VU VRIJE UNIVERSITEIT AMSTERDAM

VU Research Portal

Optimal Quality of Service Control in Communication Systems

Bosman, J.W.

2014

document version
Publisher's PDF, also known as Version of record

Link to publication in VU Research Portal

citation for published version (APA)
Bosman, J. W. (2014). *Optimal Quality of Service Control in Communication Systems*. [PhD-Thesis - Research and graduation internal, Vrije Universiteit Amsterdam].

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:
vuresearchportal.ub@vu.nl

Download date: 04. Aug. 2024

Autonomous Runtime QoS Control for Composite Services in SOA

7

In this chapter, we propose a runtime closed loop control mechanism that dynamically optimizes service composition in real time by learning and adapting to changes in third party service response time behaviors.

Negotiating multiple SLAs in itself is not sufficient to guarantee end-to-end QoS levels as SLAs in practice often give probabilistic QoS guarantees and SLA violations can still occur. Moreover probabilistic QoS guarantees do not necessarily capture time-dependent behavior, (e.g., short term service degradations). Therefore, the negotiation of SLAs needs to be supplemented with *run-time QoS-control* capabilities that give providers of composite services the capability to properly respond to short-term QoS degradations (real-time composite service adaptation). Motivated by this we developed an approach that captures (temporarily) QoS degradations by tracking the behavior of third party services.

In our approach response-time realizations are used for learning an updating the response-time distributions. The currently known response-time distribution is compared against the response-time distribution that was used for the last policy update. Using well known statistical tests we are able to identify if an significant change occurred and the policy has to be recalculated. Our approach is based on fully dynamic, run-time service selection and composition, taking into account the response-time commitments from service providers and information from response-time realizations. The main goal of this run-time service selection and composition is benefit maximization for the composite service provider and ability to adapt to changes in response-time behavior of third party services. To demonstrate the usefulness of the mechanism, we have implemented our approach in a simulation environment. Moreover, we evaluate the influence of the control parameter settings on the effectiveness of our control mechanism.

7.1 Background

By tracking response times the actual response-time behavior can be captured in empirical distributions. In [122] we apply dynamic programming (DP) and we derive a service-selection policy based on response-time realizations. With this approach we assume that the response-time distributions are known or derived from histori-

[7] This chapter is based on [21], [120] and [122].

cal data. This results in a lookup-table which determines what third party alternative should be used based on actual response-time realizations.

We extend this work such that we can learn an exploit response-time distributions on the fly. Reinforcement-learning techniques are known to be able to do this but our model has a special structure that complicates the use of the classical TD learning approaches. The solution of our DP formulation searches the stochastic shortest path in a stochastic activity network [34]. Typically RL techniques solve complex learning and optimization problems by using a simulator. This involves a Q value that assigns utility to state-action combinations. Most algorithms run off-line as a simulator is used for optimization. RL has also been widely used in on-line applications. In such applications, information becomes available gradually with time. Most RL approaches are based on environments that do not vary over time. We refer to [53] for a good survey on reinforcement learning techniques.

The dynamic program (DP) in our solution has a special structure, such that the solution of a smaller sub-problem can be used to solve our problem at a more complex level. This DP can be characterized as a hierarchical DP [53, 8]. Therefore classical RL is not suitable and hierarchical RL has to be applied [8]. Also changes in response-time behavior are likely to occur which complicates the problem even more. Both the problem structure and volatility are challenging areas of research in RL.

In our approach we tackle both the hierarchical structure, and time varying behavior challenges. To this end we are using empirical distributions and updating the lookup table if significant changes occur. As we are considering a sequence of tasks, the number of possible response time realizations combinations explodes. By discretizing the empirical distribution over fixed intervals we overcome this issue. Furthermore this enables the approach where the empirical distribution is updated using a smoothing approach. An advantage of this is that no bookkeeping of previous response-time values is necessary as all samples are already represented in the smoothed distribution.

The remainder of this chapter is as follows. We start in section 7.3 with the introduction of our closed loop control learning approach. Next in Section 7.2 we introduce the workflow model that needs to be optimized. In Section 7.4 we introduce the tools for handling empirical distributions and change point detection. Using the Empirical distributions we define comprising a dynamic program that optimizes expected benefit. Experiments on our approach are performed in Section 7.5. Results are presented and discussed in Section 7.6. Finally we conclude in Section 7.7.

7.2 Model

We consider a composite service that comprises a sequential workflow consisting of N tasks identified by $T_1, \ldots, T_N$. The tasks are executed one-by-one in the sense that each consecutive task has to wait for the previous task to finish. Our solution is applicable to any workflow that could be aggregated and mapped into a sequential one. Basic rules for aggregation of non-sequential workflows into sequential workflows have been illustrated in, e.g. [122, 119, 34]. However, the aggregation leads to coarser control, since decisions could not be taken for a single service within the aggregated workflow, but rather for the aggregated workflow patterns themselves.

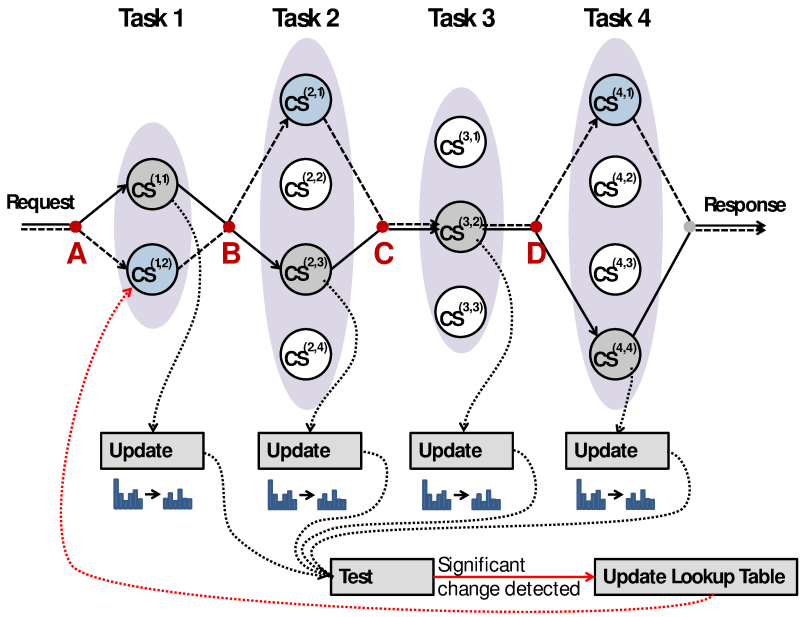


Figure 7.1: *Orchestrated composite web service depicted by a sequential workflow. Dynamic run-time service composition is based on a lookup-table. Decisions are taken at points A-D. For every used concrete service (CS) the response-time distribution is updated with the new realization. In this example a significant change is detected. As a result for the next request concrete service 2 is selected at Task 1.*

The workflow is based on an unambiguous functionality description of a service (``abstract service''), and several functionally identical alternatives (``concrete services'') may exist that match such a description [93]. Each task has an abstract

service description or interface which can be implemented by external service providers.

The workflow in Figure 7.1 consists of four tasks, and each task maps to a number of concrete services (alternatives), which are deployed by (independent) third-party service providers. For each task T_i there are M_i concrete service providers $CS^{(i,1)}, \ldots, CS^{(i,M_i)}$ available that implement the functionality corresponding to task T_i. For each request processed by $CS^{(i,j)}$ cost $c^{(i,j)}$ has to be paid. Furthermore there is an end-to-end response-time deadline δ_p. If a request is processed within δ_p a reward of R is received. However, for all requests that are not processed within δ_p a penalty V had to be paid. After the execution of a single task within the workflow, the orchestrator decides on the next concrete service to be executed, and composite service provider pays to the third party provider per single invocation. The decision points for given tasks are illustrated at Figure 7.1 by A, B, C and D. The decision taken is based on (1) execution costs, and (2) the remaining time to meet the end-to-end deadline. The response time of each concrete service provider $CS^{(i,j)}$ is represented by the random variable $D^{(i,j)}$. After each decision the observed response time is used for updating the response time distribution information of the selected service. Upon each lookup-table update the corresponding distribution information is stored as reference distribution. After each response the reference distribution is compared against the current up-to date response time distribution information.

7.3 Closed loop control

In this section we explain our closed loop approach. The main goal of this approach is benefit maximization for the composite service provider, and ability to adapt to changes in response-time behavior of third party services. We realize this by monitoring/tracking the observed response-time realizations. In the introduction we explained that the response-time based selection comprises a lookup-table that is calculated using DP (see Section 7.4.1). The DP needs information about response-time distributions and costs in order to determine lookup-table. This lookup-table corresponds to a strategy that optimizes expected benefit. In our approach, observed response-time realizations are used for learning an updating empirical response-time distributions. The currently known response-time distribution is compared against the response-time distribution that was used for the last policy update. Using well known statistical tests we are able to identify if an significant change occurred and the policy has to be recalculated. Our approach is based on fully dynamic, run-time service selection and composition, taking into account the response-time commitments from service providers and information from response-time realizations. We illustrate our approach using Figure 7.2.

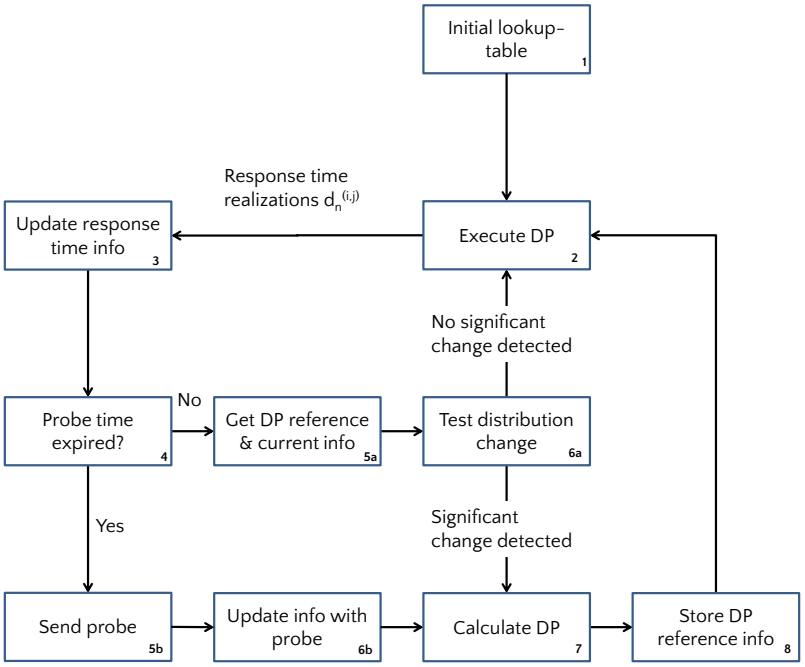


Figure 7.2: *Closed loop control approach.*

The execution starts with an initial lookup table at step (1). This could be derived from initial measurements on the system. After each execution of a request in step (2) the empirical distribution is updated at step (3). A DP based lookup-table could leave out unattractive concrete service providers. In that case we do not receive any information about these providers. These could become attractive if the response-time behavior changes. Therefore in step (4), if a provider is not visited for a certain time, a probe request will be sent at step (5b) and the corresponding empirical distribution will be updated at step (6a). After each calculation of the lookup-table, the current set of empirical distributions will be stored. These are the empirical distributions that were used in the lookup-table calculation and form a reference response-time distribution. Calculating the lookup-table for every new sample is expensive and undesired. Therefore we propose a strategy where the lookup-table will be updated if a significant change in one of the services is detected. For this purpose the reference distribution is used for detection of response-time distribution changes. In step (5a) and step (6a) the reference distribution and current distribution are retrieved and a statistical test is applied for detecting change in the response-time distribution. If no change is detected then the lookup-table remains unchanged. Otherwise the lookup-table is updated using the DP. After a probe update in step (5b) and step

(6b) we immediately proceed to updating the lookup-table as probes are sent less frequently. In step (7) and step (8) the lookup-table is updated with the current empirical distributions and these distributions are stored as new reference distribution. By using empirical distributions we are directly able to learn and adapt to (temporarily) changes in behavior of third party services.

Using a lookup-table based on empirical distributions could result in that certain alternatives are never invoked. When other alternatives break down this alternative could become attractive. In order to deal with this issue we use probes. A probe is a dummy request that will provide new information about the response time for that alternative. As we only receive updates from alternatives which are selected by the dynamic program, we have to keep track of how long ago a certain alternative has been used. For this purpose to each concrete service provider a probe timer $U^{(i,j)}$ is assigned with corresponding probe time-out $t_p^{(i,j)}$. If a provider is not visited in $t_p^{(i,j)}$ requests ($U^{(i,j)} > t_p^{(i,j)}$) then the probe timer has expired and a probe will be collected incurring probe cost $c_p^{(k,j)}$. If for example, in Figure 7.1, the second alternative of the third task has not been used in the last ten requests, the probe timer for alternative two has value $U^{(3,2)} = 10$. After a probe we immediately update the corresponding distribution. No test is applied here as probes are collected less frequent compared to processed requests. Probe cost introduces a typical trade-off between benefit due to up-to date information and cost on acquiring information. In Section 7.5 we explore the probe frequency-cost trade-off.

7.4 Algorithms

In this section we elaborate on the algorithms and detection mechanisms that are used in the closed loop control approach. These include dynamic programming (DP) in Section 7.4.1, empirical distribution discretization in Section 7.4.2, sliding window smoothing in Section 7.4.3 and exponential smoothing in Section 7.4.4. In Sections 7.4.3, and 7.4.4 we also cover change point detection using the Kolmogorov-Smirnov statistical test.

7.4.1 Determining the lookup-table

We now modify the dynamic program in Section 6.4 that calculates the optimal strategy given the current empirical distributions. Therefore we need distretized empirical distributions. Let h be the discretization step size. Let T^* be the end to end deadline: $T^* = \left\lceil \frac{\delta_p}{h} \right\rceil$. Furthermore we define $q_k^{(i,j)}$ as the discretized empirical distribution of concrete service alternative j at task j at $t = hk$. Using the discretization

approach of [34] we discretized the empirical distributions as follows for $i = 1, \dots, N$, $j = 1, \dots, M_i$, $k = 0, \dots, T^*$:

$$q_k^{(i,j)} = \begin{cases} \sum_{t=1}^{W} \mathbb{1}\left\{h[k-0.5] < d_{n-t}^{(i,j)} \leq h[k+0.5]\right\} & \text{if } k < T^*, \\ \sum_{t=1}^{W} \mathbb{1}\left\{d_{n-t}^{(i,j)} > h[k-0.5]\right\} & \text{otherwise.} \end{cases} \tag{7.1}$$

Here is $d_t^{(i,j)}$ the t-th response-time realization for service alternative j at task i, and $\mathbb{1}_{\{A\}}$ is the indicator function over A which is 1 if A is true and 0 otherwise. More details about determining and using the discretized empirical distribution can be found in Section 7.4.2. Using the discretized empirical distributions backward recursion formulae can be formulated. We start with the terminal reward function for $b = 0, \dots, T^*$:

$$P_b^{(N+1,*)} = \begin{cases} R & \text{if } b > 0, \\ -V & \text{otherwise.} \end{cases} \tag{7.2a}$$

Using this function we iterate backwards using the following equations for $i = 1, \dots, N$, $j = 1, \dots, M_i$, $b = 0, \dots, T^*$:

$$P_b^{(i,*)} = \max_j \left\{ -c^{(i,j)} + R_b^{(i,j)} + V_b^{(i,j)} \right\}, \tag{7.2b}$$

$$R_b^{(i,j)} = \sum_{k=0}^{b} q_k^{(i,j)} P_{k-b}^{(i+1,*)} \quad \text{and} \tag{7.2c}$$

$$V_b^{(i,j)} = \sum_{k=b+1}^{T^*} q_k^{(i,j)} P_0^{(i+1,*)}. \tag{7.2d}$$

Here, the term $P_b^{(i,*)}$ represents the expected benefit per request given time budget b at task i under the optimal dynamic programming decision strategy. The term $R_b^{(i,j)}$ represents the expected reward, when concrete service j (assigned to task i) is executed for the given time budget value b. Finally the term $V_b^{(i,j)}$ represents the expected penalty for exceeding the overall deadline at task i while executing concrete service j for the given time budget value b. The expected reward and penalty functions take into account the impact of future decisions as represented by terms relating to $P_b^{(i+1,*)}$ in (7.2c) and (7.2d).

While applying formulae (7.2b)-(7.2d), the corresponding decisions (actions) $A_k^{(*,i)}$ can be obtained by storing the maximum arguments for $i = 1, \dots, N, \quad j = 1, \dots, M_i, \quad k = 0, \dots, T^*$:

$$A_k^{(i,*)} = \underset{j}{\operatorname{argmax}} \Big\{ -c^{(i,j)} + R_k^{(i,j)} + V_k^{(i,j)} \Big\}.$$

7.4.2 Empirical distribution

In 7.4.1 we introduced an approach for discretizing empirical distributions such that these become suitable for using in DP. This is derived from the discretization approach in [34] for $i = 1, \dots, N, \quad j = 1, \dots, M_i, \quad t = 0, \dots, T^*$:

$$\begin{aligned} q_k^{(i,j)} &= \begin{cases} \mathbb{P}\big(\hat{D}^{(i,j)} \le h[k+0.5]\big) - \mathbb{P}\big(\hat{D}^{(i,j)} \le h[k-0.5]\big), & k < T^*, \\ \mathbb{P}\big(\hat{D}^{(i,j)} > h[k-0.5]\big), & k = T^*, \end{cases} \\ &= \begin{cases} \sum_{t=1}^{W} \mathbb{1}\big\{h[k-0.5] < d_{n-t}^{(i,j)} \le h[k+0.5]\big\}, & k < T^*, \\ \sum_{t=1}^{W} \mathbb{1}\big\{h[k-0.5] < d_{n-t}^{(i,j)}\big\}, & k = T^*. \end{cases} \end{aligned} \tag{7.3}$$

In Equation (7.3), $\hat{D}^{(i,j)}$ is the empirical response-time process for concrete alternative j at task i, and
$d_n^{(i,j)}$ is the response time of the nth sample for concrete alternative j at task i.

Consider the discretized distribution $q_k^{(i,j)}$. Actually $q_k^{(i,j)}$ is a histogram where kth bin is bounded by
$\big[\, h(k-0.5); h(k+0.5) \,\big)$ and where the sum of the frequencies is normalized to 1. There is a tradeoff in choosing the bin size. When h has a small value the histogram will consist of many bins. In the case that a few large bins are used, too much information about sample location is lost. A usual method for constructing an empirical distribution histogram $\tilde{q}^{(i,j)_n}$ from samples is as follows: Let n be the number of samples that are already included in the histogram. Let $d_{n+1}^{(i,j)}$ be a new sample from the distribution we want to estimate. Then the histogram is updated as follows:

$$\tilde{q}_{n+1}^{(i,j)} = \frac{n}{n+1} \left[\tilde{q}_n^{(i,j)} + \mathbb{1}_{\big\{(k-0.5)h \le d_{n+1}^{(i,j)} < (k+0.5)h\big\}} \right], \tag{7.4}$$
$k = 0 \dots, K.$

note that this corresponds to the definition of the empirical distribution defined in (7.3). Define:

$$\hat{F}_n(t) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{[x_i \leq t]\}} \tag{7.5}$$

the empirical distribution over t based on samples x_i. The central limit theorem states that *pointwise*, $\hat{F}_n(t)$ has asymptotically a normal distribution (see [102, p265] for more details). Essentially $\mathbb{1}_{\{x_i \leq t\}}$ has a Bernoulli distribution with success probability $F(x_i \leq t)$.

$$\sqrt{n}\big(\hat{F}_n(t) - F(t)\big) \xrightarrow{d} \mathcal{N}\Big(0, F(t)\big(1 - F(t)\big)\Big). \tag{7.6}$$

Using the empirical distribution over all samples is a suitable approach when the distribution does not change over time. However, note that new samples will have a decreasing impact on the empirical distribution. If the distribution changes over time this is an undesired property.

7.4.3 Empirical distribution based on sliding window

A natural approach for tracking changes is that we define a sliding window of W samples. Let $\mathcal{X}_n = \{d_{n-n+1}, d_{n-W+2}, \dots, x_n\}$ be the current set of samples in the sliding window after inserting the nth sample d_n. These sets are used to determine the empirical distribution that serves as an input for the dynamic program. A disadvantage of this method is that the samples that are included in the sliding window need to be stored for the sliding window. The empirical distribution of the nth sample in the sliding window approach is defined by:

$$\hat{F}_n(t) = \frac{1}{W}\sum_{i=0}^{W-1} \mathbb{1}_{\{d_{n-i} \leq t\}}. \tag{7.7}$$

If we take a $\mathcal{X}_n$ and $\mathcal{X}_m$, $m > n$ we could apply statistical tests (e.g. the two sample Kolmogorov Smirnov test) in order to determine if a significant change occurred in the distribution. Let $D_{n,n'}$ be the Kolmogorov–Smirnov (KS) statistic on two empirical distributions with n and n' observations respectively. Let $\hat{F}(x)$ and $\hat{F}'(x)$ be the corresponding empirical distribution functions. Then the KS statistic is defined as:

$$D_{n,n'} := \sup_x |\hat{F}(x) - \hat{F}'(x)| \quad \text{and} \quad \sqrt{\frac{nn'}{n+n'}} D_{n,n'} \tag{7.8}$$

has a Kolmogorov distribution.

7.4.4 Exponentially smoothed empirical distribution

To prevent overhead of sliding window bookkeeping, we apply a smoothing approach. In this approach new samples can be included using the following weighting scheme:

$$\tilde{p}_{n+1}^{(i,j)} = \kappa \tilde{p}_n^{(i,j)} + (1-\kappa)\mathbb{1}_{\left\{(k-0.5)h \le d_{n+1}^{(i,j)} < (k+0.5)h\right\}}. \tag{7.9}$$

This is similar to the exponentially weighted moving average estimate in [32]. Here $0 < \kappa < 1$ is a smoothing factor. When κ is close to one new samples have a relatively small impact and it will take a long time before the empirical distribution converges to a new distribution. If κ is close to zero new samples have a big impact resulting in an empirical distribution that quickly follows changes in the real sample distribution but with more noise in the histogram.

Modified Kolmogorov-Smirnov test As we use smoothed empirical distributions, a statistical test on smoothed distributions is required. To this end we modified the two sample Kolmogorov-Smirnov test, suitable for comparing two smoothed empirical distribution functions $q_k^{(n)}$ and $q_k^{(m)}$. The Kolmogorov Smirnov test is based on the result that the empirical distribution as defined in (7.7) converges pointwise to a normal distribution then the number of samples goes to infinity. The original Kolmogorov Smirnov test relates this to the Brownian bridge on which the test statistic is based [102]. We want to apply a similar central limit theorem result for the smoothed process such that we can apply statistics like the Kolmororov Smirnov statistic. We try to relate κ to a virtual window size W such that the variance of any point in the smoothed empirical distribution corresponds to the variance of the original sliding-window empirical distribution consisting of W i.i.d. samples.

Let $q_k^{(n)}$, $q_k^{(m)}$ be empirical distributions, discretized according to (7.3), let X_i be i.i.d. random variables, and let Y_i be the smoothing process on the variables. For exponential smoothing (ES) the variance results from geometric terms:

$$\begin{aligned}\mathrm{Var}(Y_i) &= \kappa^2 \mathrm{Var}(Y_{i-1}) + (1-\kappa)^2 \mathrm{Var}(X_i) \\ &= (1-\kappa)^2 \sum_{k=0}^{\infty} \kappa^{2i} \mathrm{Var}(X_{i-k}) = \frac{1-\kappa}{1+\kappa} \mathrm{Var}(X_i).\end{aligned}$$

Thus for for given κ the virtual window size is $W = \frac{1+\kappa}{1-\kappa}$. Let K_α is the critical value of the Kolmogorov distribution corresponding to significance level α. We replace n and n' in Equation (7.8) with the virtual window size W:

$$\sqrt{\frac{1+\kappa}{2(1-\kappa)}} D_\kappa > K_\alpha. \tag{7.10}$$

The statistic is obtained from smoothed, discretized empirical distributions $q_k^{(n)}$, and $q_k^{(m)}$ as follows:

$$D_\kappa := \sup_k |q_k^{(n)} - q_k^{(m)}|. \tag{7.11}$$

7.5 Experimental setup

To test the closed-loop approach defined in Section 7.3 we define an experimental workflow that can be used for investigating the impact of the closed-loop control approach parameters. We emphasize that this experimental set-up is tailored for evaluating responsiveness of our closed-loop approach with respect to response-time distributions and does not limit us in tracking other systems with different response-time models.

Figure 7.3 represents a work flow consisting of four tasks: For each task four concrete

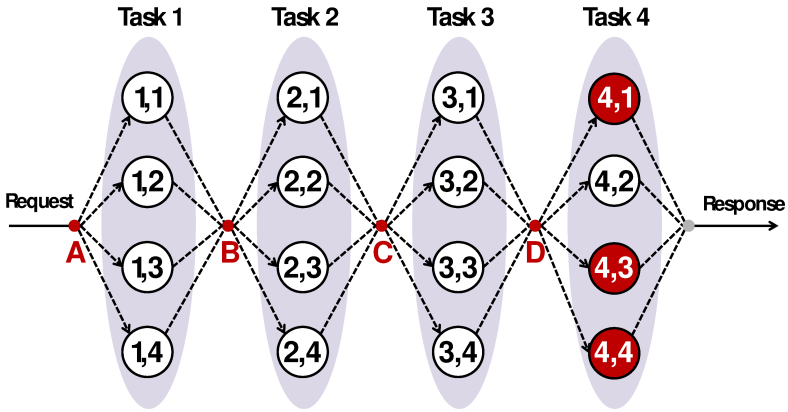


Figure 7.3: *Default experimental work flow. For task T_4 alternatives 1, 3 and 4 are slow. The optimal strategy should detect this and use alternative 2.*

service alternatives are available. At each alternative j for task T_i the response-time distribution is summarized in terms of mean $\mu^{(i,j)}$ and variance $\left(\sigma^{(i,j)}\right)^2$.

7.5.1 Experimental model

We model the burstiness of response-time behaviors using the classical Gilbert-Elliott [51, 45] Discrete Time Markov Chain (DTMC) model. Each concrete service is modeled with its own DTMC with underlying state which can be either fast or slow. For sake of readability we omit the superscript indexing for all variables in the experimental model definition ($\square^{(i,j)} \rightarrow \square$), but we emphasize that *each concrete service has its own set of parameters*. The DTMC enables us to capture the rate of change between fast and slow response-time behavior. Essentially the DTMC model results in taking a mixture of distributions represented by the fast and slow states. The DTMC is defined by the following transition matrix corresponding to state the vector $\begin{bmatrix} fast & slow \end{bmatrix}^T$ and corresponding state probabilities $\begin{bmatrix} p_{fast} & p_{slow} \end{bmatrix}^T$:

$$P = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}. \tag{7.12}$$

The transition probabilities α and β are parametrized by factors a, b, t_{cycle}, and k. In the parametrization $0 < a < 1$ is the scaling factor for the fast state mean μ_{fast}, $b > 1$ is the scaling factor for the slow state mean μ_{slow}. We define the expected cycle time $t_{cycle} > 0$ as the expected time it takes to leave the current state and return to its original state e.g. fast→slow→fast or slow→fast→slow. Furthermore the variances of the fast and slow states are related by scaling factor $k > 0$ where $k > 1$ if $\sigma^2_{fast} > \sigma^2_{slow}$ and $k < 1$ if $\sigma^2_{fast} < \sigma^2_{slow}$. Both high and low states are represented by log-normal distributions with means μ_{slow}, μ_{fast} and variances σ^2_{slow}, σ^2_{fast}. A motivation that supports this burstiness model is that random variables, holding times of various message types were found to be log-normal mixtures (conversation time in land and mobile telephone networks, voice mail message length, facsimile transmission time) [17] The log-normal distributions are related to the parametrization as follows:

$$\begin{aligned} \mu_{fast} &= a\mu, \qquad \mu_{slow} = b\mu, & 0 < a < 1, \quad 1 < b, \\ \sigma^2_{fast} &= v\sigma^2, \qquad \sigma^2_{slow} = kv\sigma^2, & v > 0, \quad k > 0. \end{aligned}$$

In order to obtain the desired mean μ and variance σ^2 we need:

$$\mu = p_{slow}\mu_{slow} + p_{fast}\mu_{fast}, \tag{7.13}$$

$$\sigma^2 + \mu^2 = p_{fast}(\sigma^2_{fast} + \mu^2_{fast}) + p_{slow}(\sigma^2_{slow} + \mu^2_{slow}). \tag{7.14}$$

From Equations (7.13) to (7.14) and the properties $\alpha p_{fast} = \beta p_{slow}$, and $\frac{1}{\alpha} + \frac{1}{\beta} = t_{cycle}$ we obtain the following parameter values in terms of a, b, t_{cycle}, and k:

$$\alpha = \frac{p_{fast}}{t_{cycle}}, \quad \beta = \frac{p_{slow}}{t_{cycle}}, \qquad p_{fast} = \frac{1-b}{a-b}, \qquad p_{slow} = 1 - p_{fast},$$
$$v = \frac{\sigma^2 + \mu^2(1 - p_{fast}a^2 - p_{slow}b^2)}{\sigma^2(p_{fast} + kp_{slow})}.$$

7.5.2 Experimental set-up

The parameters $\mu^{(i,j)}$, $\sigma^{(i,j)}$, $c^{(i,j)}$ of our experiments (Figure 7.3) are summarized in Table 7.1. We consider a run consisting of 10000 requests with a warm-up of 1000 requests. Each run is replicated 48 times. Furthermore, we consider a time horizon of 40 time units. Each successful response to a request will gain a reward of $R = 20$ money units while for a failed request $V = 50$ money units has to be paid. All concrete service response-time distributions use the same scaling factors for the low and high response-time means $a^{(i,j)} = 0.85$, $b^{(i,j)} = 2$, and the same variance factor $k^{(i,j)} = 100$ for all $i = 1, \ldots, N$, $j = 1, \ldots, M_i$. Furthermore, we keep the values of $t_{cycle}^{(i,j)}$, $W^{(i,j)}$ and $t_p^{(i,j)}$ fixed for all concrete services. Therefore, we omit index (i,j) for t_{cycle}, W, and t_p in our experiment values. We vary t_{cycle} and W_p in the range $\{10, 15, 20, 50, 100\}$ and vary t_{cycle} in the range $\{200, 500, 1000, 2000, 4000\}$. The parameter effects that we want explore are sample window size, test significance level, test type (e.g. Kolmogorov Smirnov), and probe interval. All experiments has been run until the confidence interval was less than 1%.

Table 7.1: *Concrete service alternatives for task i*

		Parameter		
		c	μ	σ
Service alternative $(\cdot, j)$	$(\cdot, \mathbf{1})$	1	5	2
	$(\cdot, \mathbf{2})$	2	3	2
	$(\cdot, \mathbf{3})$	5	2.5	2
	$(\cdot, \mathbf{4})$	10	1.25	3

7.6 Results

Using the experimental set-up in Section 7.5.2 we obtained the following performance measures:

- Expected benefit (see Section 7.6.1),
- Number of updates (see Section 7.6.2),
- Number of probes (see Section 7.6.3),
- Average probe cost per request (see Section 7.6.4).

For expected benefit we compare three approaches: (KS) the sliding window approach based on Kolmogorov Smirnov statistic for change detection, (KSS) the smoothing approach with adjusted Kolmorov Smirnov statistic for change detection, and theoretical (TH) from the policy based on the *known in advance* mixture of log-normal distributions. The TH is compared as benchmark where analysis is done on historical data from services and short term changes in response-time distributions are aggregated in to one long term response-time distribution. For all other performance measures we only compare KS and KSS as no updating or probing is involved in the TH approach as the distribution is known/analyzed in advance. However the mixture of log-normal distributions is controlled by a Markov Chain and therefore we expect the KS and KSS approach to perform better.

7.6.1 Expected benefit

Figure 7.4 presents the experimental results on average benefit per request with KS, KSS, and TH as functions of cycle time t_{cycle} (Figures 7.4a, 7.4b), probe time-out t_p (Figure 7.4c), and sliding window sample count W (Figure 7.4d). If not specified we kept $t_{cycle} = 1000$ requests, $t_p = 100$ requests, $W = 20$ requests, and $\alpha = 0.01$. Note that in Figure 7.4 probe cost is not incorporated and so all probes have cost $c_p^{(i,j)} = 0$, $i = \{1, \dots, N\}$, $j = \{1, \dots, M_i\}$. In Figure 7.4a we observe that (as expected) for short cycle time t_{cycle} KS and KSS perform close to TH. This is because for small t_{cycle} the distribution becomes effectively a mixture of log-normal distributions on which the dynamic program in the TH approach is solved. For very small t_{cycle} the TH approach performs better but this is the case where effectively no change in response-time distributions occurs. Here we are simply comparing learning versus knowing the response-time distributions. For larger t_{cycle} the difference increases as all concrete services are slowly alternating between two distribution states. Figure 7.4c illustrates that a higher probe time-out t_p will decrease expected benefit for KS and KSS. This corresponds to the fact that a higher probe time-out will cause the system to respond slower to changes in less frequently used alternatives which could have become attractive. Also a larger sliding window size W decreases expected benefit for KS and KSS as new response-time observations are smoothed out across more samples.

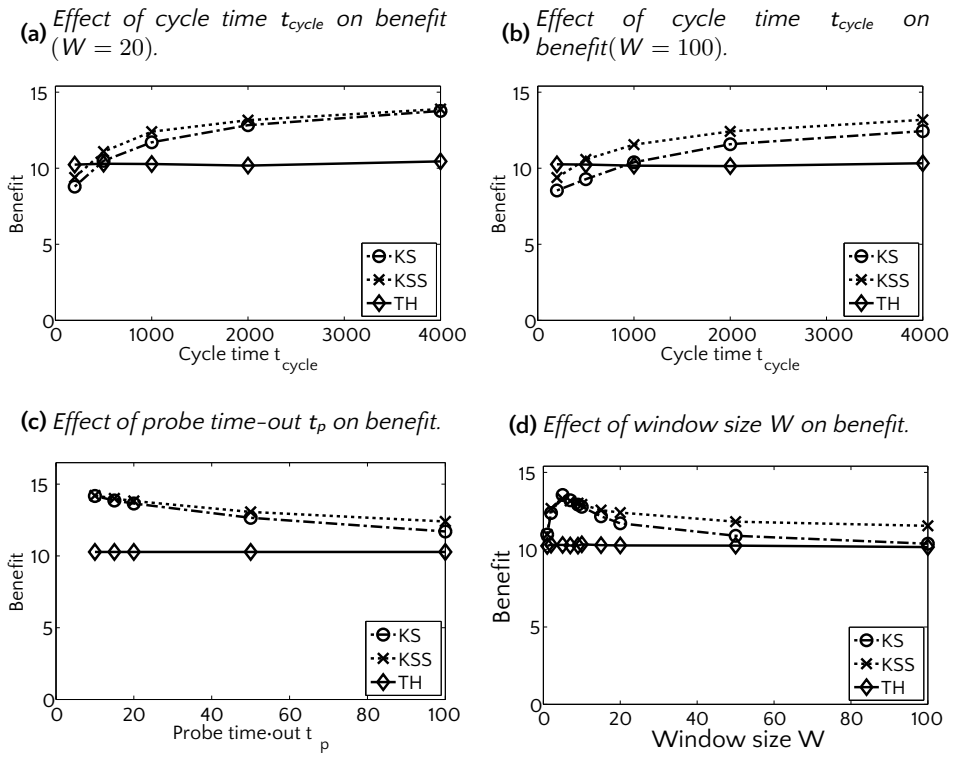


Figure 7.4: *Effect of cycle time, probe time-out and window size on expected benefit per request.*

7.6.2 Number of updates

In the experiments we also recorded the number of lookup table updates. Figure 7.5 contains the required number of lookup-table updates for KS, KSS, and TH as function of cycle time, probe time-out and window size. If not specified we kept $t_{cycle} = 1000$ requests, $t_p = 100$ requests, $W = 20$ requests, and $\alpha = 0.01$. As expected Figure 7.5a illustrates that a larger cycle time corresponds to less updates. This corresponds to the behavior of our experimental set-up that a larger cycle time results in less jumps in the distribution (Markov Chain). Therefore less significant changes should be detected. In Figure 7.5b we observe that the probe time-out has a big impact on the number of lookup table updates. As we update the lookup-table after each probe, the probe time-out should be not too low. In Figure 7.5c we observe quite constant behavior over the window sizes. For large sliding window size the number of updates slightly reduces as the empirical distributions approaches

the actual response-time distributions. However as we observed in Figure 7.4d we obtain a distribution aggregated over a longer time span and are not able to track short term changes in response times.

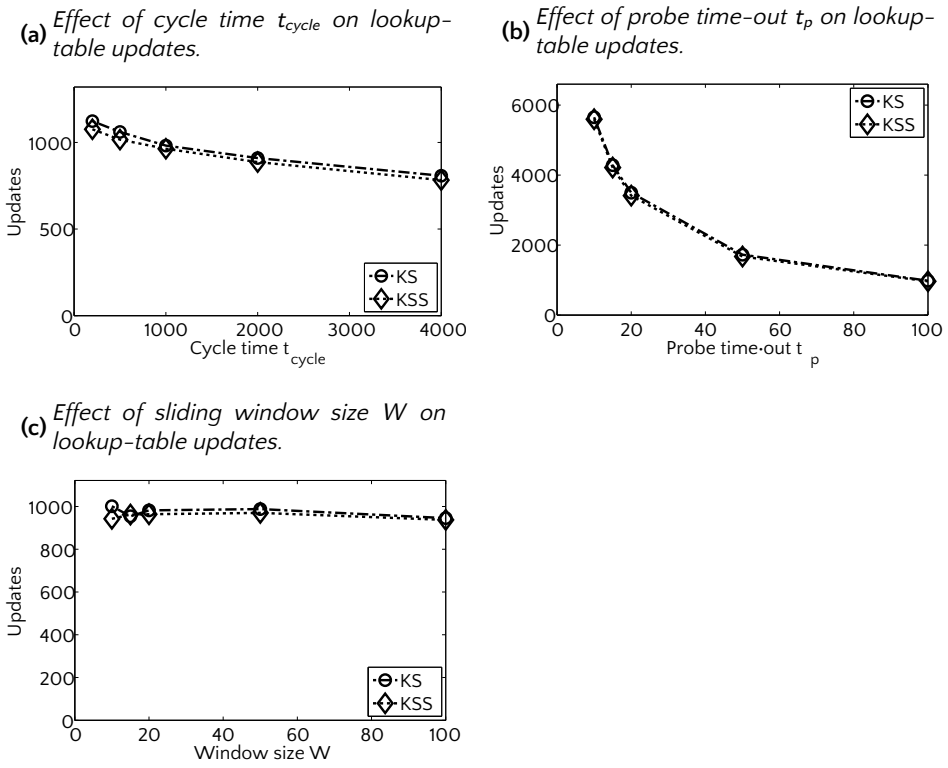


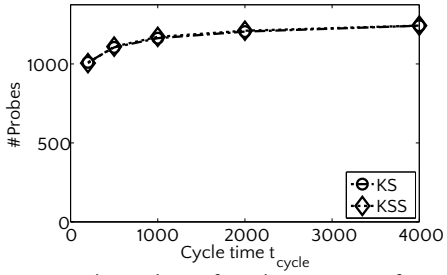
Figure 7.5: *Effect of cycle time, probe time-out and window size on the required number of lookup-table updates.*

7.6.3 Number of probes

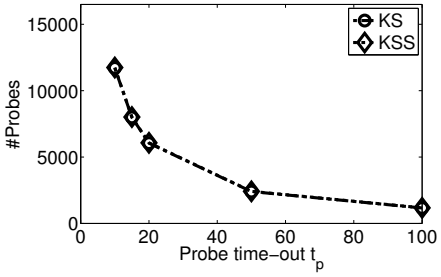
Another interesting measure is the total number of probes that is sent as result of our closed-loop approach. In Figure 7.6 we present the effect of cycle time, probe time-out and window size on the total number of probes sent, for KS, KSS, and TH. If not specified we kept $t_{cycle} = 1000$ requests, $t_p = 100$ requests, $W = 20$ requests, and $\alpha = 0.01$. We observe in Figure 7.6a the interesting behavior that the total number of probes sent increases as the cycle time increases. An explanation for this

behavior is that all concrete service alternative distributions in our experimental set-up are controlled by independent Markov Chains. Therefore, a certain combination of response-time distribution states is observed for a longer time which allows the lookup table to adapt to these specific situations. As a result services that would rarely be invoked are invoked more frequently resulting in less probes (less probe time-outs). As expected we observe in Figure 7.6b that the probe time-out has big effect on the total number of probes sent. This is a result of the probing strategy we proposed in Section 7.3. Figure 7.6c illustrates that the window size has hardly effect on the number of probes sent. A larger window size will make distribution estimates more accurate and will not cause the lookup to change dramatically. So rarely invoked concrete service alternatives are not necessarily visited more frequently as result of a change sliding window size.

(a) *Total number of probes sent as function of cycle time t_{cycle}.*



(b) *Total number of probes sent as function of probe time-out t_p.*



(c) *Total number of probes sent as function of window size W.*

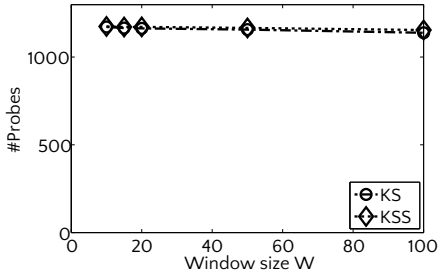


Figure 7.6: *Effect of cycle time, probe time-out and window size on the total number of probes sent.*

7.6.4 Mean probe cost

In Section 7.6.3 we discussed the results on the total number of probes sent for the different algorithms. Alternatively this can be seen as the behavior of probe cost when cost of probing for all concrete service alternatives is equal to 1. In our experiments we also individually tracked the number of probes sent of each concrete service alternative individually. The graphs in Figure 7.7 represent the expected probe cost per request (total cost/#probes) if the probe costs are proportional to the costs of invoking a concrete service alternative: $c_p^{(i,j)} \propto c_p^{(i,j)}$. The graphs represent the case where $c_p^{(i,j)} = c_p^{(i,j)}$, for $i = \{1, \ldots, N\}$, and $j = \{1, \ldots, M_i\}$. For this case we observe similar behavior of the graphs compared to Figure 7.6 in Section 7.6.3. If not specified we kept $t_{cycle} = 1000$ requests, $t_p = 100$ requests, $W = 20$ requests, and $\alpha = 0.01$.

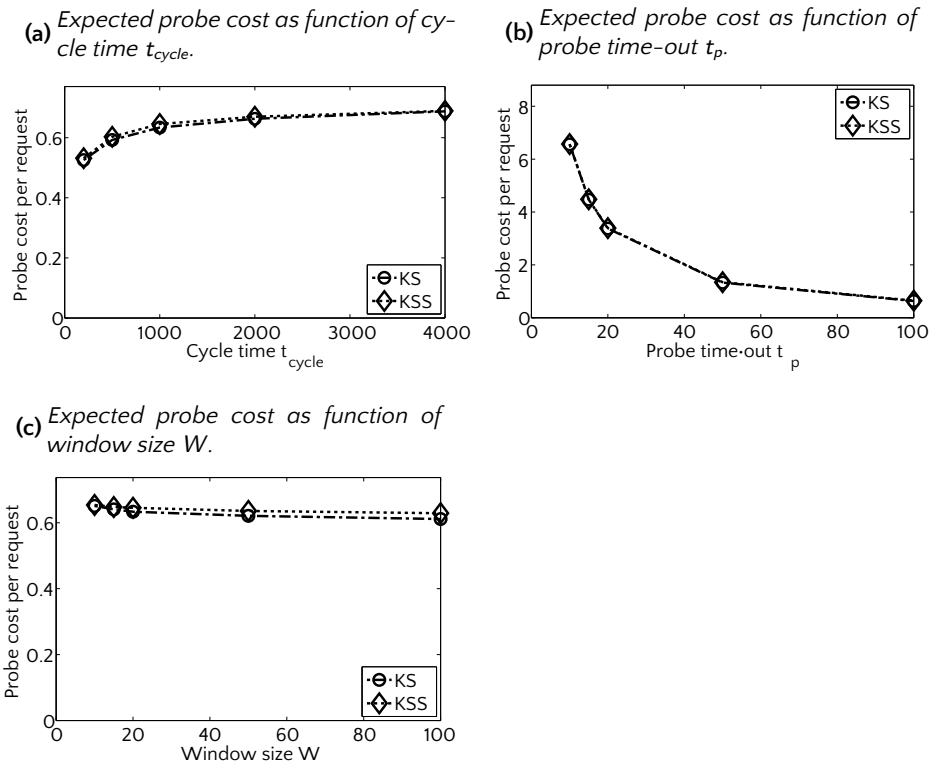


Figure 7.7: *Effect of cycle time, probe time-out and window size on the expected probe cost per request.*

7.7 Discussion

We modeled and implemented a closed-loop approach where dynamic programming is applied on empirical distributions resulting from the actual realized response-time distributions of concrete service providers. Our approach is robust to changes in the sense that it adapts to changes in response-time distributions of concrete service alternatives. To achieve this we use a smoothing approach or sliding window approach on the empirical distribution. The smoothing approach has advantage that there is no overhead in bookkeeping of sliding window samples while past response-time realizations have limited impact in the smoothed empirical distributions. When using our approach there is a trade-off between parameters that we need to optimize. These parameters are the sliding window W or exponential smoothing parameter κ and the change point detection test significance α. The constraints are here determined by computational power and probe cost. Typically we would like to update our lookup table every request and probe frequently. However it takes time to compute a new strategy. Furthermore cost is connected with probes. It would be a waste if a new probe is sent while a previous lookup table computation is still running. We should choose probe time-outs such that we can exploit information about improved service without spending too much cost on probing and using too much computational power. Experimental results indicate that in an environment with changing response-time behavior our closed-loop approach has a significant advantage compared to a static lookup table as our approach has the strong advantage that it learns and exploits response-time behavior on the fly.

Tuning window size W (or corresponding smoothing factor κ) and α creates a second layer of control where these parameters are adapted to optimal values. The update of these parameters is typically on a larger time scale that is not in the scope of our experiments. This is a interesting direction for further research.