Profit-maximizing trustworthiness level of composite systems

Costas Kalogiros, Michalis Kanakakis, Shenja van der Graaf, Wim Vanobberghen,

Athens University of Economics and Business
GR1034, Athens, Greece
76, Patission Str.
T: 0030-2108203154
{ckalog, kanakakis}@aueb.gr

iMinds-SMIT, Vrije Universiteit Brussel
Pleinlaan 9, 1050 Brussels, Belgium
T: 0032-470509885
{shenja.vandergraaf}@iminds.be and {wim.vanobberghen}@vub.ac.be

Abstract. Service providers face the ever-increasing problem of meeting customer expectations while maximizing profits. This optimal balance is very important for delivering better service quality to users and keeping costs under control through efficient resource allocation. In this paper we suggest optimal strategies for managing system trustworthiness in two different contexts. In the first one the provider has limited information about the users' trustworthiness preferences, which have to be satisfied on every transaction. In the second context, the provider knows what the effect of possible outcomes on customer's trust level and, given that the customer will perform a certain number of transactions, would like to know whether the system trustworthiness should be managed at any point in time in order to meet customer's expectations in a cost-effective way. The optimality of the proposed strategies is demonstrated via both analytical techniques and simulations.

Keywords: Trustworthiness management, Trust management, optimal strategies, trust, trust computational model, run-time, composite systems, economics of security

1 Introduction

Most user interactions on the Internet, e.g., checking email correspondence, sharing thoughts with online friends, buying from digital stores, watching movies online usually involve more than one provider, even though this is not directly observable by the average end-user. Retail providers of ICT (Information and Communications Technology) services, for example, are increasingly relying on cloud computing providers for computational, storage and networking services.

It is expected, however, that the outsourcing trend will not diminish in the future. Recent initiatives, such as smart transportation systems, ambient-assisted living, etc., follow the paradigm of Service-Oriented Architectures in which application components are orchestrated to provide services to other components over a network. Each of those components can belong to different providers, and in most cases, supply is under competition.

While the popularity of cloud computing is mostly attributed to reduced average cost, elasticity and reliability stemming from economies of scope and scale, the increased interdependencies and the inability to control all operational aspects increase the complexity for providers of composite services to meet security and other trustworthiness objectives. A system's trustworthiness can be assessed with several metrics, such as mean availability, mean response time, minimum encryption, etc. For a detailed analysis of trustworthiness metrics, including security ones, the interested reader is redirected to [1]. We should note that a trustworthiness metric can be objectively measured, in the sense that two separate entities can agree on a single formula and given the same (or a sufficiently large) set of run-time observations/evidences they would eventually reach the same outcome.

This paper focuses at the production phase of such composite systems and, more specifically, how retail providers should meet customer expectations while maximizing profits. This optimal balance is very important for delivering better service quality to users, increasing market share by gaining users' trust, mitigating adverse effects and keeping costs under control through efficient resource allocation. However, achieving this balance is not an easy task for the following reasons. The retail provider has no perfect information about the actual trustworthiness of each individual component/service instance and the customer expectations are usually unknown.

The provider could compare the trustworthiness of candidate components by querying a marketplace that carries detailed trustworthiness certificates, such as the ones described in [2]. Furthermore, the provider could offer SLA's (Service Level Agreements) where the exact security/trustworthiness levels are described as a set of metrics and their respective target values. Then we could assume that the user would not trust the provider again in the future if any threshold value was not met. In this paper , however, we will assume that the retail provider does not want to offer SLAs.

More specifically in Section 2 we assume that the provider is paid for each successful transaction only and has some information about the distribution of users' trustworthiness preferences. Its purpose is to maximize the expected profits by finding the component to replace a failed one. While the authors of [3] suggest a Genetic Algorithm for selecting the components of a system so that a function of cost and an aggregate metric of trustworthiness is maximized, they ignore user's expectations.

In Section 3 we will assume that the user will interact with the provider's system several times and the number of transactions is known in advance (i.e., is the only term of the contract). For example, a bank manager that wants to process all saving accounts (e.g., calculate interests etc.) at the end of the day and enters into a monthly contract with a cloud computing provider. Whenever the cloud provider believes that the customer's trust is lower than a certain threshold the former could make the necessary changes to system in order to regain customer's trust after a few transactions.

The optimal changes that should take place at any point in time are based on the finite-stage dynamic programming model of [4], which has been adapted so that changes are restricted by the available components (instead of assuming that trustworthiness is a continuous function of effort which is more suitable for services offered by humans). In order to estimate the current user's trust level we employ a trust computational model that has been described and validated in [5]. Finally, we conclude the paper with summary and possible future extensions in Section 4.

2 Managing system trustworthiness for individual transactions

2.1 The model

We assume that whenever the system during a particular transaction fails to meet user expectations regarding response time (e.g., a deadline specified in SLA or system performance exceeds user's patience) then the user has the ability to cancel the transaction and pay nothing to the provider. Otherwise, she pays r units to the provider. The exact user patience is unknown to the provider. Suppose, however, that the provider knows that user's patience T is exponentially distributed with mean $1/\beta$, so that $P(T \geq t) = e^{-\beta t}$.
The provider is interested in maximizing her expected profit from each transaction. She is able to monitor the behaviour of all system components at run-time and thus can identify when a particular component is unavailable; the adverse behaviour we are focusing on this section. Let system s perform functions $1, \dots, f$ and assume that l components are capable of performing function m $(m \leq f)$. These components, depending on the context, can be machines running software instances, networking assets (e.g., routers), as well as, individuals (such as personnel performing a task) and their equipment. Each candidate component $g \leq l$ has known trustworthiness metrics and cost c_g, which is paid to the suppliers only for successfully completed transactions. This information can be retrieved from a marketplace of alternative subsystems or supplied by the provider itself.

Furthermore, we denote with $a_g \geq 0$ the probability that the component g will be functioning after being installed. The mean delay of component g to produce the required output is D_g, which is assumed to be exponentially distributed with parameter λ_g, while D_g, D_h are independent $\forall g \neq h$. The fixed time that is needed for integrating any new component to the system and checking its availability is denoted with $d > 0$; there is always the possibility that the newly deployed component is found unavailable after being installed and another component has to be deployed.

At t_0 a transaction starts and assume that the provider checks every z time periods (e.g., seconds) whether all components are in healthy condition, or unavailable due to an attack, or failure. Suppose also that at t_{n-1} all components were running but at the next inspection time (t_n), and before the transaction in question has been concluded or the user patience is exhausted, the provider finds component k to be unavailable. Her options would be to either replace it with a new one or do nothing (if for example the expected transaction revenues don't recover the expected costs).

We have the following theorem:

Theorem 1: The provider should try candidate components in the order $1,2,\dots,l$ provided that $W_1 \geq W_2 \geq \dots \geq W_l \geq 0$, where $W_g = \frac{(r-c_g)a_g\frac{\lambda_g}{\lambda_g+\beta}\mathrm{e}^{-\beta d}}{1-(1-a_g)\mathrm{e}^{-\beta \mathrm{d}}}$.

Proof: If the provider selected component g and given that at time t_n the user's patience was not exhausted, then the probability that the both the component integration and the transaction will be successfully completed is given by:

$$Pr[D_g = \min\{D_g, T\}\,|T > d] = Pr[D_g = \min\{D_g, T\}\,\&T > d]/Pr[T > d] \qquad (1).$$

Given that at time t_n the user's patience is not exhausted, then the probability that the user will still be waiting for the outcome after d time units is given by:

$$Pr(T > t_n + d|T > t_n) = P(T > d) = e^{-\beta d} \qquad (2).$$

Furthermore $D_1, D_2, \dots, D_l$ and T are independent exponentially distributed random variables with rate parameters $\lambda_1, \lambda_2, \dots, \lambda_l$, and β respectively, the $\min\{D_g, T\}$ is also an exponentially distributed random variable with rate parameter $\lambda_g + \beta$. Thus

$$Pr\big(D_g = \min\{D_g, T\}\,|T > t_n + d\big) = \lambda_g/(\lambda_g + \beta) \qquad (3).$$

Substituting Eq. (2) and Eq. (3) into Eq. (1) results in

$$Pr\big[D_g = \min\{D_g, T\}\,\&T > t_n + d\big] = \frac{\lambda_g}{\lambda_g + \beta}e^{-\beta d}$$

The provider should replace component k with g instead of h if the following inequality holds:

$$\begin{aligned}(r - c_g)a_g\frac{\lambda_g}{\lambda_g + \beta}\mathrm{e}^{-\beta d} + (r - c_h)(1 - a_g)a_h\frac{\lambda_h}{\lambda_h + \beta}\mathrm{e}^{-2\beta d} \\ \geq\; (r - c_h)a_j\frac{\lambda_h}{\lambda_h + \beta}\mathrm{e}^{-\beta \mathrm{d}} + (r - c_g)(1 - a_h)a_g\frac{\lambda_g}{\lambda_g + \beta}\mathrm{e}^{-2\beta d}\end{aligned}$$

, where $a_g\frac{\lambda_g}{\lambda_g+\beta}\mathrm{e}^{-\beta d}$ is the probability that the component g is found to be available (after d time units) and produced its output before the user's patience had been exhausted. With simple algebra transformations and rearrangements we have the following inequality:

$$(r - c_g)a_g\frac{\lambda_g}{\lambda_g+\beta}\mathrm{e}^{-\beta \mathrm{d}}\big[1 - (1 - a_h)\mathrm{e}^{-\beta \mathrm{d}}\big] \geq (r - c_h)a_h\frac{\lambda_h}{\lambda_h+\beta}\mathrm{e}^{-\beta \mathrm{d}}\big[1 - \big(1 - a_g\big)\mathrm{e}^{-\beta \mathrm{d}}\big] \qquad (4)$$

Since $\beta > 0$, $d > 0$ we have that $\mathrm{e}^{\beta \mathrm{d}} > 1$ and given that $a_h > 0$ it follows that $a_h > 1 - \mathrm{e}^{\beta \mathrm{d}} \Leftrightarrow \mathrm{e}^{\beta \mathrm{d}} - 1 + a_h > 0$.

Furthermore, $\mathrm{e}^{-\beta\mathrm{d}} > 0$ and we have that $\mathrm{e}^{-\beta\mathrm{d}}\left(\mathrm{e}^{\beta\mathrm{d}} - 1 + a_h\right) > 0 \Leftrightarrow 1 - \mathrm{e}^{-\beta\mathrm{d}} + a_h\mathrm{e}^{-\beta\mathrm{d}} > 0 \Leftrightarrow 1 - (1 - a_h)\mathrm{e}^{-\beta\mathrm{d}} > 0$.

Similarly, we have that $1 - \left(1 - a_g\right)\mathrm{e}^{-\beta\mathrm{d}} > 0$ and thus we can divide each term of Eq. (4) with $\left[1 - (1 - a_h)\mathrm{e}^{-\beta\mathrm{d}}\right]\left[1 - \left(1 - a_g\right)\mathrm{e}^{-\beta\mathrm{d}}\right]$, getting the following inequality:

$$\frac{\left(r - c_g\right)a_g \dfrac{\lambda_g}{\lambda_g + \beta}\mathrm{e}^{-\beta\mathrm{d}}}{1 - \left(1 - a_g\right)\mathrm{e}^{-\beta\mathrm{d}}} \geq \frac{(r - c_h)a_h \dfrac{\lambda_h}{\lambda_h + \beta}\mathrm{e}^{-\beta\mathrm{d}}}{1 - (1 - a_h)\mathrm{e}^{-\beta\mathrm{d}}} \Leftrightarrow W_g \geq W_h$$

Again, the expected transaction revenues should cover the expected costs and thus $W_\mathrm{g} \geq W_\mathrm{h} \geq 0$. ■

2.2 Evaluation

In this section we will compare the cost-effectiveness of the optimal strategies to other simpler strategies, using simulations. The rest strategies are: 'Least Cost Component', 'Highest Mean Availability Component', 'Least Mean Response Time Component' and 'Random Selection'.

We created a discrete-time simulator where one transaction is processed each time and the initial system (composed of a single component) has a certain probability of failing/becoming unavailable. Each experiment consisted of 10 candidate components, 10,000 transactions for each strategy, with fixed revenue (150 monetary units), and integration delay d (100 time units). Furthermore, each experiment is performed for 10 possible user-acceptable durations, from 100,000 down to 100 time units.

For each transaction, costs and mean response times are drawn from a uniform distribution and assigned to each candidate component (in [1, revenue] and [200, 2000] respectively). Furthermore, the mean availabilities for both initial and candidate components are selected from a uniform distribution in [0.7,1]. Finally, the user patience is selected from an exponential distribution based on the experiment's parameter.

The provider monitors periodically the system in order to be able to manage trustworthiness; each poll can be seen as a Bernoulli experiment. Whenever the system is found unavailable, a replacement takes place using one of the supported strategies. We assume that when a component is found unavailable then it remains unavailable, so in an extreme scenario the transaction fails because none of the components can handle the request. If the selected component is available then the transaction will fail if only its response time is larger than the remaining user's patience/ deadline.

In Figure 3 the provider's average profit for each strategy in case of unknown user patience is presented. As expected, the proposed strategy achieves the highest average profit, followed by the strategy 'Least Cost Component'. Furthermore, when users are patient the 'Least Mean Response Time Component' reaches significantly lower profits than the previous two, but for more critical systems this strategy scores higher than the 'Least Cost Component' one. The 'Random Selection' and 'Highest Mean Availability Component' strategies achieve consistently lower profits. Similar trends are obtained for different parameter values.

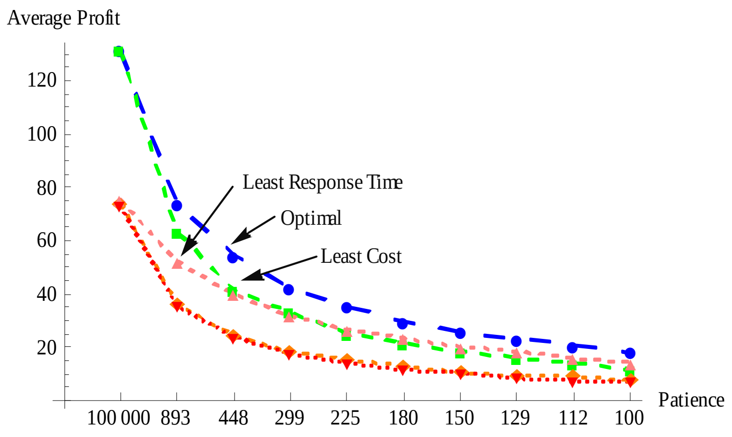

Fig. 1. The average profit for each strategy in case of unknown user patience

3 Managing system trustworthiness for a set of transactions based on estimated user's trust level

In this section we will describe an approach for allowing a provider of composite system to minimize the total expected cost of meeting the expectations of a certain customer. We will first describe the trust computational model that is used for translating the subjective user expectations into objective trustworthiness targets.

3.1 A personalized trust computational model

The trust computational model distinguishes between four different types of users, whose differences could be explained by three major techno-socio-economic factors: a) "trust stance" (e.g., "I usually trust a person until there is a reason not to"), b) "motivation to engage in trust-related seeking behaviour'"(e.g., "I look for guarantees regarding confidentiality of the information that I provide") and c) "trust-related competences" (e.g., "I'm able to understand my rights and duties as described by the terms of the application provider"). More specifically, each user is clustered to one of the four identified segments, namely "High Trust" (HT), "Highly Active Trust Seeking" (HATS), "Medium Active Trust Seeking" (MATS) and "Ambivalent" (A), which were found to have statistically significant differences. The personal attributes of users in each segment, affect not only their initial trust level, but also the way that this value is updated based on the experienced system outcomes.

We now proceed with a formal definition of the computational model. Suppose that for each system s there is a non-empty set H_s of trustworthiness factors for which any trustor i is interested in and let $w_j$be its trustworthiness value for factor j. Examples of trustworthiness metrics include availability, successful completion of transaction, etc. For each factor j the consumer has a personal opinion on how likely it is that the system will behave as expected. In other words every user has a subjective estimation of the probability that the system will be available, will store transaction results, etc. This subjective opinion is called trust level and we formulate it by means of a Beta pdf. In a mathematical formulation, the trust level of a user i for metric j after n transactions, where k were successful, is given by:

$$\tau_i^j(n) = \frac{\alpha_0 + \alpha * k}{\alpha_0 + \alpha * k + \beta_0 + \beta * (n-k)} \qquad (5)$$

More specifically, a_0 is a measure of the trustor's confidence that the system will fulfil its objectives, β_0 is a measure of the trustor's belief that the system will not meet its expectations while α, β are the two parameters that correspond to the effect of each system's success or failure respectively on trust level. In the availability metric for example, n would represent the number of attempts and k those were the system was responding (even if the output was incorrect, intercepted by an eavesdropper, etc.).

In [5], we describe and validate our approach for estimating those parameter values for each segment and consequently computing the trust levels. Notice that if $a = \beta$, then we have the classic Beta pdf whose mean value is known to converge to the average trustworthiness level after a sufficiently high number of transactions. According to our analysis this property appears only for the "HATS" segment, which means that these users have an accurate estimation of trustworthiness. On the contrary, users in "HT" segment, whose attributes result into trustworthiness overestimation, place greater importance on a success compared to a failure (meaning that $a > \beta$), while those in "MATS" and "A" segments underestimate the trustworthiness level ($a < \beta$).

3.2 The dynamic programming model

Suppose that the provider offers a single service plan to all interested buyers, which allows them to place a fixed number of n transactions for an upfront payment p_s, or unit price $p = p_s/n$. All candidate customers, being rational entities, will investigate whether they should engage with that provider, or not (the interested reader is redirected to [5] for a detailed decision criterion). If multiple providers exist in the market then obviously each customer would select the one that maximizes her expected net benefit. First-time buyers will have not experienced any system outcome before and thus their initial trust metrics ($\tau_i^j(0)$) would depend on their personality (e.g., predisposition) and any information that they can find in service description, or from their peers. For simplicity, in the following we will assume that a single binary trust metric j is important for the system only and thus the overall trust at any time is given by $\tau_i = \tau_i^j$.

Let us assume that a certain customer i has found this service plan to be beneficial. After making the upfront payment, the customer answers a questionnaire that helps the provider to identify the trustor's segment. This would allow the provider to use the trust computational model to compute the initial trust metrics. This value could be considered as a safe, minimum target for the overall trust (or respective trust metric in the general case) after n transactions in order for the trustor to renew the business relationship. The rationale is that *ceteris paribus* the user's decision would be positive if its trust level after n transactions will not have decreased.

The next step would be to compute k, the minimum number of successes necessary for reaching the initial trust level. Again, the complexity of this step can be significantly reduced by relying on the mechanics of the trust computational model. More specifically the minimum number of successes can be computed by solving the following equation for k:

$$\tau_i(0) = \frac{\alpha_0+\alpha*\mathrm{K}}{\alpha_0+\alpha*\mathrm{K}+\beta_0+\beta*(n-\mathrm{K})} \Leftrightarrow K = \frac{\tau(\alpha_0+\beta_0+\beta n)}{\alpha-\tau(\alpha+\beta)} \tag{6}$$

The last step is to create a contingency plan for reaching the initial trust level in the most cost effective way, or abandon serving the customer as early as possible. This contingency plan would suggest to the provider the optimal level of system trustworthiness at any possible situation. A situation is characterized by the tuple (number of successful transactions still necessary, number of transactions remaining). Obviously, such a contingency plan requires that the provider is able to make the necessary changes to system trustworthiness between two consecutive transactions. For example, in case of a composite system the provider could replace a component with another one and obtain the target trustworthiness value. In case of a monolithic system the trustworthiness could be affected by a different configuration. We should note that usually different system compositions, or configurations, entail a change in provider's costs. Furthermore, we would expect that increasing a component's trustworthiness is costly for its developer, e.g., a component's cost in the market equilibrium is an increasing function of trustworthiness.

Thus, the provider has received p_s monetary units in advance and knows the conditions for securing that revenue stream in the future. Suppose that the provider can query an online application marketplace and find information about the trustworthiness t_m and cost c_m of any component m that is compatible with the rest system. At the beginning or at state (k,n) she has 2 main options:

1. Serve the customer and hope that will be trustworthy enough for getting an extra amount p_s for the next set of transactions.
2. Keep the money and do nothing.

Depending on the trustworthiness level $\widehat{t_s(k,n)}$ of the system s chosen at state (k,n) there are two cases:

- With probability $\widehat{t_s(k,n)}$ we go to state $(k-1,n-1)$
- With probability $1-\widehat{t_s(k,n)}$ we go to state $(k,n-1)$

The same options are valid at any later state apart from the following situations:

- (k^-,n^+) where $k^- \leq 0$, $n^+ > 0$ and the provider has no incentive to keep placing effort, since effort is costly and would not further increase its future revenues.

- $(k^+, 0)$ where $k^+ \geq 0$ and the provider will have exhausted the number of attempts before satisfying the customer.

Thus, the run-time provider's problem can be phrased as "what is the most cost-effective trustworthiness level for the next transaction given the total number of transactions remaining and the minimum number of successes required to meet the customer's expectations?".

Such a problem can be solved by employing a finite-stage dynamic programming model, like the one described in [4]. This contingency plan can be produced proactively and be used by the provider to take any corrective actions deemed necessary at run-time. Note that the contingency plan suggests a trustworthiness level for the overall system. In the case of a composite service for example, the provider would have to replace a subcomponent with another one (or add a new) so that that the overall system meets the new security level. This is not a trivial task, but the provider could rely on tools that allow estimating the end-to-end trustworthiness of a particular system composition, like the [6].

Similar to [4], let $V_n(k)$ be the minimal expected cost incurred when the provider is at state (k, n), which refers to the path on the tree shown in Figure 2 below with the minimum total remaining expected cost. Then the provider's maximum expected profit π^* is given by $\pi^* = 2p_s - V_n(k)$. The first term represents the maximum revenues that the provider can receive in this 2-period setting, while the latter includes both the operating costs, as well as, any missed opportunities.

Furthermore, assume that:

- $V_0(x) = p_s$, where $x > 0$, which means that the provider misses the opportunity to renew the contract with the customer, and
- $V_y(x) = 0$, where $x \leq 0$ and $y \geq 0$, which means that there is no "penalty" when the minimum number of successful transactions is met.

Then, the Bellman optimality equation for this problem can be written as

$$V_n(k) = \min_{p_s \geq c \geq 0} \left\{ c + t_s\widehat{(k,n)} V_{n-1}(k-1) + \left(1 - t_s\widehat{(k,n)}\right) V_{n-1}(k) \right\}$$

where $c = \sum_m c_m$ is the total cost and we would expect that the provider considers system compositions/configurations whose total cost does not exceed the retail price.

This dynamic programming model is equivalent to the one studied in [4]; the only difference being that there is no SLA between the two parties and thus the penalty refers to the missed opportunity for receiving another upfront payment. Since the upfront payment $\mathrm{p_s}$ is fixed, the condition $\mathrm{V_0(x+2) - V_0(x+1) \geq V_0(x+1) - V_0(x)}$ is still satisfied and the optimal policy that was found is still valid. Thus, in general, the contingency plan would instruct the provider to do the following:

1. increase the TW the closer we get to contract's end and the minimum number of successful transactions was not reached. Furthermore, the higher the number of pending successful transactions the higher the increase of TW would be.
2. decrease the TW as the number of pending transactions to reach a certain number of successful transactions increases. Furthermore, the higher the number of pending transactions the higher the decrease of TW would be.

Note that this contingency plan could be used for finding the optimal design-time trustworthiness $t_s\widehat{(k,n)}$ for that particular user, or segment in general. Furthermore, the provider would have to prepare one contingency plan for every trustworthiness metric $j \in H_s$. The computational complexity of producing such a contingency plan is $O(\frac{n(n+1)}{2}|m|)$, where n is the number of transactions and $|m|$ is the number of candidate components. In order to see this remember that the dynamic programming problems inherently support recursion and thus the total number of states is given by a finite arithmetic series, $1+2+3+\cdots+(n+1)$ (as shown in Figure 2). Furthermore, in each state we have to compute $|m|$ expected costs in order to find the minimal (assuming that $m=1$ refers to a dummy component representing the "do nothing" strategy).

3.3 An example

Suppose for simplicity that the provider can manage system trustworthiness (security) by replacing one component with another from the marketplace, while the rest components are proprietary. The following table presents the cost c_m per transaction and the trustworthiness t_m of each candidate component (again we focus on a single trustworthiness metric), where $m=1$ refers to a dummy component representing the "do nothing" strategy.

Table 1. Candidate components

Component m	1	2	3	4	5
Cost c_m	0	0.3	0.4	0.8	1.9
Trustworthiness t_m	0	0.54	0.6	0.8	0.85

Furthermore let us assume that the offered service plan covers 4 transactions for an upfront payment of 3 monetary units; thus $n=4$ and $p_s=3$. If the customer, upon registration, had answered a questionnaire for revealing the trustor segment and it was found that she belongs to the "High Trust" segment, then her initial trust level would be $\tau_i(0)=\frac{\alpha_0}{\alpha_0+\beta_0}=\frac{2.2144}{2.2144+0.7106}=0.7571$. Thus, in order for her trust level after 3 transactions to be >=0.7571 the provider would have to succeed in at least $\lceil k\rceil = \lceil 2.35424\rceil = 3$ transactions, where k is given by equation (6) or more specifically:

$$\frac{2.2144+0.9583*k}{2.2144+0.9583*k+\ 0.7106+0.4399*(4-k)}=0.7571 \Leftrightarrow k=2.35424$$

Figure 2 presents the contingency plan that would be produced in this example. Solid lines represent the trustworthiness of the optimal component that should be selected, while the dashed line gives the transition probability to a state where the remaining number of successful transactions remains the same. Rectangles denote a final state and any missed revenues. Note that during the first two transactions the provider should employ $m=2$. Furthermore, at state (1,1) the provider would maximize its expected profits by using $m=4$ that has increased trustworthiness and cost.

Finally, note that whenever the provider realizes that, either the minimum number of successful transactions cannot be met, or it has already been achieved, then the optimal component is $m = 1$ (doing so reduces costs without affecting future revenues).

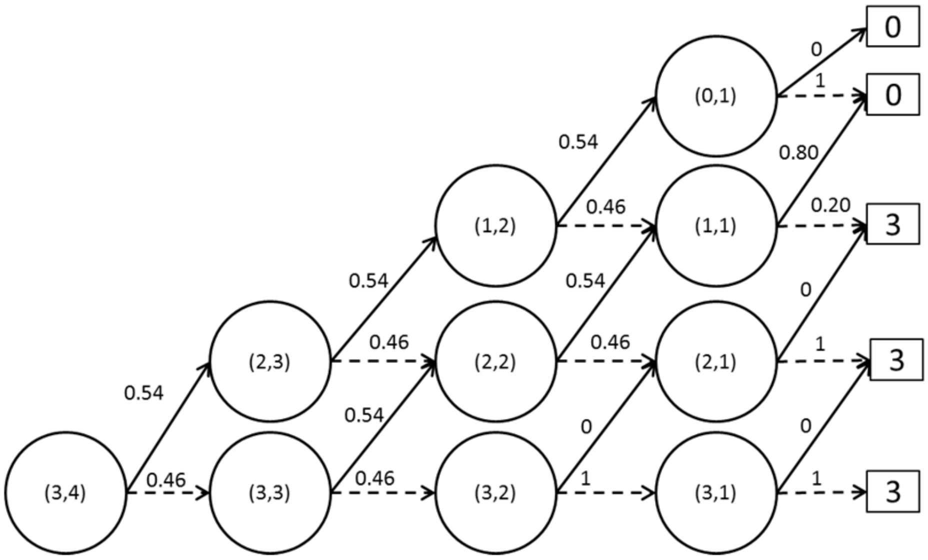

Fig. 2. An example of the contingency plan

The maximum expected profit is $\pi^* = 2 * 3 - V_4(3) = 6 - 2.72028 = 3.277972$. To see why this problem is not trivial, let us examine the following two extreme cases. The first option of the provider would be to place no effort at all. In that case the provider's expected profit during the two phases would be $\dot{\pi} = 2 * 3 - V_0(3) = 6 - 3 = 3$ (the upfront payment of the first phase, only). The other strategy would be to employ the most trustworthy component so that the customer will have observed the maximum number of successes and the probability of renewing the contract is maximized. However, c_5 is so high that the total expected cost is higher than the missed opportunities from further revenues; more specifically $\ddot{\pi} = 2 * 3 - V_0(-1) = 6 - 3.99009 = 2.00991$.

4 Conclusions and Future Work

In this paper we have provided strategies of trustworthiness management that, under certain assumptions, maximise the provider's expected profits in presence of uncertainty regarding customers' trustworthiness expectations. While in the first model we have assumed that the user's patience is exponentially distributed, in the second model we have relied on knowledge about the effect of trustors' properties on the trust dynamics. In the latter case we help service providers in producing an optimal contingency plan proactively and which allows them to keep the users' trust level high enough so that their profits are maximized.

In the future, we plan to extend the dynamic programming model in order to support contingency plans for systems that are not segment-specific. In this way, a provider will be able to manage a single system for all of its customers.

Acknowledgements

This work was supported by the EU-funded project OPTET (grant no. 317631).

References

1. Mohammadi, Nazila Gol, et al. "Trustworthiness Attributes and Metrics for Engineering Trusted Internet-Based Software Systems." *Cloud Computing and Services Science*. Springer International Publishing, 2014. 19-35.
2. Di Cerbo, Francesco, et al. "Towards Trustworthiness Assurance in the Cloud." *Cyber Security and Privacy*. Springer Berlin Heidelberg, 2013. 3-15.
3. Elshaafi, Hisain, et al. "Trustworthiness monitoring and prediction of composite services." *Computers and Communications (ISCC), 2012 IEEE Symposium on*. IEEE, 2012.
4. Derman, Cyrus, et al. "Optimal system allocations with penalty costs." *Management Science* 23.4 (1976): 399-403.
5. Kanakakis, Michalis, et al. "Towards Market Efficiency Using a Personalised Trust Computational Model. Telecommunications Policy" UNDER REVIEW
6. Mohammadi, Nazila Gol, et al. "Trustworthiness Evaluation of Socio-Technical-Systems using Computational and Risk Assessment Approaches" CAiSE Forum 2015, TO APPEAR