Project no.	609551
Project acronym:	SyncFree
Project title:	*Large-scale computation without synchronisation*

European Seventh Framework Programme
ICT call 10

Deliverable reference number and title:	D 5.3 Plan and design of massive scale crowd-sourced empirical validation.
Due date of deliverable:	September 29, 2015
Actual submission date:	4 February 2016
Start date of project:	1 October 2013
Duration:	36 months
Name and organisation of lead editor for this deliverable:	Rovio Entertainment Oy
Revision:	1.1
Dissemination level:	CO

Contents

1 Executive Summary

The document describes the experiments proposed by the Syncfree partners to validate CRDT solutions. All these experiments evaluates efficiency of using the CRDTs solutions when scaled either in number of replica, database, updates or a combination of them. The experiment results would underline the benefits of using a CRDT solution in a real world problem.

We will use the FMK use case to evaluate the transactional support in Antidote platform and see if it can resolve problems identified by Trifork when using Riak.

The FMK system is used by the Danish Healthcare to track patients and their treatment, in particular their prescriptions. All events relating to the patients are recorded, such as a drug that have been administered by a nurse or handed out at a pharmacy. The FMK has high availability as one of its main design criteria. This is based on the fact that, in the context of health care decisions, having some information is better than none and often having old information is better than none.

This use case is the one that explores the insights of SyncFree the most in terms of going beyond CRDTs and finding the limitations — if any — of the approach. Due to the criticality of the FMK system, we will simulate (a subset of) its functionality in a controlled environment using fake patient data.

The decision of using the FMK use case instead of the Virtual Wallet use case is based on the feedback received at the second year review, which recommended that we investigated the possibility of using the FMK use case. It has turned out that the FMK use case is good for exercising the innovations of SyncFree in relation to specific problems identified by the industrial partner Trifork. As the moment, we expect to be able to scale the FMK experiments to appropriate numbers using only free computing resources available in Grid 5000. Thus, we are not requesting any additional funds beyond what has already been allocated to Rovio.

The Rovio use case verifies the effectiveness of using a geo-replicated CRDTs as the alternative backend store for Ad serving logic that handles massive concurrent user load. ESL will validate the use of CRDTs as the solution for solving the business problem of managing global configurations across multiple nodes. The Lasp evaluation will evaluate scalability and practicality of Lasp as an application framework both in the data center and edge cases.

The following table capture the important parameters for the experiments.

Experiment	Partner(s)	Backend	No. of Nodes	Peak User Load
Ad Counter	Rovio	Riak 2.0	18 Riak nodes in 2 DC	8000 requests/sec
Configuration Management	ESL	Antidote	5-120 nodes in 1-2 DC	NA
FMK	KL, INRIA, Trifork, NOVA	Antidote	100-1000 nodes in 2-10 DC	To the breaking point
Lasp	UCL	Riak core	1,000-10,000	NA

Contractors contributing to the Deliverable

The following contractors contributed to the deliverables:

Rovio Vivek Balakrishnan.

ESL Viktória Fördős, Csaba Hoch, Diana Corbacho.

INRIA Alejandro Tomsic, Santiago Alvarez, Taylor Crain, Marc Shapiro.

UNL Nuno Preguica, Valter Balegas, Sergio Duarte.

UCL Peter Van Roy, Chris Meiklejohn.

UNIKL Annette Bieniusa, Deepthi Akkoorath.

Basho Torben Hoffmann.

2 Large-Scale Evaluation inspired by the FMK application

The aim of the experiment is to evaluate the availability and scalability of the Antidote CRDT platform. In particular it will be investigated if the stronger guarantees such as atomic transactions and invariant preservation are of practical use and without bottlenecks.

We will simulate the FMK use case from WP1 in a controlled environment for evaluation. The FMK system manages medicine prescriptions for patients in Denmark.

Trifork has identified a need for atomicity on different operations. This requirement mostly arises from the need to atomically update information in more than one object and from the use of non-normalized data. To address this requirement, we will use Antidote HATs (Highly Available Transactions) and evaluate its use in a real-world scenario. Each operation in the application is implemented as a transaction using appropriate CRDTs. Each of these operations consists of multiple read and/or update operations involving one or more entities in the FMK system.

In FMK, it is paramount to maintain availability even it is is impossible to access the most recent version of the data. In this context, it is important to have information on the potential divergence of the accessed information. To provide such information, we will rely on mechanisms for providing divergence information developed in the context of the project.

The Antidote FMK will be developed and evaluated incrementally, in the following steps:

1. Mimic subset of existing system with eventual consistency.
2. Use transactions and invariants preservation mechanisms.
3. Investigate partial replication.

4. Evaluate information on divergence.

Using Antidote for implementing a subset of the real FMK system, i.e. without support for advanced features such as transactions, will give us feedback on how well Antidote performs and help us weed out problems. This will also serve as a baseline to measure the overhead introduced by the additional guarantees provided by Antidote.

Once the base functionality is in place, we will look at maintaining consistency on operations that access multiple data items. This addresses a problem that current implementation of Riak cannot address (and requires special handling in the code of applications, leading to a lot of coding). We will use both the Antidote support for transactions and the mechanisms to maintain invariants.

We will use the partial replication as a principled approach to maintain a subset of the data in different nodes of the system: hospitals, medical facilities and pharmacies.

In a real deployment, due to costs (and problems that arise from increasing the replication factor), it is not possible to maintain a copy of all the patients' data in the hospitals. However, it is important for an hospital to have a copy of the data of patients that are being treated in that hospital. Otherwise, there might be ongoing treatments that the hospital staff is unaware of. Some treatments could cause the emergency treatment to become lethal.

We will use partial replication to have, in a given hospital or medical facility, a copy of data for patients being treated in the hospital and for the patients that live close to the hospital.

As for pharmacies, each pharmacy will get information on the prescriptions assigned to the pharmacy.

Finally, we plan to evaluate the mechanisms for providing information on data divergence. We are still studying which mechanisms are more appropriate in this case, but we expect to evaluate, at least, a mechanism for providing information on data staleness.

2.1 Large-Scale Evaluation inspired by the FMK application

In the FMK experiment we will try to simulate an environment similar to the existing FMK deployment. As such, we will simulate both the backend of the system, consisting in a number of data centers, and the frontend consisting in hospitals and pharmacies. The amount of records in the system will be defined by the sum of the number of patients, doctors, and other medical staff. Operations consist of transactions reading and updating records. The sytem will be evaluated by measuring both performance and scalability as well as its tolerance to faults and reconfiguration.

For performance we will measure the throughput T (the number of transactions executed per second) and latency of transactions (L) as the load increases. To generate the workload, we will use statistics provided by Trifork to simulate a realistic ratio between different operations. Besides the experiments that simulate realistic workloads, we will also create a series of benchmarks to measure how performance is influenced by a number of parameters (including number of operations in a transactions, varying ratio of reads and writes). Furthermore, given that availability is essential we will measure the impact of various types of failure scenarios (including network partitions, node failures and periods of high load) on performance, availability, and correctness.

We will measure the scalability of the system, by evaluating how performance (throughput and latency) vary with an increasing number of nodes, both at the backend and in the peripheral entities (hospitals, medical facilities and pharmacies). For this the number of data centers will range from 2 to 5. Each data center consists in a number of nodes that store data, with data being partitioned among the nodes. We will use from 5 to 40 nodes in each data center. The number of frontend nodes that simulate hospitals, medical facilities and pharmacies will also be varied from 10 up-to hundreds. Each hospital will be simulated by 2 to 3 nodes, holding a partial replica of the data. Medical facilities and pharmacies will be simulated by 1 to 2 nodes, also holding a partial replica of the data.

Additionally, we will increase the size of the database, increasing the number of patients, hospitals, pharmacies and medical facilities. We will use ratios between these entities, provided by Trifork, to guarantee a realistic evaluation.

Hardware All experiments will be performed on the public Grid'5000 infrastructure. Given that Grid'5000 consists of nine sites located in different cities across France, with each site containing from approximately 20 to 300 nodes, we believe it is an appropriate platform to execute the FMK experiment. Also, as the FMK application maintains information of a single country, simulating it in an infrastructure located in a single country mimics the real deployment more closely than using some international cloud provider.

Within the Grid'5000 infrastructure each of the 9 sites are divided into several clusters, with the compute, storage and network hardware being homogeneous per cluster. Thus, each of the 2 to 5 DCs will be run at different sites, each within a single cluster. Separate DCs will be running similar, but not equal hardware using nodes with 2 Intel CPUs each with 4 cores and 10 gigabit network connections within the cluster.

The medical facilities, pharmacies, and hospital nodes will be run across all nine sites of Grid'5000 in separate clusters from the DCs. This means a wide variance in hardware and location as we might expect in a realistic environment.

Risks Through through Grid'5000 we have the advantage of not needing to pay for using the infrastructure, we also increase the risks of having difficulties when compared to using commercial infrastructures. The first risk is that the infrastructure is shared with many other researchers, making it difficult to reserve large portions of the nodes for long periods of time. The second is that each site consists of different hardware that is managed by different groups, sometimes meaning using different rules for running the experiment at different sites. The third is that the infrastructure itself is experimental meaning frequent maintenance, possible inconsistent performance variations, and reservation and node errors. This said, at present we are able to run basic experiments across several sites with close to one hundred nodes, but running a realistic FMK experiment should be much more challenging.

2.2 Tools

Basho Bench is a benchmarking tool to conduct repeatable performance and stress test. The pluggable driver interface of Basho Bench allows it to be used as clients of FMK application. Thus measurements could be taken at the application level operations as well as for micro benchmarks. This will be initially used for quick prototyping the experimental infrastructure.

MegaLoad is a scalable load testing tool that provides automatic deployment on cloud environments. At the later stages of evaluation MegaLoad will be used for deploying large number of nodes and clients to conduct performance and stress test.

3 Rovio: Ad Counter

The user experience of a game player can be improved by decreasing the round-trip latency of user requests to the game servers. In the current "free to play" ad-oriented games, the above statement is also valid for the ad content that is served to the user. The overall round trip latency can be reduced by spreading the user load across several geographically-separated data centers (DC) and serving the user from the closest data center. As a part of this process, it is important that the datastore is also replicated in multiple datacenter (geo-replication).

Rovio's Ad Service keeps track of impressions and clicks for ads per campaign/ad/country. Typically, these counts have some upper bound after which the ad should not be shown anymore. The Ad Service runs on multiple service nodes in a single DC and each of those nodes has its own document for the impression and click counters in Riak, in order to avoid write conflicts.

Each campaign/ad/country info that tracks the impression (how many times the ad was shown to user) and click (how many times the user clicked on the ad) counters are stored as a domain object. Impressions and Clicks are standards KPI measures reported for the Advertisers in the Ad industry. The most common acronyms used in these measures are CVR (conversion rate, impressions driving visits/install) and CTR (click-through rate, click per impression)

These counters are implemented as a simplified version of Riak's counter CRDTs. In the massively large-scale experiment that is the aim of WP 5, Rovio will compare and validate geo-replication in CRDTs to the simple single-DC CRDT in the context of its Ad Service.

3.1 Hypothesis

The aim of the experiment will be to validate the following hypothesis:

> Geo-replicated CRDTs have as least as good performance as that of single DC solutions, while honouring the service restrictions and being easier to program correctly, especially when scaling is required.

Service restrictions for an ad campaign define how the ad is to be displayed. E.g., a campaign may have certain rules associated with it, such as serve only in US and Finland. The rules may also have a limit associated with them, such as serve until 50 impressions in US and 100 impression in Finland and 500 impressions globally. The Ad service serves the ad to the user based on such restrictions.

This experiment will validate that the Riak CRDTs, with geo-replication, improve the user experience without significant degradation in performance, for a distributed large scale service with massive user load.

3.2 Experiment Description

In the following, impact relates to:

- Geo-replication should not introduce additional service latency.
- Ad service should serve ad only till the limit mentioned in above question is reached. (campaign performance).

The experiment will compare the impact of geo-replicated CRDT on Rovio's ad serving logic to the existing simple intra-DC CRDT. We will measure this impact on overall ad content service rather than on the datastore. Observing the impact on ads content served from the Ad Service node is a good measure on the latency experienced by the game user. Further, it will also validate that the upper bound on ad traffic is respected irrespective of the delay introduced by inter-DC replication.

The legacy single-DC CRDT experiment will be the control experiment for comparing the impact of geo-replicated CRDTs. Both experiments will target a peak user load of 8 000 requests/second (as observed in production) against a production-like data store. The data will not be a live production replica, but it will be a production snapshot from an earlier date; the traffic is not live traffic, but either is synthetic or is replayed from anonymised traces of production traffic.

Both the geo-replicated CRDT and single-DC CRDT tests will use the same production snapshot and traces. Both tests will be repeated multiple times on varying hours of the day to cancel the bias introduced due to the computational load of the data centers.

3.3 Experiment Design

3.3.1 Variables

The experiments will measure the number of ads served (N) and the service latency (L) as a function of the independent variables user load (U) and experiment duration (T).

The simple single-DC CRDT experiment will measure N_{crdt} (the numbers of ads served using CRDTs) and L_{crdt} (the service latency measured when using CRDTs). Likewise, the geo-replicated experiment will measure $N_{geo\text{-}crdt}$ and $L_{geo\text{-}crdt}$. The geo-replicated experiment allows for a small deviation in the upper bound for the number of ads served ($\Delta_{geo\text{-}crdt}$), which will be capped at 10% of N_{crdt}.

3.3.2 Architecture

The planned experiment is not a live production large-scale test. Instead, the user load will be obtained from a load generator and the experiment will be tested in a cloud environment separate from Rovio's production cloud. The following reasons explain the above decision:

- Rovio cannot conduct the experiment against its services in production, as this would degrade the game user experience.
- A live experiment would require to update the client application; as users may choose to ignore the update, the numbers would not faithfully represent reality.

The user may choose to ignore the game update. On the other hand, with a load generator the we can vary the user load to the experiment which is beneficial. The client application here actually refers to the game, not the Ad service.

The experiments will use Riak 2.0 instead of SyncFree's research platform Antidote as it is not production ready. Also, as Antidote is based on `riak-core`, and its CRDTs are compatible with Riak 2.0, we believe that the results obtained with Riak 2.0, are likely to apply to Antidote also, and will highlight the power of the CRDT concept in general.

The set-up of the test environment is sketched in Figure 1. Rovio will set up two identical cloud environments, separate from its production cloud, in Amazon AWS locations, Ireland and US-East. The environment will use Riak 2.0, which includes support for CRDTs. We will establish a VPN tunnel across the Amazon environments for geo-replication. Each of these environments will host a stripped-down Ad Service, that is linked only to the Ad Livestats service. The Ad Service is the user-facing endpoint that serves ads. The Livestats service is the book-keeping service for the Ad Service, and maintains ad tracking data.

The user load will be generated from an in-house load generator tool called PerformancePig (see below). These instances will be run from one of Rovio's staging environment separate from the experiment setup. We will leverage on the existing

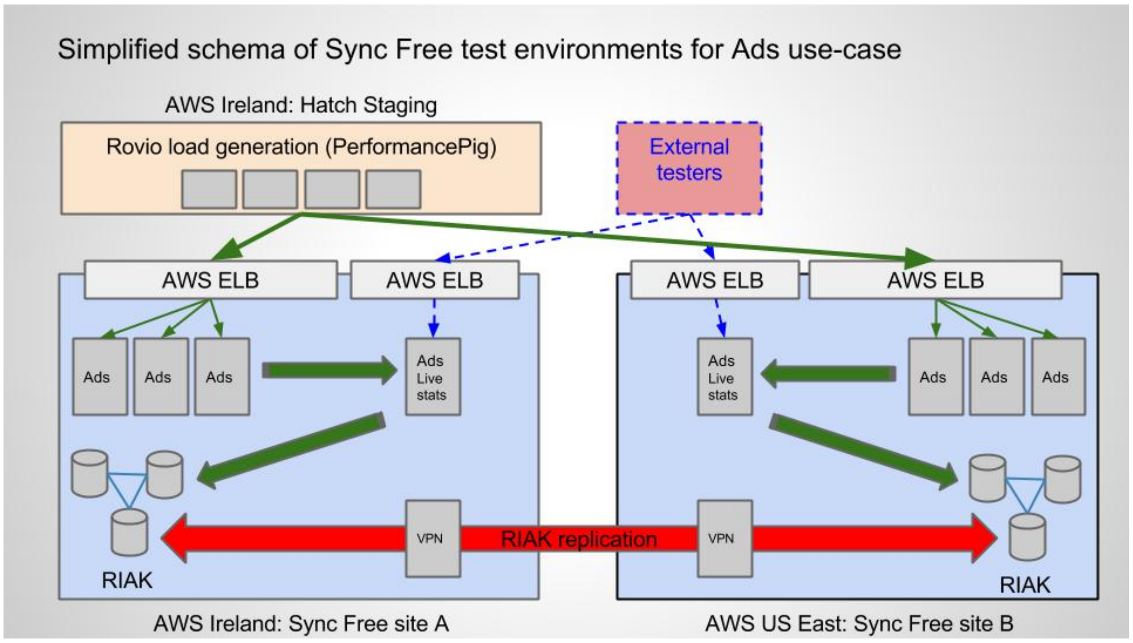

Simplified schema of Sync Free test environments for Ads use-case
AWS Ireland: Hatch Staging
Rovio load generation (PerformancePig)
External testers
AWS ELB
AWS ELB
AWS ELB
AWS ELB
Ads
Ads
Ads
Ads Live stats
Ads Live stats
Ads
Ads
Ads
VPN
RIAK replication
VPN
RIAK
RIAK
AWS Ireland: Sync Free site A
AWS US East: Sync Free site B

Figure 1: Test environment for Ad use-case.

PerformancePig instances in Rovios staging environment. This way Rovio doesnt need to deploy PerformancePig instances in the newer AWS environment. This was decision taken to reduce the time to actual experimentation.

The experiment setup will have an Amazon Elastic Load Balancer (ELB) to evenly distribute the user load across the Ad Service.

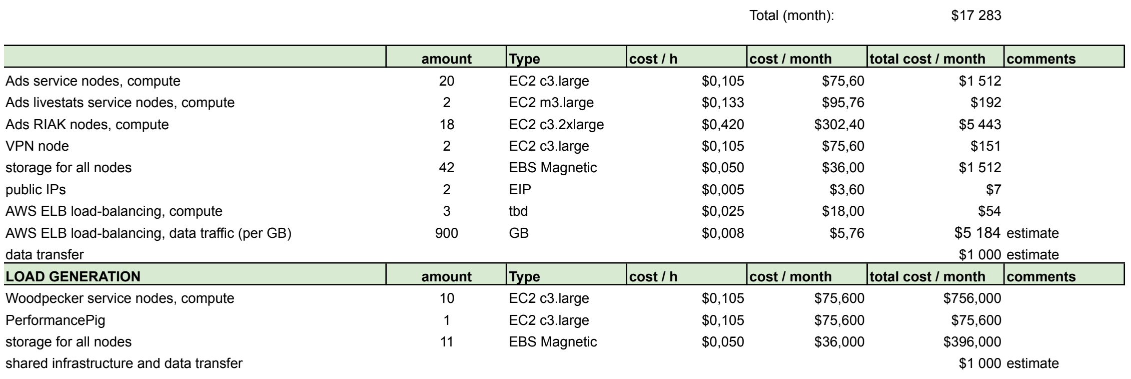
3.3.3 Measurement tools

JMonitor is a distributed resource monitoring tool written in Java that is used by all Rovio services to log and monitor service access. This tool logs the performance timing for all important service calls and presents them at varying levels of granularity. The tool is also capable of monitoring a distributed counter. The experiment will use this tool in Ad Service to measure the number of ads served (N) and service latency (L) values.

PerformancePig is Rovio's inhouse load generator tool to run and visualize performance test data. Together the with Woodpecker library, PerformancePig can be used to deploy test nodes in EC2 instances to simulate peak loads. The tool has means to calculate the average latency, latency range, transactions per second and user count for every service call of the test suite. The experiment will use this tool to generate user load for the experiments.

Basho Bench is a benchmarking tool from Basho for conducting stress and performance tests and to produce performance graphs. It is exposed as a pluggable driver that can be extended using Erlang. Basho Bench will primarily be used for quick prototyping, while stabilizing the cloud environment and geo-replication across VPN tunnel.

PerformancePig is an in-house load generator that is used for testing Rovio



Total (month): $17 283

	amount	Type	cost / h	cost / month	total cost / month	comments
Ads service nodes, compute	20	EC2 c3.large	$0,105	$75,60	$1 512	
Ads livestats service nodes, compute	2	EC2 m3.large	$0,133	$95,76	$192	
Ads RIAK nodes, compute	18	EC2 c3.2xlarge	$0,420	$302,40	$5 443	
VPN node	2	EC2 c3.large	$0,105	$75,60	$151	
storage for all nodes	42	EBS Magnetic	$0,050	$36,00	$1 512	
public IPs	2	EIP	$0,005	$3,60	$7	
AWS ELB load-balancing, compute	3	tbd	$0,025	$18,00	$54	
AWS ELB load-balancing, data traffic (per GB)	900	GB	$0,008	$5,76	$5 184	estimate
data transfer					$1 000	estimate
LOAD GENERATION	**amount**	**Type**	**cost / h**	**cost / month**	**total cost / month**	**comments**
Woodpecker service nodes, compute	10	EC2 c3.large	$0,105	$75,600	$756,000	
PerformancePig	1	EC2 c3.large	$0,105	$75,600	$75,600	
storage for all nodes	11	EBS Magnetic	$0,050	$36,000	$396,000	
shared infrastructure and data transfer					$1 000	estimate

Figure 2: Budget calculation for Rovio's Add counter.

services. Basho bench on the other-hand, load testing tool to specific to Riak. We plan to use Basho bench for initial cloud setup to verify that the geo-replication works, et cetera, but not for actual tests. Furthermore, Basho bench is specifically designed for Riak performance test and it cannot be used for Rovio services.

3.4 Budget calculation

The infrastructure budget calculation considering a peak load of 8 000 request/second is presented in the following table. All calculation are computed on a monthly basis considering the current Amazon prices.

3.5 Further evaluation

In addition to executing the here described environment, Rovio will expose APIs to its LiveStats node to the other Syncfree partners to run their own tests. These APIs will be separate from Rovio's internal API for obvious security reasons. The Ad Service node will not be accessible to other partners. The details of the tests to be conducted by other partners will explained in a separate document.

4 ESL: Configuration management

4.1 Brief overview of ESL's contributions

ESL joined to SyncFree consortium at M24 in order to strengthen the Erlang expertise, to ensure industrial standards, to help in tooling and to evaluate and exploit Antidote. Since M24 ESL reviewed the implementation of Antidote and gave lessons on Erlang best practices allowing the consortium to refine the current implementations.

ESL expects benefit from SyncFree project result in Wombat's commercial exploitation as a new feature will be developed as part of SyncFree activities. ESL will strongly contribute to WP5 fulfilling many objectives of T 5.2 and T 5.3 detailed as follows.

Objectives of T5.2 will be delivered as part of D5.2 deliverable by ESL:

- ESL will develop two approaches to global configuration management;
- ESL will compare the two approaches along several axes, e.g., performance, guarantees, code quality, or ease of development. Standardised metrics (such as ISO 9126) will be used to evaluate the complexity, reliability, and maintainability of the rebuilt code in comparison to the original application code. User satisfaction surveys will take into account the point of view of the application developers.

Objectives of T5.3 will be delivered as part of D5.3 deliverable by ESL:

- ESL will plan and design the evaluation of the approaches to configuration management including the rationale of its design parameters and ensuing resource costs;
- ESL will include scenarios with varying network topologies, and diverse failure modes to its evaluation plan;
- ESL will reveal the value for money of the expected results, and the business case.

Objectives of T5.3 will be delivered as part of D5.4 deliverable by ESL:

- ESL will perform the evaluation and assess its result. The assessment will include the observations of the behaviour of Antidote over large numbers of replicas, varying network topologies, and diverse failure modes.

WombatOAM is a product Erlang Solutions has been working on for three years, originally the result of a STREP FP7 funded project. ESL has over a dozen deployments world wide, and is using it internally as the foundation to all the support and development work ESL does for customers. ESL is expanding WombatOAM by placing all generic operations and maintenance functionality which can be reused across projects and industry verticals. Generic functionality which exists to date include notifications (logs), metrics and alarms. The next two major features include configuration management and software upgrade during runtime, without taking down the system.

4.2 Problem description

Configuration management in large scale, distributed systems where network connectivity is unreliable and clusters span across data centers is today often limited to static files created and deployed at startup. Obviously, there are some existing approaches, but all of them are limited. Namely, they can only do offline and require a soft restart of the system. They rely on strong consistency, which cannot be both consistent and available because the distributed system is prone to failures. Therefore, the industry cannot rely on them. Instead, changing configuration after the initial deployment is often done in the business logic of the system, forcing developers to write their own layers to guarantee consistency.

Even though the developers do their best, there are scenarios that cannot be avoided using the existing approaches. Consider the scenario when the propagation is not done completely due to network issues. Thus, some nodes with the old configuration remain in the cluster, whilst the other nodes use the new configuration. This situation results in a misconfigured system, which is hard to correct and can easily cause service degradations.

How can an operation team change the configuration parameters in a cluster consisting of thousands of nodes in different geographical regions without restarting the system? How can the consistency across these nodes be guaranteed, even when the network is unreliable? And how can the propagation be monitored and inconsistencies among nodes be detected? The correctness of these operations is critical, as human mistakes and infrastructure errors can result in system outages in the best of cases and incorrect behaviour in the worse. Also, it is time critical, because until the propagation of the configuration data is completed, the system will be in an inconsistent state. An automated approach which both propagates the information and monitors consistency is required. ESL believes CRDTs and causal consistency is the right approach to solve the problem.

Erlang/OTP has node configuration on an application basis which can be changed during runtime. They are called *application environment variables*. Today, any changes in these environment variables needs to be done and persisted in the business logic of the system. That means the problem is (if at all), solved in every system, reinventing the wheel. In some cases, the changes are done manually directly in the Erlang shell and by editing the configuration files.

4.3 Requirements

It should be possible to change global configuration parameters in clusters consisting of thousands of nodes running in clusters spread across different geographical regions. These changes could be initiated centrally through the Wombat Console. They should propagate across all nodes as quickly as possible, and be able to withstand packet loss and network partitions.

It should be possible to automatically detect any inconsistencies among the nodes. If the values of a certain global configuration parameter are different in clusters then it is an anomaly, and the inconsistency should be immediately reported to users. This monitoring functionality could be the part of WombatOAM, which should raise an alarm automatically when different values are set to any global configuration or a global configuration is missing from any node of the cluster. The alarm should be cleared automatically when the values of all the global configurations are the same and also set on all nodes.

4.4 Business Case, Necessity

There are many OAM tools for the mainstream programming languages and generic monitoring and deployment tools. None of them, however, are Erlang specific and take advantage of its programming model. This is a real issue as Erlang drives the telecom industry (e.g. Ericsson), large betting portals (e.g. Bet365, WIlliam Hill), market leading payment providers (Klarna), gaming engines (MachineZone) and online ad services (AdTech, OpenX, AddRoll). ESL wants to satisfy the customers needs by introducing the configuration management feature in its existing OAM tool, called WombatOAM. It shall enable safely changing parameters in large-scale clusters to avoid outages. Configuration management is one of the required features, which, once it works, will lead the way to automated dynamic software upgrade and orchestration.

80% of Erlang Solutions revenue comes from professional services. ESL's own products are critical in changing the company from a times and material one to recurring revenue based model. WombatOAM allows ESL to remain focused on its core business whilst making this transition.

ESL expects the ability to manage configuration management in an innovative way to allow ESL to double its sales of Wombat to about 400,000 GBP in the 12 months which follow the deployment of the functionality. ESL expects this as an enabler to initiate conversations with enterprise customers who are running thousands of nodes in production. ESL's largest customer currently runs 200 nodes.

4.5 Assessment From A Real Customerś Perspective

Illustrating the prominence of the issue, ESL presents the configuration updating process of one of its customers. ESL emphasises that other customers face similar problems. One of ESLs customers currently change their application environment variables during software upgrades using word documents. They have to do it manually on every node by entering the shell and calling `application:set_env(...)`. The engineers performing this task often have no Erlang expertise. To ensure the values persist after a restart, they also need to change a configuration file, manually, through cut and paste from a word document. Their process is tedious, error prone, and has caused outages.

Another customer, one of the major telcos in their country of operation, engaged ESL to write a system which would gather configuration parameters across systems they had deployed in different data centers, raising alarms when parameters differed. When running the system the first time, ESL discovered that half of their equipment was not configured correctly and was unable to send alarms in case of malfunction. They could have lost half of their processing capacity and only discovered it when customers would have dialed in and complained of service malfunction. This happened despite their configuration procedures being fully automated.

4.6 Evaluation

As described above, ESL wants to develop an approach to global configuration management that ensures the causal consistency of these values. The approach will be implemented by the ESL developers, and will be built on the top of the current architecture[1], which is a centralised, tree based approach. After that, the approach will be rebuilt by the ESL developers in a decentralised manner that relies on Antidote. The rebuilt approach will use Antidote to store data and also commands, therefore to instruct the plugins running on the managed nodes indirectly. A report on the two approaches will be part of the D5.2 deliverable, whilst the realisation of the approaches will be presented as part of D5.2.1.

[1]RELEASE Project. Deliverable D4.5: Scalable Infrastructure Performance Evaluation report, March 2015.

The decentralised approach will be evaluated to observe the behaviour of Antidote over large numbers of replicas, varying network topologies, and diverse failure modes. As part of the evaluation ESL wants to investigate

- how the configuration management can exploit the CRDT concept;
- how Antidote can accommodate itself to requests arriving in bursts;
- the scalability capabilities of Antidote in terms of size of replica; and
- resilience and network partitions.

Last but not least the experiments ESL wants to perform will reveal business critical information essential for ESL's customers. If customers would prefer the decentralised approach relying on Antidote in order to minimise propagation failures due to network issues, they will have extra costs. Namely, the cost of hosting the Antidote nodes. Therefore, the increase in the overall reliability in the function of the Antidote nodes needs to be known to select the configuration best fitting to their needs. For instance, an increase of five per cent isn't worth the extra cost of hosting more Antidote nodes for some customers, whilst it is essential to ensure the normal operation of the others' business. Determining this information is the real value of the experiments, since it leads to commercial exploitation. Therefore, it appropriately justifies the expenses.

4.7 Hypothesis

The aim of the experiments will be to validate the following hypotheses.

> The prominence of the benefits provided by Antidote in terms of resilience and network partitions will be increased in the function of the number of the Antidote nodes.

These experiments will validate that the success of the propagation driven by Antidote can be entrusted even if unexpected network issues occur, whilst the centralised approach will not be capable of reaching the goal – leaving the cluster in an inconsistent state.

> Antidote is capable of handling requests arriving in bursts.

This experiments will validate that Antidote will not become unresponsive if the level of load greatly increases.

Besides that, the experiments will provide data allowing ESL to determine how much increase in the overall reliability can be expected by increasing the number of Antidote nodes.

4.8 Experiment Description

ESL will carry out the most common system maintenance tasks using WombatOAM, which will be presented as case studies.

First, ESL will focus on the new capabilities of WombatOAM from the operation perspective. To this end, ESL will perform experiments checking the maintenance capabilities: How efficiently can the frequent maintenance operations be performed using WombatOAM on Riak clusters consisting of hundreds of nodes? For instance, consider changing the value of a global configuration parameter cluster-wide. This may be limited to the following changes: modifying the limit of concurrent hand-offs, changing the port of the Protocol Buffer or activating the consensus subsystem used for strongly consistent Riak operations.

Second, ESL will evaluate WombatOAM focusing on its troubleshooting capabilities on a cluster built up from hundreds of nodes. After adding the cluster consisting of nodes with different values set as a global configuration parameter, WombatOAM shall be able to detect the anomalies. Once anomalies are detected, alarms will be raised to warn the users.

Using the case studies ESL will

- judge whether WombatOAM's configuration management is a useful feature providing valuable support to operation teams;
- determine the recommended number of Antidote nodes in function of the managed nodes' count;
- observe how the replication between Data Centers affect the usability of the configuration management feature;
- observe how Antidote nodes can repair the diverged data after network issues;
- observe how network issues effect Wombat and the plugins (i.e. the ongoing changes will be completed, the active alarms will describe the current status of the managed nodes correctly).

ESL will also examine failure scenarios where the following two components cannot communicate with each other due to network issues:

- WombatOAM with the Antidote nodes;
- WombatOAM with the plugins running on the managed nodes;
- Antidote nodes with the plugins running on the managed nodes;
- Antidote nodes with the rest of their cluster;
- Antidote nodes with the other Data Center.

The assessment of the experiments and the lessons learnt will be reported in Deliverable D5.4.

4.9 Experiment Design

4.9.1 Variables

For a certain customer the following parameters are given and fixed.

- The number of managed nodes (M). M will vary between 50 and 400 to simulate different customers.
- The number of configuration parameters per managed node (C). Generally speaking, C is between 200 and 400. For the sake of simplicity we will assume that $C = 300$.
- The number of fallback nodes (F) to which plugins running on a certain managed node can connect. This parameter has a positive impact on the overall reliability of the system, however, it also increases the load put on the Antidote nodes. It will be defined by customers based on their needs. F will vary between 1 and 3.
- The number of Data Centers (D) the customer hosts to provide its business worldwide. Generally speaking, D is between 1 and 5, however, ESL will experiment with the most cases: $D = 1$ and $D = 2$ cases.

Based on the functional specification of the decentralised approach the following estimates related to the Antidote clusters can be made.

- $M * C$ small objects stored in total;
- $M * C$ PUT requests in one burst;
- $M * F$ GET requests periodically,
- M PUT requests coming in bursts each of which invoking M GET requests.

As the estimates forecast, ESL's use case will use Antidote differently. Instead of continuous moderate load, long periods with quasi zero load alternating with short periods with large load can be expected. Therefore, the capacity of Antidote specifically for ESL's use case should be determined. The number of managed nodes connecting to the same Antidote node (M_L) can describe the capacity. By the experiments, we want to determine the maximal M_L that enables fast operations to minimise the resource cost of operating the Antidote nodes.

With M, C, F and M_L, the number of Antidote nodes (A) can be determined.

$$A = \left\lceil \frac{M * F * {}^{C}/_{300}}{M_L} \right\rceil .$$

First, M_L will be determined based on a very few values set to M. Based on ESL's Riak expertise, ESL expects that M_L will be around 10. Next, several experiments will be carried out. Each experiment will use different values set to the variables, which will be performed multiple times. A selected set of experiments will be repeated by creating network issues described in the failure scenarios.

The experiments will measure the total time (T) required to a) change the value of a global configuration parameter on all the managed nodes and b) notice inconsistencies and then raise alarms in the function of M, F, and D.

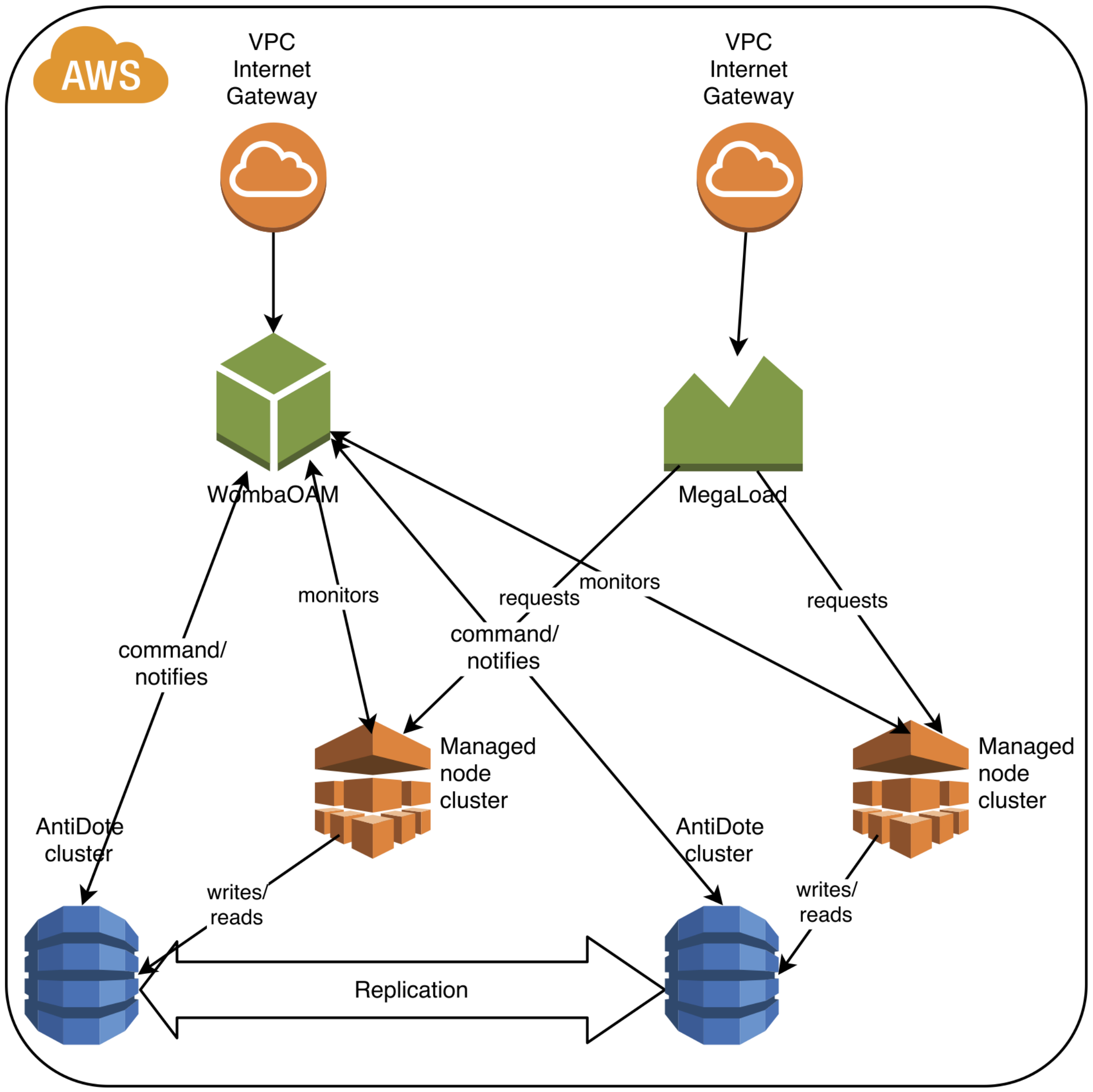


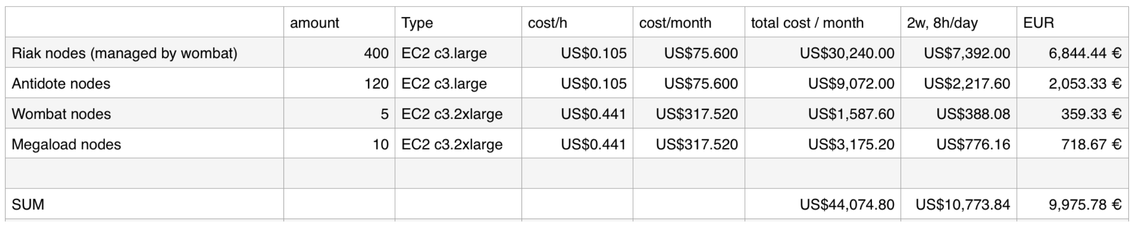
Figure 3: Test environment for the Configuration Management use case.

4.9.2 Architecture

The planned experiment is not a live production large-scale test, as ESL believes that only well-tested, production ready software should be installed by customers. Neither Antidote nor the decentralised approach of the configuration management feature is production ready.

Riak is a popular key-value store used by companies building on the top of the Erlang VM. Therefore, WombatOAM will monitor and configure Riak nodes during the experiments. Considering production systems, the Riak nodes are never idle, so a moderate number of requests will be generated by MegaLoad for the Riak nodes.

The set-up of the test environment is sketched in Figure 3.



	amount	Type	cost/h	cost/month	total cost / month	2w, 8h/day	EUR
Riak nodes (managed by wombat)	400	EC2 c3.large	US$0.105	US$75.600	US$30,240.00	US$7,392.00	6,844.44 €
Antidote nodes	120	EC2 c3.large	US$0.105	US$75.600	US$9,072.00	US$2,217.60	2,053.33 €
Wombat nodes	5	EC2 c3.2xlarge	US$0.441	US$317.520	US$1,587.60	US$388.08	359.33 €
Megaload nodes	10	EC2 c3.2xlarge	US$0.441	US$317.520	US$3,175.20	US$776.16	718.67 €
SUM					US$44,074.80	US$10,773.84	9,975.78 €

Figure 4: Budget calculation for ESL's Configuration management.

4.9.3 Tools

WombatOAM WombatOAM is an operations and maintenance tool for Erlang systems that has been developed by ESL. It automatically gathers and stores metrics, notifications and alarms from the monitored Erlang nodes. That is achieved by starting non-intrusive agents at the managed nodes.

WombatOAM also played a prominent role in the context of the EU-project RELEASE. There, the main task of WombatOAM was to provide the scalable infrastructure for deploying thousands of Erlang nodes. WombatOAM was aimed at providing a broker layer capable of creating, managing and dynamically scaling heterogeneous clusters consisting of thousands of Erlang nodes. The challenge was tackled by introducing the Meta Wombat approach. Meta Wombat is a scalable, distributed and fault-tolerant approach relying on a tree built from independent WombatOAMs, which are called as Worker Wombats. WombatOAM is a product of Erlang Solutions already having satisfied end-users such as EE, Orange, Cisco, and SQOR.

MegaLoad MegaLoad is a scalable load testing tool that provides automatic deployment on cloud environments or physical hardware. It allows to simulate a massive amount of load to stress test a system. The powerful real-time measurement system provides all the information that is needed to monitor system tests through a graphical user interface. MegaLoad has proven to be ideal for online business, SaaS and telecoms companies.

4.10 Budget calculation

As WombatOAM and MegaLoad are proprietorial software, deploying them to public computational infrastructures' resources (e.g., Grid5000, EGI or PRACE) is not preferred. Instead, ESL will use its private infrastructure to perform experiments with moderate hardware requirements to reduce the expenses, whilst the other experiments will be performed in the AWS environment.

All calculations are computed considering the current Amazon prices[2]. Figure 4 shows the calculated budget.

[2]https://aws.amazon.com/ec2/pricing/

5 Large-scale evaluation of Lasp

Lasp and Selective Hearing are major results of SyncFree's second year. During SyncFree's third year, we will continue to develop both Lasp and Selective Hearing, to implement industrial application scenarios and to improve the performance, scalability, and functionality. In this context, we will do large-scale evaluation of both Lasp and Selective Hearing at large scales limited by our budget of EUR 10,000 (ten thousand Euros). The evaluation will be managed by UCL. The budget was transferred to UCL in the beginning of SyncFree's third year from the evaluation budget that was originally attributed to Rovio. Our cost estimates show that this budget should be sufficient for ramping up to from 1,000 to 10,000 nodes on a commercial infrastructure such as Amazon. The maximum scale depends on how many problems we encounter during the ramping up process.

Section 5.1 briefly recapitulates Lasp and Selective Hearing. Section 5.2 explains the four evaluation scenarios we propose to implement. Section 5.3 explains how we will manage the ramp-up in the most economical manner possible, both in implementation effort and budget.

5.1 Lasp and Selective Hearing

Lasp and Selective Hearing are explained in detail in Deliverable D4.2. This section briefly recapitulates their properties that are relevant for evaluation. Lasp is a programming model based on using CRDTs as primitive data structures and composing them using operations derived from functional programming and relational (database) programming. Our first implementation of Lasp was on a data center, but our current work is focused on Selective Hearing, which is an execution model of Lasp that targets an edge network. An edge network consists of a large set of heterogeneous, loosely coupled physical nodes, of all sizes from data centers, points of presence, down to individual computers, mobile devices and IoT devices. Edge computing is an area for which Lasp seems to be particularly appropriate because there exists currently no general-purpose programming framework for edge networks.

Selective Hearing uses the Plumtree epidemic broadcast algorithm to ensure that the Lasp computation progresses despite the typical issues of an edge network, i.e., intermittent connectivity, node failures, and network dynamicity. Plumtree successfully combines the resilience of gossip with the efficiency of a spanning tree. Plumtree is used in industry, for example it is used in Riak for improving robustness. Lasp and Plumtree are a good match because Lasp tolerates weak message ordering and intermittent connectivity, which are the exact properties of Plumtree.

To distinguish an edge implementation of Lasp from an implementation that uses data center nodes, we will call the former "Gossip Lasp" and the latter "Datacenter Lasp". This distinction is actually a spectrum, with pure Gossip Lasp at one extreme and pure Datacenter Lasp at the other extreme. In practice, the system is tiered with nodes that have varying degrees of computation power and storage. In the evaluations that we plan to do, we will take this spectrum into account as one aspect to be evaluated.

5.2 Lasp evaluation scenarios

We plan the following four evaluation scenarios:

- Evaluation of Gossip Lasp as a scalable framework for edge applications. We will gradually increase the scale of the system with a simulated edge computing workload. This requires optimization of Selective Hearing and adaptation of the Plumtree broadcast algorithm to improve scalability. The purpose is to give a proof of concept that Lasp is a practical platform for edge applications.

- Evaluation of Lasp (both Gossip and Datacenter Lasp) as a scalable application framework. This will evaluate the Ad Counter scenario with millions of clients, using Datacenter Lasp for scalabiity. The best architecture of this scenario is still to be determined, i.e., what combination of Gossip Lasp and Datacenter Lasp gives best results. This evaluation requires optimization of Datacenter nodes used in a Lasp implementation. The purpose is to show that Lasp is a scalable platform that can take advantage of (some) datacenter centralization to increase its scalability.

- Evaluation of Gossip Lasp as a practical computation framework. We will evaluate the expressiveness of Lasp in a large edge computing scenario. We will implement a computation-intensive workload to be executed on an edge network using Selective Hearing. The purpose is to show the viability of Gossip Lasp as a computation framework, even on an edge network with its dynamic membership and intermittent connectivity. This is important to demonstrate that centralized data centers are not the only approach for large-scale computation, since future networks will have a hierarchy of nodes of different strengths, from large datacenter nodes to numerous small leaf nodes. For scalability and latency reasons, it will be important to offload computations from the data center onto the edge network, which is what this evaluation will test.

- Evaluation of the expressiveness of Lasp language compared to traditional approaches. We will attempt to answer the question: is it possible to compare the same application written in three ways: (1) in Lasp, (2) using CRDTs but without Lasp, and (3) without CRDTs, using traditional distributed programming techniques? This will be done in concertation with the evaluations of (2) and (3) explained elsewhere in this D5.3 report. We remark that this fourth evaluation is the most speculative of all evaluations we intend to do: it depends on our progress in improving the expressiveness of Lasp.

At the moment, we consider the Ad Counter scenario as the one most likely to work out for these evaluations. However, we are also studying the Leaderboard and Wallet scenarios. We have implemented Leaderboard and it seems to be uninteresting: it uses a single CRDT without composition. Wallet is much more interesting because it has atomic transactions with CRDTs, and these transactions can involve not just money but also goods (like a network of B-to-B applications).

We emphasize that these evaluations assume progress in the implementation of Lasp and Selective Hearing, which is hard to predict given that Lasp did not even exist one year ago. In any case, Lasp development is moving very quickly, and any evaluations we do in the third year will be proof of concept only, because much more progress will be possible *after* the SyncFree project ends.

5.3 Scale and ramping up of the Lasp evaluation

Evaluation at large scale is a difficult endeavor. Given our limited resources in personnel and budget, we will simplify our implementation plan to the greatest degree. To maximize the relevance and realism of our results, we target a system with from 1,000 to 10,000 physical nodes as the final, largest step of our evaluation. We will use the following approach:

- *Same infrastructure for all scales.* To minimize the implementation effort, we will do all our evaluations on the same physical infrastructure, namely a commercial infrastructure such as Amazon. It is impractical to implement small-scale evaluations on a public infrastructure such as Grid5000 and use a commercial infrastructure only for the largest scale. This is not just because it increases the implementation effort, but also because new bottlenecks appear at each step, and changing the infrastructure will invalidate much of our reasoning on predicting and solving the "next" bottleneck. Because the ramp-up is exponential, it is only the final step of the ramping up that will consume the lions part of the budget. The decision to use only commercial infrastructure for all steps therefore will have little effect on the budget.

- *Judicious use of simulation, but not for the largest scale.* Before going to large numbers of physical nodes, we will if necessary take the step of using large numbers of simulated nodes. This will help us identify bottlenecks due to the Lasp implementation itself. The early, smaller evaluations will be done on small numbers of physical nodes and larger numbers of simulated nodes. However, it is clear that simulation cannot be our final stage: a real system differs significantly from a simulation. The only way to test true scalability is on large numbers of physical nodes. The bottlenecks of real hardware do not appear in a simulation unless the simulation is ridiculously detailed, and in that case it would use at least as many physical nodes as the real execution.

- *Identifying and solving bottlenecks is critical.* We will start with small number of nodes and ramp up in incremental factors. At each new scale, we expect new bottlenecks to appear that will require new design and implementation work. The final result that we will be able to achieve before the end of the SyncFree project depends on our ability to identify and solve these bottlenecks.

- *Stability of the large-scale infrastructure is critical.* Our practical experience shows that it is difficult to keep a steady state of hundreds of physical nodes working together online, nevermind our ultimate goal of 10,000 nodes. This is true on all infrastructures we have used, including commercial infrastructures such as an Amazon cloud. A significant part of our work will be needed to manage the instability of cloud infrastructures at this scale.

5.4 Final remark

As a final remark, we are in the process of contacting companies interested in using Lasp in different edge scenarios, including scenarios that use data center nodes as well as scenarios in Internet of Things. We are currently in contact with several companies that are working on edge networks in the area of Internet of Things. One of the goals of our large-scale evaluations is to convince them of the viability of our approach.