

Using real-time dependability in adaptive service selection

Markus C. Huebscher, Julie A. McCann
Department of Computing
Imperial College London
{mch1,jamm}@doc.ic.ac.uk

Abstract

In Service-Oriented Architectures (SOA), services advertise a description of the type of service they can provide. Sometimes, services also advertise Quality of Information (QoI) attributes describing the quality of their provided service. When there are multiple alternative service providers for the same type of service, a client can use the advertised QoI to select the best alternative. However, these QoI attributes may not be reliable, leading the client to select a sub-optimal alternative.

We suggest a mechanism in SOAs for rating the current dependability of the advertised QoI of each alternative, and using this dependability to automatically select the best alternative service provider. We present results of a case study that considers weather forecast services.

1 Introduction

Services in SOAs advertise a description of the type of service they can provide. This may include measures of the quality of their provided service. For instance, a location service which can provide a user's PDA with an estimate of its location may also be able to advertise its expected error, which is dependent on the current location of the user. The expected error in a location system is an example of Quality of Information (QoI) metric that a service may advertise. Advertising ones QoI is important, as there may be multiple services offering the same type of information, e.g. location, but with varying degrees of quality.

For example, WiFi positioning, which compares the signal strength of nearby access points to a previously measured survey map, can provide up to 1.5 metres accuracy if conditions are optimal [2], whereas ultrasonic tags can have a precision of under 10 centimetres [1], but their area of coverage is likely to be smaller. Thus, clients can use advertised QoI to select the best currently available service. Note that "best" is a client-specific notion, different clients may have different requirements and therefore a different

notion of what is best. For instance, a client interested in location information may be interested solely in precision (e.g. locating storage containers in a warehouse), while another client may need to trade-off precision for high refresh rate (e.g. reacting quickly to someone entering a room).

In practice, at present it is often the case that services do not advertise their QoI. Moreover, even if they did, selecting a source based on the advertised QoI of each source means that we trust the reliability of each source's advertised QoI. However, this is often a wrong assumption to make. Indeed, service providers may be unable to provide accurate QoI, even without any malicious intent.

Consequently, we describe in this paper a means for a client to rate the reliability of a service provider, which we call *dependability*, and use this dependability to select the currently most "dependable" alternative, thus implementing an automatic service selection based on a client-side computed measure of each alternative's quality. Dependability is continuously updated as the client receives new data from the service. Thus, dependability is dynamic and can change over time: it is not always the case that the same alternative is always the most dependable.

2 Using dependability with QoI metrics at SOA middleware level

Figure 1 illustrates the idea of automatic service selection based on dependability. A service selector is placed between the multiple alternative service providers for the same type of service (e.g. location) and the clients. The service selector keeps track of each service's QoI metrics, which may change over time or, as in the case of location, may be location- and thus client-specific. Further, the selector continuously chooses for each client the currently best alternative depending on each client's notion of best, which the client relates to the selector in the form of a utility function $u(\text{QoI}, d)$ that takes as input QoI metrics and dependability and outputs a quantitative measure of an alternative's client-specific quality. Thus, the service selector can be implemented at a middleware level, where the mid-

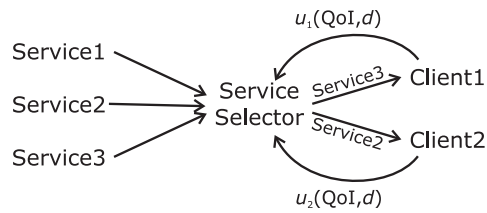


Figure 1. Adaptive service selection.

Middleware selects for each client their “best” service. As QoI and dependability may change over time, the middleware may decide to switch to another alternative service should the currently used service not be optimal anymore. Depending on the type of service, this adaptation may happen transparently at the middleware, or may require client notification and confirmation. Adaptation may also occur when a new (and better) service provider should appear (e.g. mobile user entering coverage area of location system), or when the currently used provider fails (e.g. exiting coverage area). Thus, the middleware implements service provision with self-healing (on service failure), self-configuring (on new service), and self-optimising (on change of dynamic QoI attributes and dependability) properties.

In the next section, we define the idea of *dependability* and describe how it can be used in our service selector.

3 Dependability

There are many ways of defining dependability, many of which are rather service-specific. We propose here a general definition that applies to various types of services.

The dependability of a service provider is a *measure of the probability that the data received is correctly within an advertised QoI metric*. This means that there can be many QoI-metric specific dependability values, e.g. one for precision and one for refresh rate.

In the remainder of this paper, we will focus on what is usually the most important form of dependability in context-awareness data, i.e. that of *precision*. For instance, a location system is dependable if it constantly advertises a precision of 5m and the actual precision is usually under 5m. Instead, a location system is not dependable if it advertises a precision of 0.01m, but its actual precision is usually above 0.01m.

For services with discrete values, the concept of precision is actually inappropriate. Instead, we use the term *probability of correctness*. Thus, a weather service has a high probability of correctness if the weather forecast mostly turns out to be correct.

Computing dependability sometimes requires clients of the service to determine whether the provider has been dependable in its delivered service. In such cases, each client

that has interacted with a service provider needs to inform the selector of the dependability outcome of this interaction, e.g. a binary yes/no feedback as to whether the provider was dependable. The selector can then collect the feedback from all clients of a provider to compute an estimate of this provider’s dependability. This actually shifts a trust problem: we are estimating a provider’s dependability because we do not trust its advertised QoI; however, by using client input into the dependability, we must trust their feedback. The important point here is that we are not assuming malicious parties. We assume that providers err in their advertised QoI not (necessarily) because they are trying to fool clients, but because for one reason or other they are simply unable to provide accurate QoI. As for the clients, assuming they are not malicious, it is in their best interest to provide accurate feedback (or otherwise none at all), as it contributes to improving (or degrading, in the case of bad feedback) the quality of the service ultimately received.

In the next section, we present a case study where dependability can be computed directly at the service selector, thus not involving clients. While the notion of dependability is a general concept applicable to most data-delivery services, our case study looks at the concrete example of the adaptive selection of a weather forecast service.

4 Case study: weather forecasts

4.1 Introduction

As an example of adaptive service selection based on dependability, let us consider weather forecast services. There are many web sites that provide free weather forecasts, so how do we select the most reliable one? Also, reliability is not a static attribute of a weather forecast site: one moment a particular site may be more reliable, while the next another site may be better.

Currently, these web sites do not normally advertise their probability of correctness (although they may give a probability of rain), so we will have to assume that they all have equal probability of correctness, say 100% for simplicity. Our goal is now to determine the most dependable weather source. Given that we have set each weather source’s probability of correctness to 100%, our definition of dependability given in Section 3 (applied to probability of correctness) can be translated into finding the current success rate of each weather source.

For our case study, we use as weather sources Weather.Com¹ and WeatherPerHour². These web sites have the advantage that they deliver hourly weather forecasts and therefore it is possible to collect many forecasts in a short

¹<http://www.weather.com/>

²<http://www.weatherperhour.com/>

Table 1. Extract of weather data log and adaptive selection decision.

Time of forecast/actual weather	Weather.Com		WeatherPerHour		Adaptive	
	Forecast	Actual	Forecast	Actual	Selected	Correct
26/04/05 23:15	Partly Cloudy	Fair	partly cloudy	fair	WC	✓
27/04/05 00:15	Mostly Cloudy	Fair	partly cloudy	fair	WC	×
27/04/05 01:15	Mostly Cloudy	Fair	partly cloudy	fair	WPH	✓
27/04/05 02:15	Mostly Cloudy	Fair	partly cloudy	fair	WPH	✓
27/04/05 03:15	Cloudy	Fair	partly cloudy	fair	WPH	✓
27/04/05 04:15	Cloudy	Fair	mostly cloudy	fair	WPH	×
27/04/05 05:15	Cloudy	Fair	cloudy	fair	WPH	×

amount time. We collected weather forecasts for New York City during a period of one month (April 2005). However, here we will present the results for a particular week.

4.2 Results

The results we present are based on one week of weather data from 22 to 29 April 2005. As already mentioned we collect, at every hour, the next hour's weather forecast. Then, at the next hour, we can compare the actual current weather with the previous forecast and check whether the forecast was correct. If it was, we increment the success counter for this source. Thus, in the case of weather data, the service selector can autonomously compute the dependability of the alternative sources.

Table 1 shows an extract of the weather log. The adaptive column shows which weather source the service selector chooses and whether this forecast turned out to be correct. For instance, in the second weather row, the selector chose Weather.Com, which turned out to give an incorrect forecast. However, in the following row, WeatherPerHour was chosen, which resulted in a correct forecast. From the log it can be seen that we accept small variations in the forecast as correct. So, "partly cloudy" is accepted as a correct forecast of "fair", while "mostly cloudy" is not.

The selector chooses the forecast of the weather source with the currently highest dependability. To allow for dynamic dependability, this is computed as the success rate in the last n forecasts. Thus, with $n = 10$ a dependability of 0.7 means that in the past 10 hours 7 forecasts out of 10 turned out to be correct. When both sources are equally dependable in the same time window, then the one with highest long-term dependability (i.e. highest success rate considering all past forecasts) is chosen.

An issue which is critical to the success of the adaptive source is the size of the sliding window. Figure 2 shows the success rate for the adaptive source in this week with a window size between 1 and 20. Interestingly, the maximum success rate is achieved with a window size of only 1, i.e. selecting the best source based on only the previous outcome. Indeed, over the course of one month of weather

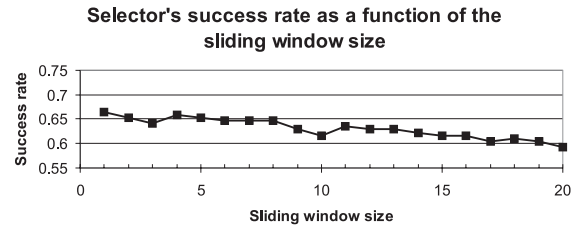


Figure 2. Success rate.

data, we have observed that the best result is obtained with a window size ≤ 5 . However, using a very small window size may be undesirable, as it can lead to a constant switch between the different sources. For instance, if the selection of a service required subscription to the service, then a switch to another provider would entail cancelling the subscription at the old source and requesting it at the new source, which involves network communication.

With a window of size 1, the total success rate of the selector over the week is 66.5%. In comparison, Weather.Com's total success rate is 59.5% and WeatherPerHour's 61.3%. Without dependability, if we assume that we would select one of the two sources equiprobably, we would get an average success rate of 60.4%. Thus, the use of dependability in the service selector improved average forecast success rate by 6.1%.

The theoretical maximum success rate, achieved when the selector always chooses the actual best alternative, is 75.0%. This value is less than 100%, because in 25% of forecasts, both weather sources erred (see the areas in Figure 4 where both sources have black bars) and therefore there was no way the selector could choose a source that would be successful.

Figure 4 shows a chart graphically representing the forecasts (the lines marked BN will be described in the next section). The week previous to the one on which we ran the selector is also shown.

In the first two rows, a black bar indicates a wrong forecast, while white means correct forecast. Thus, notice

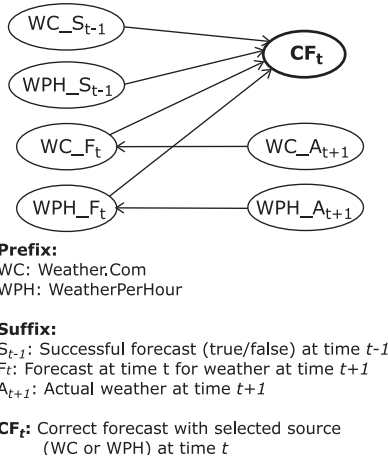


Figure 3. Bayesian Network for estimation of weather source's dependability.

that we have chosen a very critical moment for our experiment, as in previous week both sources were very reliable, whereas in the chosen week both sources have trouble correctly predicting the weather. We chose such a week (they are quite common) as it is a particularly good example of where an adaptive source using dependability can improve on the average quality of the original alternatives.

In the “Adapt SR Selection” row we show the selection based on dependability: a white bar means that Weather.Com was selected, a black one WeatherPerHour. Finally, the “Adapt SR Correct” shows whether the selector was correct (compare this to the correctness of Weather.Com and WeatherPerHour in the first two rows).

4.3 Trying to improve selection with a Bayesian Network

We now describe an approach that is generally applicable and usually improves the estimation of a service provider's dependability prediction. However, we will show that in our case study this approach performs surprisingly disappointingly compared to the very simple success rate method with window size 1 of the previous section.

We point out that what we are trying to do is predict a service provider's future dependability based on its behaviour so far. This can be done with a Bayesian Network (BN).

Broadly speaking, Bayesian Networks capture statistical correlation between random variables. We will now further explain BNs by looking at our specific case study. Figure 3 shows the BN we have constructed to predict a weather forecast service's dependability. It is a graph in which the nodes represent random variables. A directed edge from a

random variable X to Y means that X has a *direct influence* on Y , or in probabilistic terms Y is directly conditional on X . Thus, what the weather will actually turn out to be in the next hour (at time t hours) has a direct influence on what a weather service will forecast (at time $t - 1$ hours) for the next hour. Moreover, we make the probability of successful forecast of a weather source *depend on what it forecasts and on whether it was previously successful in forecasting this weather*. The former dependency (what it forecasts) is drawn from the assumption—which has been observed in actual logs—that some weathers, e.g. *sunny*, may be easier to forecast than others, e.g. *mostly cloudy*. The latter dependency (whether it was previously successful) is drawn from the assumption that if a weather source has been successful in the past in forecasting *sunny*, then it is also likely to forecast *sunny* correctly in the future, or likewise if the weather source has been unsuccessful in correctly forecasting *mostly cloudy* in the past, it is likely to be unsuccessful also in the future in correctly forecasting *mostly cloudy* (whether this assumption generally holds throughout the year can be debated).

Once the BN graph has been defined, it has to be completed by defining a conditional probability table (CPT) for each random variable. Every random variable is conditional on the variables it is directly influenced by, if any. Thus, we need to define the probabilities $P(WC_F_t|WC_A_{t+1})$, $P(WPH_F_t|WPH_A_{t+1})$, $P(WC_A_{t+1})$, $P(WPH_A_{t+1})$, $P(WC_S_{t-1})$, $P(WPH_S_{t-1})$.

$P(WC_F_t|WC_A_{t+1})$ and $P(WPH_F_t|WPH_A_{t+1})$ are the probability of a forecast value occurring given what the weather will actually be (which is still unknown at the time of the forecast). So, if the weather is going to be *sunny*, it is (hopefully) more likely that the forecast will be *sunny* as opposed to *heavy rain*. This CPT can be compiled by keeping track of the number of occurrences of every combination of pair (forecast, actual weather) observed up to the present. $P(WC_A_{t+1})$ and $P(WPH_A_{t+1})$ are the probability of each weather actually occurring. Again, we need to count the number of occurrences of each type of weather observed up to the present. $P(WC_S_{t-1})$ and $P(WPH_S_{t-1})$ are the probability of the weather source's forecast being correct. This is simply the success rate observed up to now.

Now that the BN has been fully defined, we can compute the dependability of a source by evaluating $P(CF_t|...)$ for both Weather.Com and WeatherPerHour, according to the dependencies of the BN graph, which results in the formula shown in Figure 5.

Notice that, at the time a source is selected (time t), i.e. d in formula of Figure 5 is computed, the actual weather at the next hour (at time $t + 1$) is still unknown. However, this is not a problem, as CF_t is not directly condi-

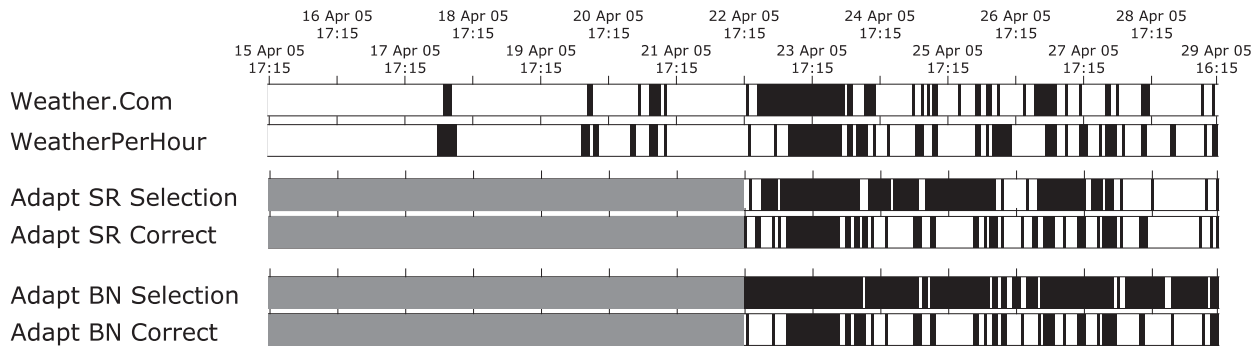


Figure 4. Chart of forecast correctness.

tional on $WC_{A_{t+1}}$ and $WPH_{A_{t+1}}$, which are therefore called *hidden variables* (there is no arrow from $WC_{A_{t+1}}$ or $WPH_{A_{t+1}}$ to CF_t). In Bayesian inference, we sum over all possible values of hidden variables, as their value is unknown.

Also, notice that whether one source is predicted to be successful in its forecast depends not only on its own past behaviour, but also on the other source's behaviour. In other words, the BN takes into account any statistical correlation that there may happen to be between the two weather sources³.

The BN is used in the following way. At time t we use the BN to compute $d = P(CF_t | \dots)$ for both weather sources based on their forecast for the next hour and select the forecast of the most dependable source (highest d). Then, in the next hour we check the actual whether (as delivered by each weather source) and update the CPTs in the BN accordingly. This process is repeated every hour. Thus, the BN is continuously trained while in use.

In practice, instead of simply selecting the source with highest $d = P(CF_t | \dots)$, you could actually select each one with probability d . However, here we wish to present average results and thus take the source that on average would be selected most often.

4.4 Results with BN approach

The Bayesian net was used to compute the dependability for the same week as with the success rate method described in Section 4.1. However, to fill the CPTs, the BN was trained with 24 days of hourly forecasts prior to this week. The chart in Figure 4 only shows one week of training data prior to the selection week.

The results are quite surprising. In the week from 22 to 29 April 2005, the BN achieves a total success rate of

³We have also experimented with a simpler BN that treats the two weather sources independently, but usually obtained worse results.

only 65.5%, compared to the 66.4% using the simple technique of Section 4.1, where a source is selected based on whether it successful in the previous forecast. Thus a technique which is completely trivial to implement, and has minuscule and constant time complexity, achieves similar and sometimes better results (as in this week) than a much more complex prediction technique.

To better understand this result, some observations are in order:

- Out of the one month of data collected, we chose to present this week, as it is a week where the two weather sources perform particularly badly, and thus there is much room for improvement for the service selector. Furthermore, as can be seen from Figure 4, the weather previous to this week was very predictable, while the weather in the week in question is not. Thus, the training data that the BN receives does not reflect well the behaviour of the weather sources during the test week. This scenario however is not a unique case, but recurs in our weather logs.
- Using the BN approach is actually better than using the previous approach with any window size > 5 (the BN approach has, in a sense, a window of infinite size). It is actually surprising that a small window size gives the best result, indicating that the dependability of weather forecasts is highly variable and does not follow predictable trends (weather as such is already difficult to predict, particularly in April).
- Using the simple success rate technique, we can achieve good results (in the specific case of hourly weather forecasts) with very little overhead, but its success is entirely dependent on the choice of the window size. In fact, while often a window size of 1 is optimal there are weeks when a size of 5 is better. Furthermore, in the test week, a window size of 20 or more produced a worse success rate than WeatherPerHour. In comparison, a Bayesian network approach is much

$$\begin{aligned}
d &= P(CF_t | WC.S_{t-1}, WPH.S_{t-1}, WC.F_t, WPH.F_t) \\
&= \alpha \sum_{y_{WC}, y_{WPH}} P(CF_t, WC.S_{t-1}, WPH.S_{t-1}, WC.F_t, WPH.F_t, y_{WC}, y_{WPH}) \\
&= \alpha \sum_{y_{WC}, y_{WPH}} P(CF_t | WC.S_{t-1}, WPH.S_{t-1}, WC.F_t, WPH.F_t) \cdot \\
&\quad \cdot P(WC.F_t | y_{WC}) \cdot P(WPH.F_t | y_{WPH}) \cdot P(y_{WC}) \cdot P(y_{WPH})
\end{aligned}$$

y_{WC} represents the hidden variable $WC.A_{t+1}$ and is summed over all weather values of WC.

y_{WPH} represents the hidden variable $WPH.A_{t+1}$ and is summed over all weather values of WPH.

α is the normalisation factor, i.e. makes sure that $\alpha P(CF_t = WC | \dots) + \alpha P(CF_t = WPH | \dots) = 1$.

Figure 5. Formula for computing dependability using BN (Bayes' rule is applied at each step).

more processing-intensive, returns similar results, but has no free parameters, and thus consistently returns good results.

Running the experiment over the entire month, the BN approach actually performed better than the simple technique (with no prior training as no prior data was available): 81.8 % success rate compared to 79.8% for the simple technique (this is the best result, achieved with a window of size 5), while Weather.Com and WeatherPerHour achieved 74.2% and 75.3%, respectively, with a theoretical maximum of 87.3% (i.e. in 87.3% of forecasts at least one of the original sources got the forecast right).

4.5 Final remarks

Even using a better Bayesian network than the one presented, we believe it is difficult to achieve a result closer to the optimum without more information on the forecast data received, e.g. advertised probability of correctness, which would thus be a QoI metric of weather forecast services. The reason is that the success rate of the two weather sources is very unpredictable: a weather source can often perform well for a long time, and thus any statistical selection approach will regard this source highly, only to suddenly fail on and off in its forecast for a certain period. The weather log in Figure 4.1 is a good example of this case.

5 Conclusions

We have presented dependability as a metric of adaptive service selection that is not advertised by the service provider, but is determined based on recent experience with the provider. Dependability evaluates either the reliability of the Quality of Information parameters advertised by the service or, if these are unavailable, the reliability of the service's data itself. An abstraction layer between service

provider and client can use dependability to automatically select and switch at run-time between providers such that the overall quality of the information received is not only as good as the best alternative (which is unknown at the time of selection, but estimated using dependability), but even better than selecting any fixed single alternative.

We have also described two generally applicable ways of estimating dependability. The first computes dependability as the success rate in the recent past. This method is extremely lightweight and ideal when installing a service selector on a device with little computing power, e.g. in wireless sensor networks. However, it has a critical parameter, the window size, that determines how well this technique performs, and depends on the variability and unpredictability of the services' reliability, although it performs surprisingly well with weather sources, which are highly unpredictable. The second method uses a Bayesian network to compute dependability, whereby the actual layout of the network is service-type specific. This approach is far more processing intensive, but has no parameters that have to be tweaked. Furthermore, it can potentially produce better results than the first technique, although our case study shows that this depends on how good past training data is useful in predicting the future reliability of the provider.

References

- [1] A. Harter, A. Hopper, P. Steggle, A. Ward, and P. Webster. The anatomy of a context-aware application. *Wireless Networks*, 8(2-3):187-197, 2002.
- [2] J. Yin, Q. Yang, and L. Ni. Adaptive temporal radio maps for indoor location estimation. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2005.