Insights on the configuration and performances of SOME/IP Service Discovery

Jan R. Seyler, Daimler AG, Mercedes-Benz Cars Development
Nicolas Navet, University of Luxembourg
Loïc Fejoz, RealTime-at-Work

Authors' preprints

Abstract

Scalable Service-Oriented Middleware on IP (SOME/IP) is a proposal aimed at providing service-oriented communication in vehicles. SOME/IP nodes are able to dynamically discover and subscribe to available services through the SOME/IP Service Discovery protocol (SOME/IP SD). In this context, a key performance criterion to achieve the required responsiveness is the subscription latency that is the time it takes for a client to subscribe to a service. In this paper we provide a recap of SOME/SD and list a number of assumptions based on what we can foresee about the use of SOME/IP in the automotive domain. Then, we identify the factors having an effect on the subscription latency, and, by sensitivity analysis, quantify their importance regarding the worst-case service subscription latency. The analysis and experiments in this study provide practical insights into how to best configure SOME/IP SD protocol.

Introduction

Context of the work. Scalable Service-Oriented Middleware on IP (SOME/IP) is a protocol on top of IP that provides service-oriented communication in vehicles. SOME/IP takes advantage of Ethernet's bandwidth, maximum frame size and IP multicast capabilities to support larger messages and reduce the overhead with regard to legacy protocols such as CAN and FlexRay. SOME/IP nodes are able to dynamically discover and subscribe to available services through the SOME/IP Service Discovery protocol (SOME/IP SD), with efficient routing strategies mixing unicast and multicast communications. Though it remains to be ascertained which use-cases would most benefit from SOME/IP, it opens the door to more flexibility in automotive communications, such as the ability to dynamically add new services or migrate existing services.

Problem definition and contribution of the paper. SOME/IP SD induces overhead at run-time due to the exchange of signaling and state messages between nodes, which will affect the rest of the traffic. In addition configuring SOME/IP involves setting numerous parameters which have some impact on the service subscription delays and the additional network load. The specification documents, nor, to the best of our knowledge, any studies published in the literature so far, provide practical guidelines on how to best configure and deploy SOME/IP SD. This work explores this question and aims to give insights into how to configure SOME/IP SD so as to:

- ensure system responsiveness: guarantee the maximum time it takes for the stations to subscribe to all the services they require,
- Identify, amongst all the system characteristics and SOME/IP SD protocol parameters, the ones that have the largest impact on the subscription latency. The observations drawn for the experiments of this paper are intended to help the designer come up with adequate trade-offs between network load (*i.e.*, SOME/IP SD overhead) and delays in connecting to services.

These questions are studied experimentally on a configuration kept simple on purpose, so as to enable us identify the main factors of the subscription latencies.

Related work. The existing work on SOME/IP SD is limited since the proposal is recent [2] and is still under development and test. In addition to the specification, there is a set of overview papers and presentations about the protocol and its rationale (see [3,4,5,6]). To the best of our knowledge, the first publication addressing the temporal performances of SOME/IP SD is [1]. In the latter study, a worst-case analysis of the subscription delay is proposed, applied to a case-study and validated against simulation results and data traces recorded on a test-bed. This paper builds on [1] and explores by sensitivity analysis the impact of the configuration of SOME/IP on the worst-case subscription latency.

Outline of SOME/IP Service Discovery

SOME/IP is a service oriented protocol for the automotive use in which services are offered by nodes and clients can subscribe to them. SOME/IP SD, the protocol part of SOME/IP in charge of service discovery and connection management, is described in the specification document [2].

In the following, a *client* refers to an entity wanting to subscribe to a single *service*, but there can be several clients and several services on the same ECU. It should be noted that there can be also several *instances* of the same service available on different ECUs.

When a service is available, that is not *down* in SOME/IP terminology, it has to be in one of the three following functioning phases: the *initial wait phase*, the *repetition phase*, and the *main phase*. A client that is active (*i.e.*, not down) has to be also in one of these phases (see Figure 1).

A service broadcasts *offer messages* on the network. These messages include a type, a service ID, an instance ID, a version, and a time-to-live fields. Upon the receipt of an *offer* message, interested clients will subscribe to the service. However, in order to reduce the subscription time, clients can also broadcast *find messages* to notify their need for a certain service and receive *offer* messages in response.

As this is described next, there are random waiting delays in the sequence of operations of SOME/IP SD so that not all services and clients start to operate at the same time. This helps to better spread over time the network and CPU load induced by the protocol.

Initial wait phase

Client behavior: a client enters the initial wait phase upon the request of the application layer. It remains in this phase for a time randomly chosen between *SdClientTimerInitialFindDelayMin* and *SdClientTimerInitialFindDelayMax* [2, §7.7.2]. In this phase the client remains silent. If an offer of the service that the client looks for is received in this phase, the client asks to subscribe to the service and then goes directly to its main phase, skipping the end of the initial wait phase and the repetition phase.

Service behavior: a service set to available by the application layer triggers the entry into the initial wait phase. Like the client, the service remains in this phase, without sending any message, for a time randomly chosen between *SdServerTimerInitialOfferDelayMin* and *SdServerTimerInitialOfferDelayMax* [2, §7.6.1]. But unlike the client, any find message received in this phase is ignored by the service (i.e., the service does not buffer subscription requests) and the repetition phase cannot be skipped. This has the consequence that the duration of the initial wait phase of a service is a part of the subscription latency that cannot be shortened. This asymmetry in the behavior of clients and services is noteworthy and, as discussed later, will have to be considered when setting the protocol parameters.

Repetition phase

Client behavior: the client sends a pre-defined number of *offer* messages (*SdClientTimerInitialFindRepetitionMax*) with an exponentially increasing waiting time between successive messages. The first message is sent upon the entry in the repetition phase, then the following messages are sent with a delay equal to $SdClientTimerInitialFindRepetitionBaseDelay \cdot 2^i$.
When a client receives an *offer* message from the service in the repetition phase, the client requests subscription to the service and then goes directly to its main phase.

Service behavior: The service sends an *offer* message upon its entry in the repetition phase, then the following messages are sent with an exponential increasing delay equal to $SdServerTimerInitialOfferRepetitionBaseDelay \cdot 2^i$. After a predefined number of *offer* messages in the repetition phase (*SdServerTimerInitialOfferRepetitionMax*), the server enters the *main phase*. When the service receives a *find* message from a client, it waits during a random time from *SdServerTimerRequestResponseMinDelay* to *SdServerTimerRequestResponseMaxDelay* [cf. 2, §7.5.3] and sends a unicast *offer* message to the sender of the *find* message.

Main phase

In this phase, corresponding to a stable operation mode unlike what happens at the startup of the system or after the transition to a new vehicle functioning mode, the network load induced by SOME/IP SD can be reduced.

In the main phase, a client does not send any *find* message, but will answer to server's offer message if needed. The service sends cyclically offer message with a period equal to *SdServerTimerOfferCyclicDelay* (see [2], §7.6.3). It will also answer to *find* message as in the repetition phase.

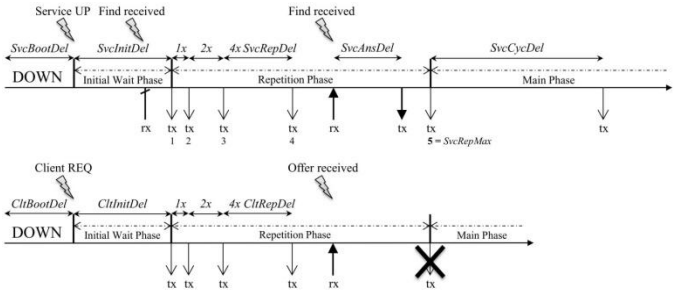

Figure 1. The startup phase of SOME/IP SD for a service and a client (figure from [2]).

Functioning modes for clients and services

SOME/IP SD communication stacks allow configuring a client in two possible modes: the *request mode*, where it will send *find* messages in the repetition phase, or the *listen mode*, where it will only wait for *offer* messages. Similarly, a service can either be in offer mode, where it will send *offer* messages in the repetition phase and main phase, or in silent mode, where it will only respond to *find* messages from clients. The request mode for both the client and the server will be the focus of the paper hereafter, since it is in our understanding of the specification the normal functioning mode, and anyway the one leading to the lowest subscription latency.

Assumptions in the automotive context

What we can foresee from the future use of SOME/IP in the automotive domain leads to the following assumptions:

1. The switched Ethernet network is made up of a few switches and a few tens of nodes,
2. A node may contain several clients of distinct remote services and several services,
3. The total number of services offered ranges from a few tens to a few hundreds,
4. The nodes are not synchronized on startup, and thus will have varying boot times,
5. A client node typically requests a fraction of the total number of services offered (at most a few tens),

6. A client may require first to subscribe to a set of services before offering its own services,
7. The services might not be requested and offered all the time because the nodes can alternate between on and off due to partial networking, and because services offered and needed will depend on the functioning mode,
8. Response times of the frames on the Ethernet network are less than a few ms,
9. In addition to the SOME/IP and SOME/IP-SD traffic, there are other frames exchanged which can be critical and be subject to timing constraints.

In this study, we additionally assume that every service does only have one unique instance within the system (*i.e.*, no service redundancy), the service is identified by a unique index j.

In the following we will be interested in the latency from the time a client is operational (*i.e.*, it leaves the DOWN region in figure 1) until it receives an offer for the requested service. This latency can be called the *pre-subscription time*. From this point on, completing the subscription requires the immediate exchange of two additional *ack* messages (one from the client and one from the server) which will just add an upper bounded delay to the subscription latency but not change the overall temporal behavior of the service subscription process.

Factors impacting the client subscription latency

The time it takes for the clients to subscribe to services will affect the responsiveness of the system, and, possibly in some use-cases such as ADAS functions, it may even impact the vehicle safety. We identify the following factors influencing the client subscription time:

1. The temporal offset between the times at which the client and the service become operational. This depends on the client's and service's ECU startup times, as well as application software and communication stack latencies on the clients' and services' ends.
2. The functioning mode of the services and clients: silent mode will have a detrimental effect on the subscription latency.
3. SOME/IP SD protocol parameters such as the length of the initial phases, the cycle time in the main phase for the services, the maximum number of transmissions in the repetition phase for both clients and services, the base delay used in the repetition phases, etc.
4. The time it takes for a service to answer a *find* message from a client.
5. To a lesser extent, the frame latencies on the network and the possible frame losses.

These factors are reflected in the formulas of the analysis proposed in [1].

It is worth pointing out that if the service is late with regard to the client (see figure [2], case 1), then the only protocol parameter impacting the subscription latency is the duration of the initial wait phase of the service, since the client will register on the message sent by the service at the end of its initial wait phase. This explains the importance of this parameter value, as it will be shown in the experiments.

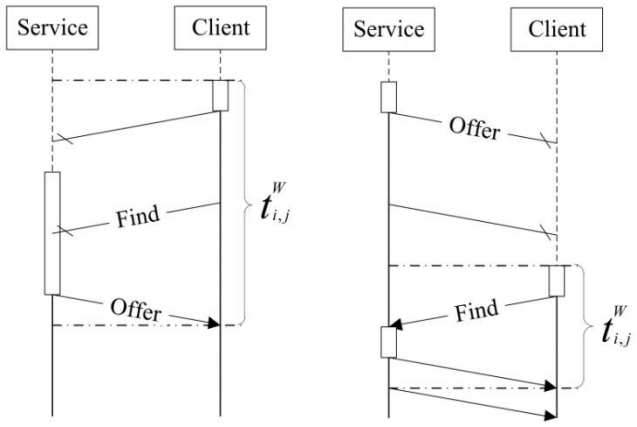

Figure 2. Message exchanges during the subscription of a service when both client and services are in request mode. Case 1 (left): the service is late, the client will be able to subscribe after receiving the message sent by the server at the start of the service repetition phase. Case 2 (right): the client is late and will subscribe after the reception by the server of the first message it sends in the repetition phase. The quantity $t^W_{i,j}$ is the service subscription latency of client *i* when subscribing to service *j*.

On the other hand, if the client is late with regard to the service (see figure [2], case 2), then it can register either on an *offer* message sent by the service (either in the repetition or main phase) or on a *find* message it sends. Here, the protocol parameters affecting the subscription latency are the following:

- The duration of the client's and service's initial wait phase,
- The maximum number of repetitions and the repetition delay for the server,
- The time it takes for the server to answer a *find* message from the client and send back in response its *offer* message,

However, client's parameters for the repetition phase will not matter since the client will register at the latest on the first message it sends at the end of its initial wait phase.

This coarse-grained analysis however ignores the possibility of transmission errors and message losses due to buffer overflows in the switches, and it assumes constant communications latencies whatever the messages. In the rest of the paper, we will quantify the weight of the different factors having an effect on the subscription latency, and discuss what it involves in terms of configuration.

Analysis of the sensitivity to the duration of the initial wait phases

Using the analysis from [1] we study here how the length of the initial phases both on the client and service ends affect the worst-case subscription latencies. The system under consideration has the following characteristics:

- One client and one service on remote ECUs,
- Upper bounded communication delays equal to 5ms,
- The maximum ECU boot delays ranges from 0 to 2ms for both the client and the service,
- The maximum repetition delay for the server (*SvrRepMax*) is set to 3 with a repetition base delay set to 0.05ms,

- The server cycle delay is set to 2 ms.

In this first experiment, both the values of the ECU boot delays and the value of the initial wait phase range from 0 to 2ms for the client and service. The sum of these two quantities is denoted by *t_S_init* for the server and by *t_C_init* for the client. As can be seen on Figure 3, three cases can be distinguished:

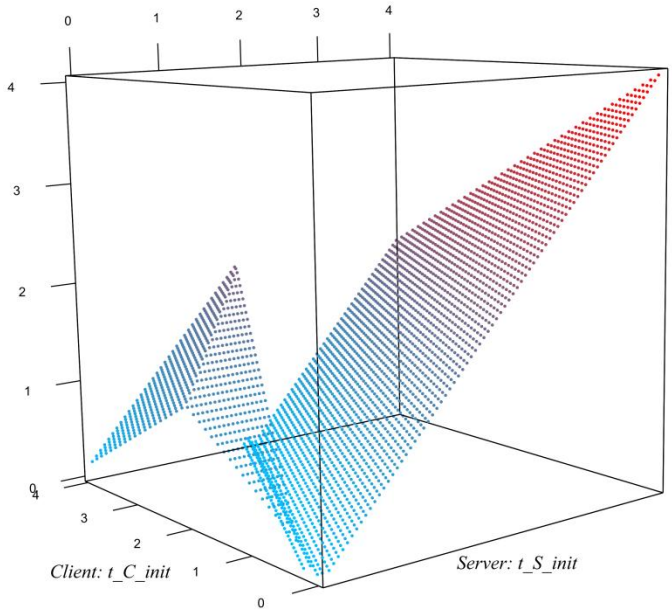

Figure 3: Worst-case service subscription latency for varying values of the end of the initial wait phase of the server and the client (unit: ms).

a) The increase with a linear slope on the right-hand side of the graph corresponds to cases where the service exits the initial wait phase after the end of the client's boot delay. Without some control on the ECU synchronization, this delay can only be shortened by reducing the duration of the server's initial wait phase.
b) The line showing very small latencies in the center corresponds to cases where the service exits its initial wait phase at around the same time as the client enters its initial wait phase (*i.e.*, end of "DOWN" period in Figure 1). The client will then subscribe on the *offer* message sent by the server at the beginning of the repetition phase.
c) The two surfaces on the left reach a maximum of 2 milliseconds. These are cases where the client is late with regard to the service and it has to wait during at most one server cycle delay.

These results suggest to us that

- The boot time of the services and the value of their initial wait phase are the main factors that influence the client subscription latency.
- Reducing the service cycle delay leads to a linear reduction of the latency when the services are ready before the clients. This can be the case when services are always available and clients subscribe upon mode changes or functions activation.

The next experiment, whose results are shown in Figure 4, highlights the importance of the time between the client boot delay (*CltBootDel*) and the time at which the service exits its initial wait phase (*t_S_init*).

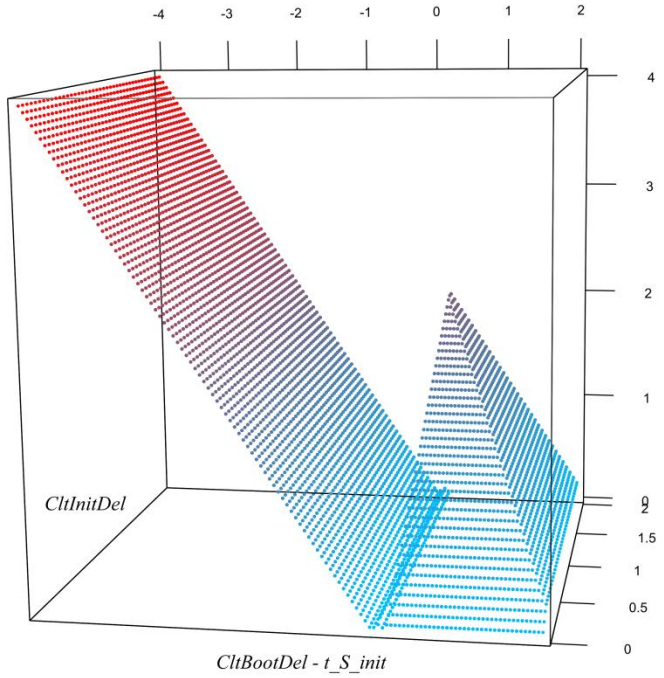

Figure 4: Worst-case service subscription latency for different startup offsets between the client and the service, and varying durations of the client's initial wait phase (unit: ms).

In Figure 4, the x-axis is the quantity *CltBootDel-t_S_init*, that is a temporal offset between the client's and the server's start of operation. Indeed, because the client can already respond to *offer* messages from the service during its initial wait phase, *CltBootDel* is considered and not *t_C_init*. The y-axis is the duration of the client's initial wait phase (*CltInitDel*). Two main situations can be identified:

a) *CltBootDel-t_S_init<0*, on the left of the graph: these are cases where the service is late (see explanations for case a) of Figure 3).
b) *CltBootDel-t_S_init>0*, on the right of the graph: these are situations where the client is late. The maximum latency in this area is the minimum between the service cycle time and the length of the client's initial wait phase.

These results show that reducing the length of the client's initial wait phase leads to a linear reduction of the worst-case subscription time, but this holds true exclusively in the case where the service is operational before the client.

Analysis of the sensitivity to the server's cyclic delay in the main phase

Here we explore the influence of the server's cyclic delay (*SvcCycDel* for short), which is the time between two successive *offer* messages in the main phase. In Figure 5, *SvcCycDel* varies from 0 to 4 on the x-axis, while the quantity *CltBootDel-t_S_init* ranges from -6 to 6.

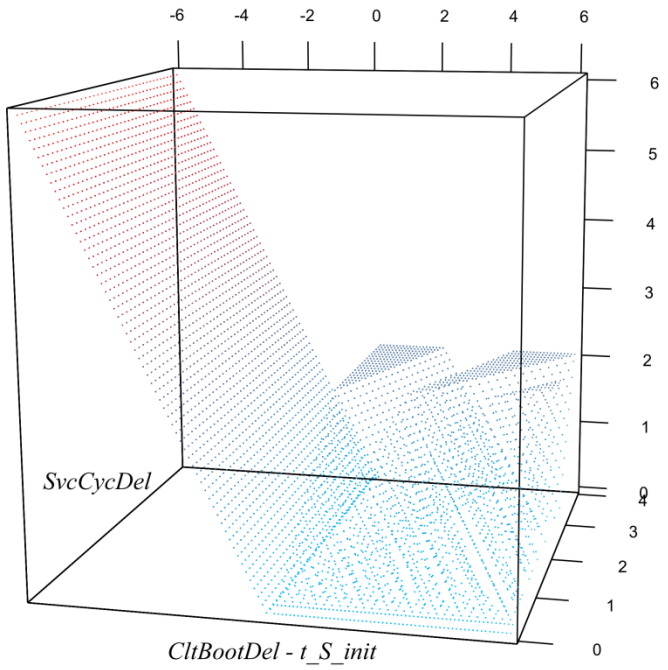

Figure 5: Influence of the server's cyclic delay on the worst-case service subscription latency for different startup offsets between the client and the service (unit: ms).

The left-hand side of the graph (*i.e.*, *CltBootDel - t_S_init<0* on the x-axis*)* are cases where the server is not ready yet to accept subscriptions. The surface on the right side of the graph reaches 2ms, which is the value of the client's initial wait phase. As figure 5 shows in the region where *SvcCycDel<2*, it is possible to reduce with certainty the worst-case subscription latency by choosing values of the server's cycle delay smaller than the client' initial wait phase. Here, there is however a tradeoff to be found between the network load and the subscription latency.

Analysis of the sensitivity to parameters of the service in the repetition phase

There are two parameters to set for the repetition phase of the service: the maximum number of repetitions (*SvcRepMax*) and the repetition delay (*SvcRepDel*). These parameters only influence the subscription delay when the service is ready before the client. In that situation, the client might register on an *offer* message sent in the repetition phase instead of the first *find* message it sends at the end of its initial wait phase. The maximum gain is thus limited to the duration of the client initial phase.

Figure 6 shows the influence of *SvcRepMax* and *SvcRepDel* for varying boot configuration of the service and the client. *SvcRepMax* ranges from 0 to 4 on the y-axis, while the latencies for values of *SvcRepDel* from 0 to 0.1 appear vertically on the graph (*i.e.*, vertical bars). The color of these latencies varies from blue (smallest value of *SvcRepDel*) to red (largest values).

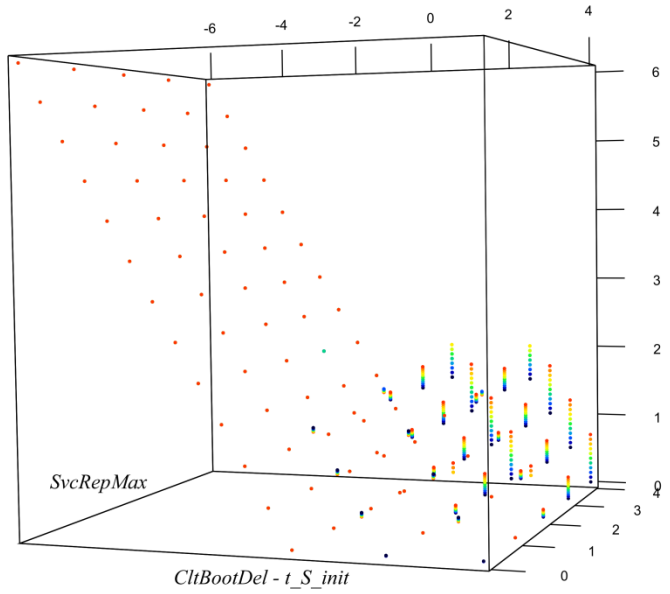

Figure 6: Influence of the server's parameters in repetition mode on the worst-case service subscription latency for different startup offsets between the client and the service (unit: ms).

The parameters of the server's repetition phase only play a role when the server exits its initial wait phase before the client exits its own initial wait phase. Reducing the value of *SvcRepDelay* and increasing *SvcRepMax* actually leads to a reduction of the client's subscription delay but to a limited extent. The larger *SvcRepMax*, the larger the gain there is to reduce *SvcRepDelay*. However, a gain is only achievable when the client exits the initial wait phase not too long after the start of the repetition phase of the server, otherwise the server might already be in its main phase. Also it must made sure that the value of *SvcRepMax* and *SvcRepDelay* are not such that the time between two successive *offer* messages in the repetition phase is larger than *SvcCycDel* otherwise there will be a loss in performances. All in all, the influence of the parameters of the server's repetition phase is limited both in magnitude and scope, since they have an effect only when server and client become operational close to each other in time.

Analysis of the sensitivity to the server's answer delay

As shown in Figure 2 (case b), the server receiving a *find* message from a client will send in response an *offer* message. The time taken by the service to answer is called the answer delay (*SvcAnsDel*).

In the experiments whose results are shown in Figure 7, the value of *SvcAnsDel* varies from 0 to 2. The three lines showing very small latencies correspond to cases where the client receives an *offer* message from the service (either the first message in the repetition phase or a message in the main phase) at the same time as it enters into its initial wait phase.

What can be observed is that larger values of *SvcAnsDel* increase the subscription latency. However, whatever the value of *SvcAnsDel*, the latency will not go above *SvcCycDel* (here 2 ms) since the client will then subscribe through an *offer* message sent by the service in the main phase.

The server's answer delay has more influence in the situation where the client's initial wait phase is short, and the client is more likely to register to the service through the *find* message it sends at the end of the initial phase, and not through an *offer* message sent by the service.

It should be pointed out that if the answer delay is larger than the server's cycle time, then the client will always register to a service through an *offer* message, and never through the *find* messages it broadcasts. In such parameter ranges, the client could be as well configured in silent mode without a detrimental effect on the temporal performances.

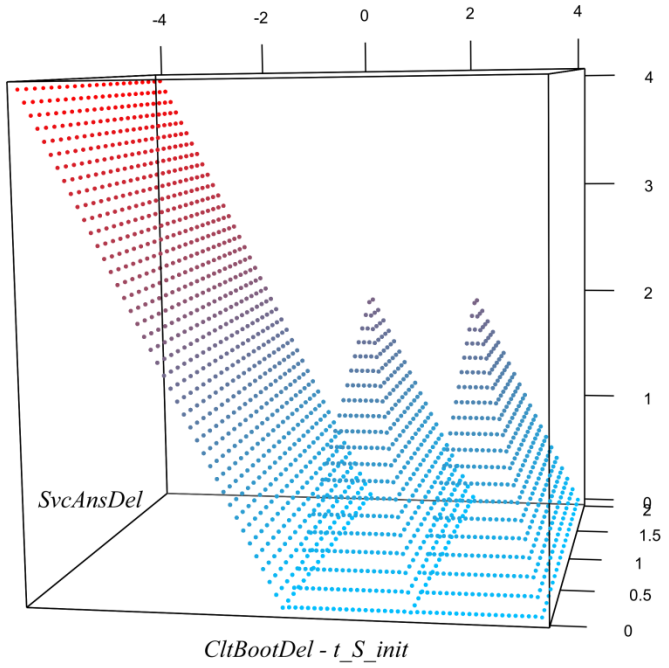

Figure 7: Influence of the server's answer delay to a *find* message on the worst-case service subscription latency for different startup offsets between the client and the service (unit: ms).

Summary/Conclusions

This paper studies the service subscription latencies on SOME/IP SD when clients and services are configured in request mode, the most efficient configuration choice from a temporal point of view. This study is, to the best of our knowledge, the first published work identifying and quantifying the importance of the factors influencing the service subscription latency for different system configurations.

The results of our experiments suggest that the parameters of the servers are the main factors that influence the client subscription latency: the boot time of the services and the value of the initial wait phases. Also, reducing the service cycle delay leads to a linear reduction of the latency when the service is operational before the clients. The duration of the client's initial wait phase, the parameters of the servers in the repetition phase, and the server's answer delays to *find* messages have had a more limited impact in our experiments.

Future works could consider more grained models of the system, with for instance, more realistic communication latencies, communication stacks and use-cases of SOME/IP by the applicative level software. Another perspective is to quantify the network overhead created by SOME/IP SD in its different functioning phases. Ultimately, what we aim at is to develop configuration algorithms that automate the choice of SOME/IP and SOME/IP SD parameters which regard to a set of performance objectives and constraints. This would ease and speed up the design and deployment of SOME/IP based communication architectures.

References

1. J. Seyler, T. Streichert, M. Glaß, N. Navet, J. Teich, "Formal Analysis of the Startup Delay of SOME/IP Service Discovery", to appear at DATE 2015, Grenoble, France, March 9-13, 2015.
2. AUTOSAR, "Specification of Service Discovery", 2013. Available at url: http://www.autosar.org.
3. R. Stolpe, "Toolunterstützung für Ethernet und SOME/IP", Hanser automotive networks, 2013. Also available in English from www.dspace.com under the title "Support for Service-Based Communication via Ethernet and SOME/IP from Real-Time Simulation Systems".
4. L. Völker, "SOME/IP Service Discovery – the Need for Service Discovery in the vehicle", Vector Ethernet Symposium, Stuttgart, Germany, May 2014.
5. L. Völker, "SOME/IP – Die Middleware für Ethernet-basierte Kommunikation", Hanser automotive networks, 2013.
6. M. Ziehensack, T. M. Galla, "AUTOSAR 4.1 IP-Stack – bereit für die Serie!", Hanser automotive networks, Nov. 2012.
7. L. Völker, "SOME/IP Tutorial", OPEN Alliance Meeting, Leinfelden-Echterdingen, Germany, Oct. 2013.
8. L. Völker, "Communication protocols for Ethernet in the vehicle", Automotive BUS Systems + Ethernet, Sindelfingen, Germany, Dec. 2013.