 and 1 in section *sOK*=1. The algorithm checks all possible combinations of retransmissions from $n = 1$ to $n_{\max}$, where maximum value of n is specified in the configuration for application),

III – call is lost:

– because time is too long and it triggers Timer B (for ICT transactions); the worst case (the biggest time values) is from the point of view of session initiator, we can specify

such a condition: if *end-to-end call setup time* $\geq$32 sec., then the call is lost.

– because neither INVITE, nor OK or ACK reach their destination,

Θ – error area related to two facts:

Θ_1 – there is only a discrete set of timer values – so we can approximate the value of the end-to-end call setup time only for the worst case (approximation from up-biggest value of time) or for the best case (approximation from down-smallest value of time) or "something" between these cases (for example arithmetic mean value).

Θ_2 – we have limited CPU power of the machine on which that algorithm is to be running and for low values of probability of successful transmission we cannot check all the situations – there are too many retransmissions; still, we make calculations only up to $n_{\max}$.

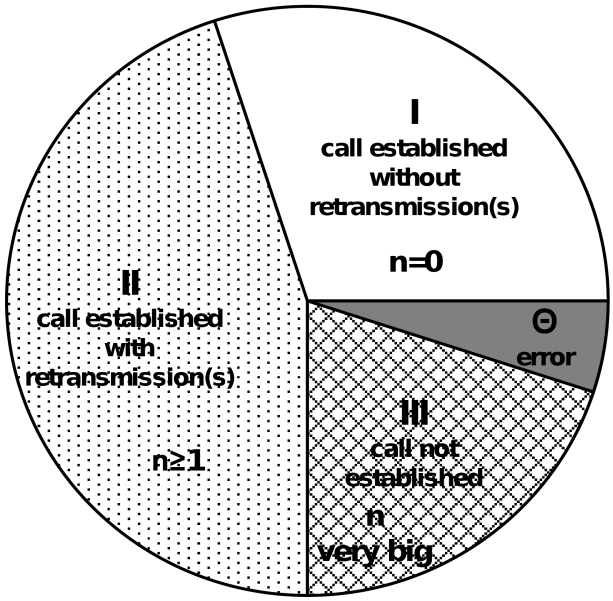

Fig. 2. Areas of possible situations during call setup (n is the number of retransmissions and it is growing counter clockwise from I to III)

3.2. Handling retransmissions. For instance, if we consider only one retransmission ($n = 1$), we will have 2 root causes of retransmission and 4 cases because:

– INVITE retransmission could happen in one of the *sINV* sections (for trapezoid we have 3 cases – INVITE retransmission can happen in the first, second or third section),

– OK or ACK retransmission in all other sections (1 case).

In general, for calculating the number of possible n retransmissions in sections the following formula must be used:

$$C^n_{sections} = \begin{pmatrix} sections + n - 1 \\ sections - 1 \end{pmatrix} = \begin{pmatrix} sections + n - 1 \\ n \end{pmatrix} = \frac{(sections + n - 1)!}{n!(sections - 1)!}. \quad (1)$$

These sequences for low values of n are listed in Table 1.

Table 1

Possible location of retransmissions in 4 sections (3 sections for INVITE messages +1 section for OK/ACK retransmission) for different number of N

$N = 1$	$N = 2$	$N = 3$
1 0 0 0	2 0 0 0	3 0 0 0
0 1 0 0	1 1 0 0	2 1 0 0
0 0 1 0	1 0 1 0	2 0 1 0
0 0 0 1	1 0 0 1	2 0 0 1
	0 2 0 0	1 2 0 0
	0 1 1 0	1 1 1 0
	0 1 0 1	1 1 0 1
	0 0 2 0	1 0 2 0
		1 0 1 1
		1 0 0 2
		0 3 0 0
		0 2 1 0
		0 2 0 1
		0 1 2 0
		0 1 1 1
		0 1 0 2
		0 0 3 0
		0 0 2 1
		0 0 1 2
		0 0 0 3

As it is shown in Table 1, the bigger number of n, the more complex it gets. We make a general assumption that end-to-end call setup time depends exclusively on the retransmission time, which is derivative of successful transmission probability. The packet transmission time is much shorter than the retransmission time, so we may assume that the transmission time ≈ 0. In this paper we concentrate on cases where the links are heavily loaded with traffic. Transmission time is an issue for another study. n is the sum of the values

$$n = \sum_{sINV=1}^{sections} n_{sINV} + n_{OK}, \quad (2)$$

where n_{sINV} is the number of INVITE message retransmissions in section s_{INV}, n_{OK} – the number of retransmitted OK messages.

We denote the time spent in section *sINV* as t_{sINV} for INVITE requests. For OK responses we take time t_{OK}. The time for ACK messages is assumed to be 0 because retransmission time of ACK messages has already been included in OK retransmission time (there are no timers directly triggering the transmission of ACK messages).

The time spent in each section depends on the number of retransmissions in a particular section. For example, in the case of two INVITE request retransmissions in section 1, we have $t_{1INV} = 0.5 + 1 = 1.5$ seconds. The set of possible values of t_{sINV} is {0, 0.5, 1.5, 3.5, 7.5, 15.5, 31.5} and it is calculated using formula (3).

$$t_{sINV}(n_{sINV}) = \sum_{i=0}^{n_{sINV}-1} T1 \cdot 2^i. \quad (3)$$

For OK messages we have a similar equation

$$t_{OK}(n_{OK}) = \sum_{i=0}^{n_{OK}-1} \min(T1 \cdot 2^i; T2), \tag{4}$$

where $T2 = 4$ seconds according to RFC 3261 [1].

The elementary end-to-end call setup time t_e equals to

$$t_e = \sum_{sINV=1}^{sections} t_{sINV}(n_{sINV}) + t_{OK}(n_{OK}). \tag{5}$$

By incorporating (3) and (4) into (5) we obtain

$$t_e = \sum_{sINV=1}^{sections} \sum_{i=0}^{n_{sINV}-1} 0.5 \cdot 2^i + \sum_{i=0}^{n_{OK}-1} \min(TI \cdot 2^i; T2), \tag{6}$$

but it is for one particular situation only. As we mentioned before, there are many more situations related to retransmission.

Each elementary situation is accompanied by a specific probability, which we can calculate. For that element we also calculated the elementary time using formula (6).

3.3. $E(T_{call})_b$, $E(T_{call})_t$, $E(T_{call})_m$ calculation. Since we have separable non-overlapping events, we can calculate $E(T_{call})_b$, i.e. it means call setup time under the condition that call setup is successful. The "call" subscript condition means that we do not sum up elements from area III. "b" index placed after the bracket means that the time is approximated from the bottom values.

Due to the fact that we approximate from the bottom values, for $n = 0$ the setup time is 0 and we must only start summing the values up from $n = 0$ in denominator – it is required to include area I from Fig. 2, as the nominator for $n = 0$ equals 0. The algorithm must perform the necessary checking during the calculation and classify the results into corresponding areas.

$$E(T_{call})_b = \frac{\sum_{n=1}^{n_{\max}} \sum_{e=1}^{C^n_{sections+1}} (t_e \cdot p_e)}{\sum_{n=0}^{n_{\max}} \sum_{e=1}^{C^n_{sections+1}} p_e} + E(T_{call}(\Theta))_b, \tag{7}$$

where t_e is the elementary end-to-end call setup time and p_e is the probability of that situation. p_e is calculated in the following manner

$$p_e = \prod_{sINV=1}^{sections} ((1 - ps_{sINV})^{n_{esINV}} \cdot ps_{sINV}) \cdot \left(1 - \prod_{sOK=1}^{sections} ps_{sOK} \prod_{sACK=1}^{sections} ps_{sACK}\right)^{n_{eOK}} \cdot \prod_{sOK=1}^{sections} ps_{sOK} \prod_{sACK=1}^{sections} ps_{sACK}, \tag{8}$$

where ps_s is the assumed probability of successful transmission in section s. The value denoted as n_{es} is the number of retransmissions in section s for elementary situation e. Values are calculated with the use of recursive algorithm for combination with repetitions. The number of retransmissions n_{es} is varying for all the cases, while probabilities of successful retransmission ps_s are the same in all cases.

Please note that in (7) the reason for adding 1 to the *sections* is due to end-to-end treatment of 2xx and ACK messages. The retransmission of INVITE can occur in any of the *section* and we need to add 1 because of the special treatment of 2xx response and its ACK (we need to look from end-to-end perspective because when 2xx response or ACK is lost in any section, 2xx need to be retransmitted by UAS).

Bearing in mind (6) and (8), we can denote (7) as (9). Since during the calculation of (9) we have only a discrete set of timer values, we made approximation from the bottom values. Approximating from top values ($E(T_{call})_t$) is fairly easy because while summing up time values we must take the upper timer limit. For example, in the case of three retransmissions in the same section, third retransmission can happen after $0.5+1+2 = 3.5$ sec. but not later than $0.5+1+2+4 = 7.5$ sec. Thus, instead of 3.5 sec. (approximation from the bottom value), we need to take 7.5 sec. (approximation from the top value). In formula (10) we must sum up the values from $i = 0$ up to n_{es} instead n_{es-1}. As the formula is extensive we do not expand p_e this time. Besides, we need to include area I in nominator because we have 0.5 sec. instead 0 for area I.

We can also make calculations for arithmetic mean ($E(T_{call})_m$) of timer intervals (11).

3.4. Probability distribution and cumulative distribution. During the main loop of performing calculation it is necessary to store data in a histogram. Then we can make cumulative distribution plot of end-to-end call setup time $E(T_{call})_b$, $E(T_{call})_t$, $E(T_{call})_m$. Two assumptions which were made earlier proved very helpful:

The timer sum values have discrete form. The result of any timer value sum always gives integer number either integer number $+0.5$. Then histogram and table for distribution should be performed with 0.5 time step for bottom and top approximation. In the case of the mean approximation, we need to have two times bigger resolution and make histogram with 0.25 time step.

The results of *time* calculation do not exceed 32 sec.

According to these assumptions we only need the tables with $32/0.25 = 128$ elements.

4. Numerical results

Under the assumptions made in section 3 the application in C++ language was written. It is possible to perform numerical calculations for trapezoid model and $n_{\max}$=12 in less than one minute on a standard personal computer (Intel I7). Similarly, it is possible to calculate end-to-end call setup time for more complex network architectures using that application.

Numerical calculations were performed for two cases with different ps_s probabilities of a successful transmission. Probability values were taken from the simulation, which was performed with the use of a dedicated SIP network simulator called SIPSIM. SIPSIM was prepared by the authors and it is a discrete event simulator, which enables to simulate different

network architectures specified by configuration file (nodes, links, generators, queues, etc.). For the purpose of this paper we used trapezoid model and we placed generators on both ends to ensure symmetrical traffic flow. Each link has the same bandwidth and we use exponential SIP traffic generators with different call generation intensities to obtain different probabilities of successful transmission in each case. For the purpose of the simulation result discussion we used 0.95 as a confidence interval value. The entire calculation process is presented in Fig. 3.

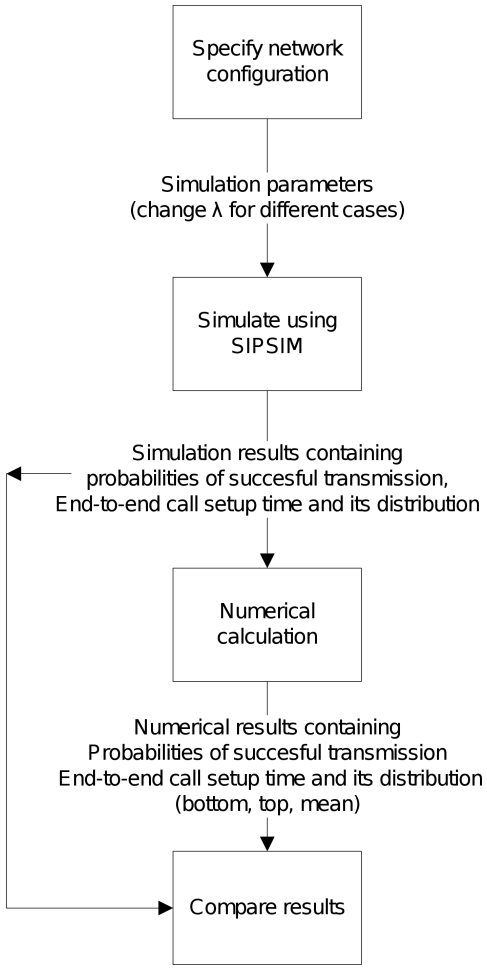

Fig. 3. The process used for obtaining numerical results

Two different cases were taken into account:

a) $ps_{1INV} = ps_{1OK} = ps_{1ACK} = 0.845725\ (\pm\ 0.002156)$
$ps_{2INV} = ps_{2OK} = ps_{2ACK} = 0.966115\ (\pm\ 0.001986)$
$ps_{3INV} = ps_{3OK} = ps_{3ACK} = 0.971626\ (\pm\ 0.002024)$

b) $ps_{1INV} = ps_{1OK} = ps_{1ACK} = 0.760943\ (\pm\ 0.004401)$
$ps_{2INV} = ps_{2OK} = ps_{2ACK} = 0.870967\ (\pm\ 0.004196)$
$ps_{3INV} = ps_{3OK} = ps_{3ACK} = 0.899251\ (\pm\ 0.003190)$

Links for case b are more loaded with traffic than in case a (link load $\rho_b \approx 0.55$ vs. $\rho_a \approx 0.33$). Please note that area where retransmissions appear has big dynamic – relatively small link load increase implies relatively significant probability of successful transmission change. We do not go into details because link load influence on probability of successful retransmission is topic for dedicated paper.

For each case three different $n_{\max}$ values were used in a numerical calculation. The calculations were made for bottom (9), top (10) and mean (11) approximation and they are compared to simulation results ($E(T_{call})_{sim}$) in Table 2 and Table 3.

Table 2
Numerical results of $E(T_{call})_b$, $E(T_{call})_t$, $E(T_{call})_m$ *case a*

$n_{\max}$	$E(T_{call})_b$ [sec]	$E(T_{call})_m$ [sec]	$E(T_{call})_t$ [sec]	$E(T_{call})_{sim}$ [sec]
7	0.673	1.955	3.236	1.201±0.06
8	0.680	1.963	3.244	1.201±0.06
9	0.683	1.967	3.246	1.201±0.06

Table 3
Numerical results of $E(T_{call})_b$, $E(T_{call})_t$, $E(T_{call})_m$ *case b*

$n_{\max}$	$E(T_{call})_b$ [sec]	$E(T_{call})_m$ [sec]	$E(T_{call})_t$ [sec]	$E(T_{call})_{sim}$ [sec]
11	3.211	4.928	6.516	5.501±0.260
12	3.218	4.929	6.516	5.501±0.260
13	3.219	4.929	6.516	5.501±0.260

As we can see in Table 2, there is an insignificant difference (less than 1%) between numerical results for $n_{\max} = 8$ and $n_{\max} = 9$ (case a), also for $n_{\max}$=12 and $n_{\max}$=13 (case b). Θ_2 does not provide any high probable events in such situations. It is unnecessary to perform calculations for greater $n_{\max}$ values to obtain greater accuracy. The difference occurs on the third place after the dot or further and we may assume that $\Theta_2 \approx 0$.

$$E(T_{call})_b = \frac{\sum_{n=1}^{n_{\max}} \sum_{e=1}^{C^n_{sections}+1} \left(\sum_{sINV=1}^{sections} \sum_{i=0}^{n_{esINV}-1} T1\cdot 2^i + \sum_{i=0}^{n_{OK}-1} \min(T1\cdot 2^i; T2) \right)}{\sum_{n=0}^{n_{\max}} \sum_{e=1}^{C^n_{sections}+1} \prod_{sINV=1}^{sections} ((1-ps_{sINV})^{n_{esINV}} \cdot ps_{sINV})} \tag{9}$$

$$\cdots \frac{\prod_{sINV=1}^{sections} ((1-ps_{sINV})^{n_{esINV}} \cdot ps_{sINV}) \cdot \left(1 - \prod_{sOK=1}^{sections} ps_{sOK} \prod_{sACK=1}^{sections} ps_{sACK}\right)^{n_{eOK}} \prod_{sOK=1}^{sections} ps_{sOK} \prod_{sACK=1}^{sections} ps_{sACK}}{\left(1 - \prod_{sOK=1}^{sections} ps_{sOK} \prod_{sACK=1}^{sections} ps_{sACK}\right)^{n_{eOK}} \prod_{sOK=1}^{sections} ps_{sOK} \prod_{sACK=1}^{sections} ps_{sACK}} + E(T_{call}(\Theta))_b$$

$$E(T_{call})_t = \frac{\sum_{n=0}^{n_{\max}} \sum_{e=1}^{C^n_{sections}+1} \left(\sum_{sINV=1}^{sections} \sum_{i=0}^{n_{esINV}} T1{\cdot}2^i + \sum_{i=0}^{n_{eOK}} \min(T1{\cdot}2^i; T2) \right) p_e}{\sum_{n=0}^{n_{\max}} \sum_{e=1}^{C^n_{sections}+1} p_e} + E(T_{call}(\Theta))_t, \tag{10}$$

$$E(T_{call})_m = \frac{\sum_{n=0}^{n_{\max}} \sum_{e=1}^{C^n_{sections}+1} \left(\sum_{sINV=1}^{sections} \frac{\sum_{i=0}^{n_{esINV}-1} T1{\cdot}2^i + \sum_{i=0}^{n_{esINV}} T1{\cdot}2^i}{2} + \frac{\sum_{i=0}^{n_{eOK}-1} \min(T1{\cdot}2^i; T2) + \sum_{i=0}^{n_{eOK}} \min(T1{\cdot}2^i; T2)}{2} \right) p_e}{\sum_{n=0}^{n_{\max}} \sum_{e=1}^{C^n_{sections}+1} p_e} + E(T_{call}(\Theta))_m. \tag{11}$$

Values of mean time $E(T_{call})_{sim}$ from the simulation are much closer to the numerical results of $E(T_{call})_m$ for case b (5.50 vs. 4.92) than for case a (1.2 vs. 1.96). In both cases simulation results $E(T_{call})_{sim}$ are between $E(T_{call})_b$ and $E(T_{call})_t$.

Cumulative distributions of call setup time were also calculated. Calculation result is shown in Figs. 4 and 5. When comparing Fig. 4 with Fig. 5, we notice that the nature of distribution is similar in both cases but *time* values are much bigger in the case b.

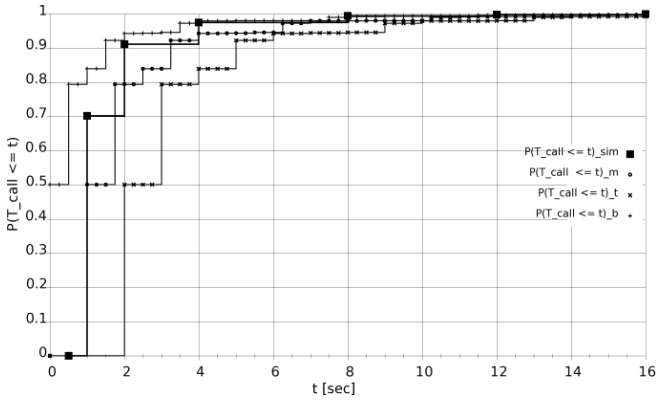

Fig. 4. Cumulative discrete distribution functions of T_{call} case a

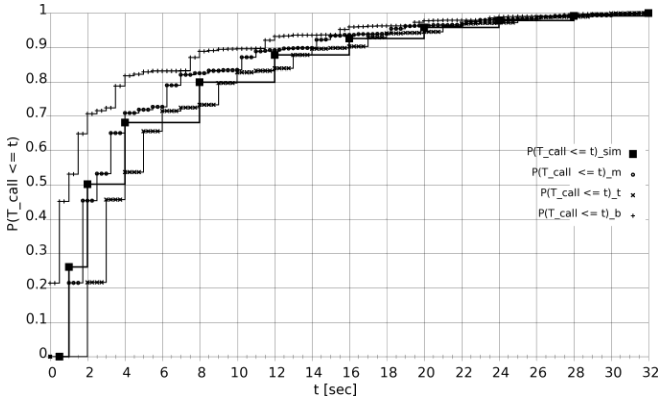

Fig. 5. Cumulative discrete distribution functions of T_{call} case b

As expected, $P(T_{call} \leq t)_m$ in Fig. 4 starts from higher value of probability (about 0.5) than $P(T_{call} \leq t)_m$ in Fig. 5 (about 0.2). It is the probability of call setup without any retransmissions (area I). Since we have bigger values of successful transmission probabilities in case a, we have lower time values.

There is a big difference between values approximated from the bottom and from the top values, in particular for low time values. Curves for mean estimation are close to the simulation results. It clearly demonstrates that in engineering practice mean approximation should be used.

5. Conclusions

In this paper we have proposed the model and algorithm for end-to-end call setup delay time calculation assuming that the call setup is successful. Numerical results show that it is possible to make calculations of end-to-end call setup time and prepare its cumulative distribution function in reasonable time, for trapezoid model. Obtained numerical results for mean approximation are close to the simulation results.

Still, there are a few disadvantages of the algorithm: in the case of very low ps_s values and networks with large number of nodes it is very time-consuming and in fact almost impossible to make such calculations, the error area θ is not described well in such cases. The model is intended for use in networks with small number of SIP nodes for approximating call setup time in the presence of SIP messages retransmission.

The model is especially helpful when dealing with highly loaded and congested networks. In networks with low traffic values there are usually no SIP retransmissions and call setup time depends mainly on transmission time. If end-to-end call setup time value is below 0.5 sec. there is a small difference from the subscribers' perspective because they usually do not notice the negative impact of such latency.

The knowledge of end-to-end call setup time and its distribution function is crucial since it well describes GoS. Also, it allows operator tracking dependency between other QoS parameter – probability of successful transmission ps_s.

In the future, the model can be extended in order to create the possibility of performing calculation for other GoS parameters such as call termination delay. That parameter is also very important from the operator's perspective (one of the possible applications may include call billing problem prevention).

REFERENCES

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, *SIP: Session Initiation Protocol*, *RFC 3261*, 2002.

[2] *Osip SIP implementation*, http://www.gnu.org/software/osip/ (2011).

[3] *GPL license*, http://www.gnu.org/licenses/licenses.html#GPL (2010).

[4] S. Salsano and L. Veltri, "QoS control by means of COPS to support SIP-based applications", *IEEE Network* 2, 27–33 (2002).

[5] L. Alcuri and M.L. Fasciana, "Transport service for session initiation protocol in SIP-T scenarios", *IEEE Proc. 10th Asia-Pacific Conf. on Communications* 1, 123–127 (2004).

[6] H. Fathi, S.S. Chakrabroty, and R. Prasad, "On SIP session setup delay for VoIP services over correlated fading channels", *IEEE Trans. on Vehicular Technology* 55 (1), 286–295 (2006).

[7] P. Gutkowski and S. Kaczmarek, "Traffic model for signaling in packet network with SIP protocol", *Internet* 2006, 275–298 (2006), (in Polish).

[8] B. Rong, J. Lebau, M. Bennani, M. Kadoch, and A.K. Elkaheem, "Modeling and simulation of traffic aggregation based SIP over MPLS network architecture", *IEEE Proc. 38th Annual Simulation Symposium* 1, CD-ROM (2005).

[9] J.-S. Wu and P.Y. Wang, "The performance analysis of SIP-T signaling system in carrier class VoIP network", *AINA* 39–44 (2003).

[10] P. Gutkowski and S. Kaczmarek, "Service time distribution influence on end-to-end call setup delay calculation in networks with session initiation protocol", *IEEE, Proc. 1st Eur. Teletraffic Seminar* 1, 37–42 (2011).

[11] *EuQoS*, http://www.euqos.eu/ (2011).

[12] G. Camarillo R. Kantola, and H. Schulzrinne, "Evaluation of transport protocols for the SIP initiation protocol", *IEEE Network* 5, 40–46 (2003).