CHALMERS
UNIVERSITY OF TECHNOLOGY

Data Minimization in Distributed Applications for More Privacy

Master's thesis in Algorithms, Languages and Logic

JAKOB BOMAN

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Master's thesis 2016:NN

Data Minimization in Distributed Applications for More Privacy

JAKOB BOMAN

Department of Computer Science and Engineering
Division of Software Technology
Chalmers University of Technology
Gothenburg, Sweden 2016

Data Minimization in Distributed Applications for More Privacy
JAKOB BOMAN

© JAKOB BOMAN, 2016.

Supervisor: Thibaud Antignac, Software Engineering
Examiner: Wolfgang Ahrendt, Software Engineering

Master's Thesis 2016:NN
Department of Computer Science and Engineering
Division of Software Technology
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2016

Data Minimization in Distributed Applications for More Privacy
JAKOB BOMAN
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

The presence of connected devices in our environment is increasing. These devices form a network often called Internet of Things (or IoT for short), where everything from light-bulbs to thermostats can be controlled by an app or by another device. These services make a lot of that data available to the end user but also to malicious parties due to the devices leaking more data than intended or by bad design. This puts the end user at risk, violating its privacy and leaking sensitive data. One simple and obvious way to prevent leakages and misuses of personal data is to collect less of this data, a principle known as data minimization. However, this solution is rarely used in practice because of business models relying on personal data harvest on one hand and because of the difficulty to enforce it once it is defined what is actually needed to provide a service.

Keywords: *some keywords will be added here*

Acknowledgements

First of all I want to thank David Frisk for this outstanding LaTeX-template I used for my master thesis.
of course others will be thanked as well

Jakob Boman, Gothenburg, October 11, 2016

Contents

List of Figures

List of Tables

tables will show up here when added

1
Introduction

1.1 Motivation

1.2 Aim

In this thesis I investigated ways to improve privacy in a special kind of IoT (Internet of Things) devices known as Wireless Sensor Networks (WSN). WSN are networks of autonomous sensors and actuators. The goal to enhance privacy for this kind of devices will be addressed by relying on data minimization. This means the project sought to improve privacy in distributed networks by limiting the amount of personal data being processed.

1.3 Scope and Limitations

1.4 Thesis Structure

saving for later when the thesis has shaped up

1.5 Background

This section should cover some background information to give the reader some background knowledge to what the project has been about that is required knowledge before moving forward.

1.5.1 Wireless Sensor Network (WSN)

A Wireless Sensor Network is recent improvement from the traditional sensor networks, made possible by advances in micro-electro-mechanical systems (MEMS) technology making sensor nodes that are smaller, multifunction and cheaper in comparison to previous sensors. Traditional sensors have two ways of being deployed; 1) They were positioned far away from the actual *phenomenon* (e.g. something known by sense perception) which required large sensors using complex techniques to distinguish the targets from surrounding noise. 2) Several sensors were deployed that only performed sensing and their communication topology had to be carefully engineered and they transmitted time series of the data to the central nodes which performed the communication.Wireless Sensor Networks on the other is constructed by deploying a large number of sensor nodes close to the phenomenon and their position doesn't need to be engineered or predetermined.[1]

discuss other good things with WSNs but try to keep it relevant

1.5.2 Data Minimization

As defined by the EDPS (European Data Protection Supervisor); "The principle of "data minimization" means that a data controller should limit the collection of personal information to what is directly relevant and necessary to accomplish a specified purpose. They should also retain the data only for as long as is necessary to fulfill that purpose. In other words, data controllers should collect only the personal data they really need, and should keep it only for as long as they need it." [2]

discuss where and how the quote came to be, there's information on the link for that.

This covers two important aspects of data minimization, the first being that data should only be kept for as long as it is useful for an application and the second being that they should only collect "relevant" data. The latter is more interesting to the project, since the project's aim is to solve part of this problem.

2

Theory

This chapter provides an introduction into the theoretical elements used throughout the course of the project.

2.1 Formal Verification

The act of formal verification means to make use of mathematical techniques to make sure that a design upholds a defined functional correctness.[3]
This means, that if we assume we have the following: a model of a design, a description of the environment where the design is supposed to operate in and some properties we wish the design to uphold. With this information, one may want to construct some input sequences, that are in the allowed in domain of the environment, that would violate the properties stated. A common practice for finding such patterns today are random simulations or directed tests.[3]

include another citation?

Formal verification allows for an extended approach to this, as it allows both to search for input sequences that violates the properties but also allows to mathematically prove that the stated properties holds when no input sequences exist.[3]

also show a work flow?

2.2 Model Checking

A traditional approach to verifying concurrent systems is based on using extensive testing and simulation to find and eliminate unwanted occurrences from the system, but this way can easily miss crucial errors when the system you're testing has a large number of possible states[4]. An alternative technique that was developed in the 1980's by Clarke et al. is called *temporal logic model checking* or "Model Checking". Model Checking is a automated technique to verify finite state concurrent systems. By letting a tool verify that a model holds for certain properties. The process of applying Model Checking to a design is separated into several tasks; *modeling*, *specification* and *verification*.

> **Modeling:** First task is to translate a design into a format which is accepted by a model checking tool. This is either a compilation task or a task in abstracting certain aspects of the design to eliminate irrelevant or unimportant details, due to limitations on time an memory.
>
> **Specification:** Second task is to state which properties the design is supposed to have. This is usually done using in a logical formalism, commonly

in temporal logic, which can express assertions on a system evolving over time.

Verification: The final step is allowing the tool to verify the specification on the model. This will either be a positive result, meaning the model satisfies the properties, or a negative result where the properties aren't. A negative result can also be that the model's state space is too large to fit into a computer, which will require the model to be further abstracted to be verified.

2.2.1 State Space Explosion

mention the problem with state space explosion

will explain the problems with having a to precise model

2.2.2 Model Checking Workflow

show structure of model checking workflow

show a structure of a model checking workflow

2.3 Promela & SPIN

The model checking tool used for this project is called Simple Promela Interpreter (SPIN) and the language it accepts is called Promela, which is an acronym for Process Meta Language.

describe usages of SPIN

2.3.1 Operational Semantics of Promela

explain why this section is relevant

Definitions 7.1-7.5 defined in Spin reference manual (p.155-157) [5]

Definition 1. (Variable)
A *variable* is a tuple (*name*, *scope*, *domain*, *inival*, *curval*) where
name is an *identifier* that is unique within the given *scope*,
scope is either *global* or local to a specific *process*,
domain is a finite set of *integers*,
inival, the initial value, is an *integer* from the given *domain*, and
curval, the current value, is also an *integer* from the given *domain*.

Definition 2. (Message)
A *message* is an ordered set of *variables* (Def 1).

Definition 3. (Message Channel)
A *message channel* is a tuple (*ch_id*, *nslots*, *contents*) where
ch_id is a positive *integer* that uniquely identifies the channel,
nslots is an *integer*, and
contents is an ordered set of *messages*(Def 2) with maximum cardinality *nslots*.

Definition 4. (Process)
A *process* is a tuple (*pid*, *lvars*, *lstates*, *initial*, *curstate*, *trans*) where
pid is a positive *integer* that uniquely identifies the process,
lvars is a finite set of local *variables* (Def 1), each with a *scope*
lstates is a finite set of *integers*,
intial and *curstate* are elements of set *lstates*, and
trans is a finite set of *transitions*(Def 5) on lstates.

Definition 5. (Transition)
A *transition* in process *P* is defined by a tuple (*tr_id*, *source*, *target*, *cond*, *effect*, *prty*, *rv*) where
tr_id is a non-negative *integer*,
source and *target* are elements from *P.lstates* (i.e. *integers*),
cond is a boolean condition from the global *system state*(Def 6),
effect is a function that modifies the global *system state*(Def 6),
prty and *rv* are *integers.*

Definition 6. (System State)
A global *system state* is a tuple of the form (*gvars*, *procs*, *chans*, *exclusive*, *handshake*, *timeout*, *else*, *stutter*) where
gvars is a finite set of *variables* (Def 1) with *global* scope,
procs is a finite set of *processes* (Def 4),
chans is a finite set of *message channels* (Def 3),
exclusive and *handshake* are *integers*,
timeout, *else* and *stutter* are *booleans.*

2.4 Related Work

2.4.1 Smart City

some background into their approach[6]
some comparison to this project

2.4.2 Privacy Enhancing Technologies (PET)

compare this to your work

3

Modeling & Specification

3.1 Definitions

3.1.1 Basic WSN

At first, a basic Wireless Sensor Network was defined to help the project start off. It consisted of a set of collection nodes (referred to as "nodes"), a central server (referred to as "the server") and finally an environment (the observed source). An illustration of this can be seen in Figure 3.1. This example became the initial working example for the project and helped shape the first models.

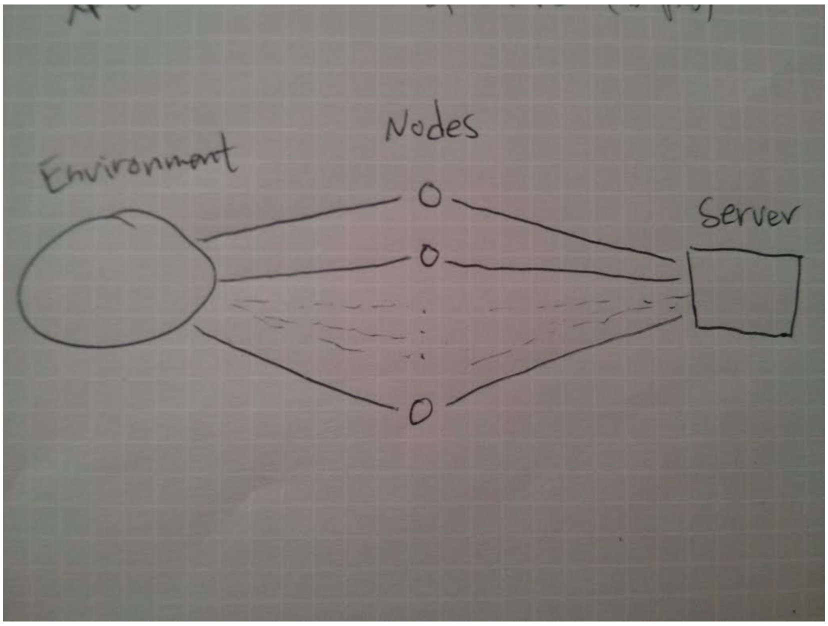

Figure 3.1: An illustration of a Wireless Sensor Network

In this setup, the environment is considered an entity (same as a node or a server). This simplification was made so the environment would be easier to manage in a modeling perspective, as an environment in reality could be a lot of different things:

- *list some environment examples*

We will henceforth refer to entities in the system as **actors** in the system.

textflow in report might be weird now

3.1.2 Actors

Now to describe the interaction between two actors in the system, a behavior model was used:

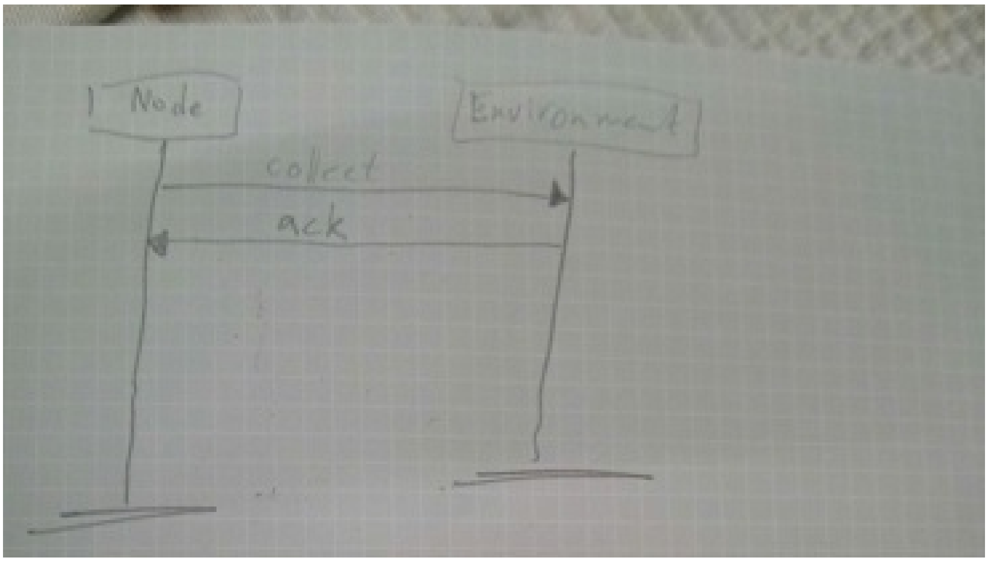

Figure 3.2: Behavior Model Example

Where the name of each actor is shown in the boxes at the top. The message channel used between them is shown as the arrows, where the arrow-head points to the actor receiving the message and the message is referenced above it. Finally the ordering of the messages are in a descending order from the top, meaning the first message sent is shown furthest to the top of the figure and the horizontal lines at the end means the end of the communications.

noted?

can 'communication' be used?

3.1.3 Decisions

A decision, or a decision procedure is an algorithm that terminates with a yes or no answer given a decision problem.[7] *more text regarding decision processes will be added here*

Definition 7. (Decision Process)
A process is called a decision process for T if it is sound and complete with respect to every formula of T.

Definition 7 also requires some more definitions, but this is an important one so added it for now.

3.1.4 Over-Collection

will explain the meaning over-collection for this project

Definition 8. (Collecting)
A process P collects a data point d in a state s if after leaving the state then $d \in \{P_{c'} \setminus P_c\}$.

Definition 9. (To Function)
define what it means for a process to function

Definition 10. (Over-Collection)
Over-collection is the state when a process collects more data than it requires to function.

Formal Definition: Let a process P be able to collect data entries and to evaluate boolean expressions.
$P_{eval} : D \to \textbf{Bool}$
Let a service $S(x, y, ...)$ be a boolean expression depending on variables $x, y, ...$
We say the process P dedicated to the service S, noted $< P, S >$, over-collects data if and only if P collects any data concerning one of the variables appearing in S after S has been evaluated to be true.
$< P, S >$ over-collects iff $\{D \in P_{collection}\} \wedge \{P_{eval}(D) = \textbf{True}\}$ something like that, but with S

3.2 Modeling

As a starting point for defining the models, first different architectural choices were considered. This was done to help define different cases of Wireless Sensor Network that could use decisions. The different variations initially considered were:

- Centralized or Decentralized decision
- One or multiple sensor nodes
- Conjunctive or Disjunctive decision procedure
- Centralized or Peer-to-Peer communication add this in the text

The first choice reflected how much the sensor nodes would analyze the data. Since Wireless Sensor Network has a processing unit, they could potentially analyze the 'potentially' maybe irrelevant wording collected data and make a decision on their own. The second choice simply reflected how many nodes were connected to the same server. The third choice reflected how the decision were processed, if the data from a single data point could trigger a entry? decision or if the decision considered data from multiple entries before triggering.
To start of, models were made for each of the choices except conjunctive decision procedures. This was due to that a conjunctive decision procedure would require a more sophisticated algorithm to analyze the data than the other choices, which would require additional time for just one variation. Also this variation wasn't considered to be crucial to the project's aim, since the disjunctive decision procedure still presented interesting features for analyzing the system. So to start off, it wasn't focused on but still was kept as a consideration for further iterations.

promela, go?

Now the project sought to define a model for the basic Wireless Sensor Network , from 2.1 ref? so three actors were defined: *Node*, *Server* and *Environment*.

3.2.1 Server Actor

The server is the is the actor receiving messages from nodes and storing it for later usage. A server's behavior will vary depending on the structure of the system. If the decision is taken centrally the server will be the one checking for over-collection, otherwise it will be a node. Also if the communication is managed through the server, if the nodes doesn't communicate with each other, the server will act as a repeater for the decision.

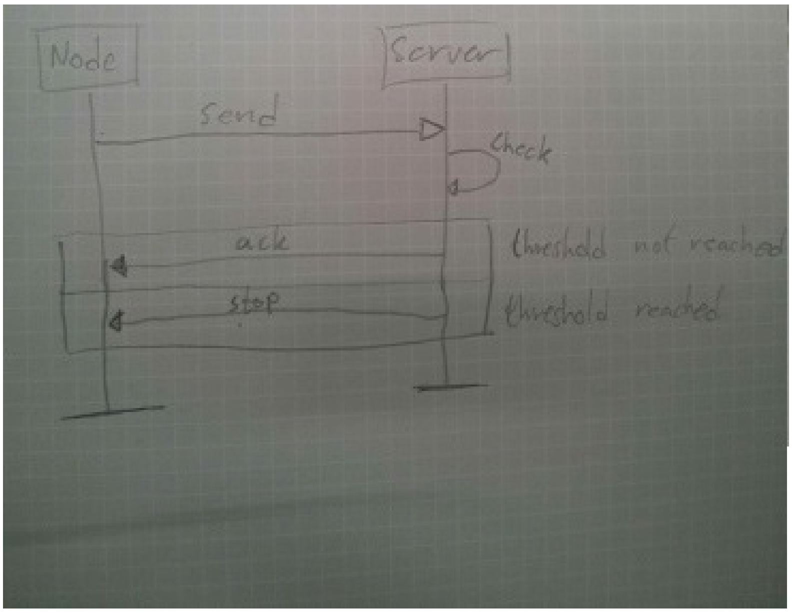

Figure 3.3: Behavior Model between Server and the Node

In Figure 3.3 is the behavior for a system where server makes the decision and nodes doesn't communicate with each other. First, the node sends some data, the server checks for over-collection and replies accordingly. The response will either be a "stop" signaling that over-collection has occurred and the node should stop collecting or it tells it that it can continue collecting.
This behavior can be described using states as well, as shown in Figure 3.4. The same notations are used for the messages sent between the actors except "check" is noted as the state named "Waiting".

3.2.2 Environment Actor

The process for the environment actor had two steps:

1. Generate random data
2. Serve random data to a requesting node

As mentioned before, the first step is not intuitive for an environment since the observed source isn't randomly varying, but for modeling purposes this is a simplification made to reduce the complexity of the model. In Figure 3.5 the behavior between a node and the the environment is described.

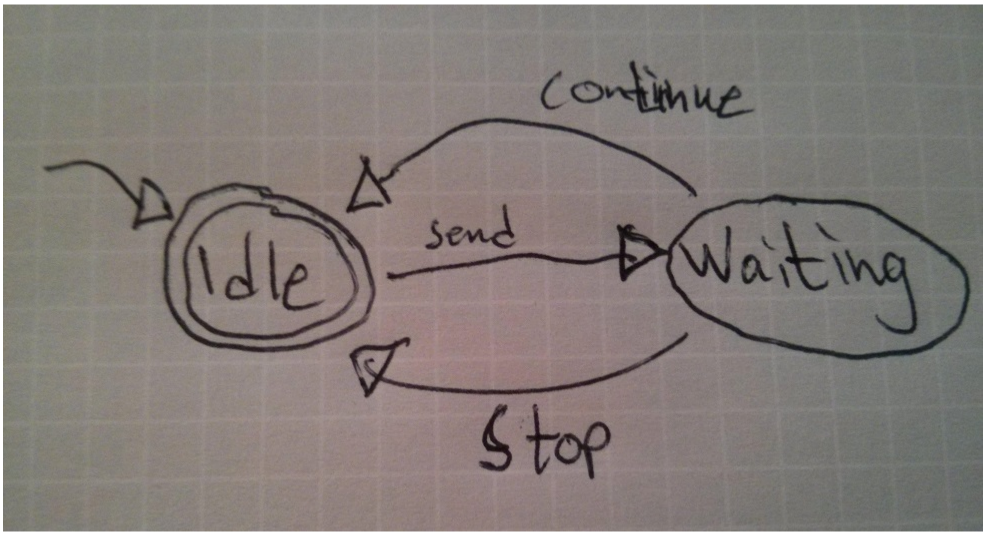

Figure 3.4: States of the Server process

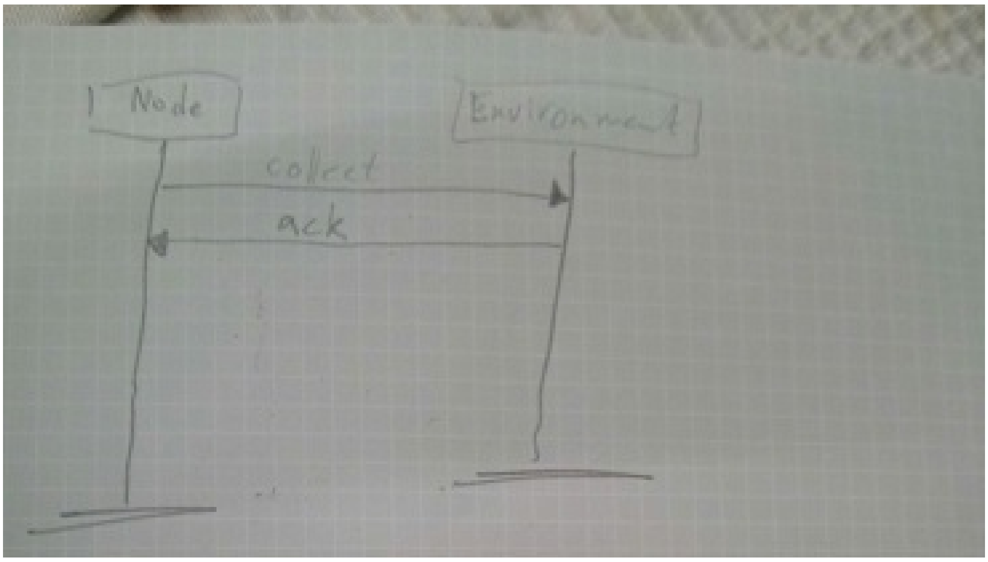

Figure 3.5: Behavior Model for the Environment

This behavior can be described using states, as shown in Figure 3.6. Here the process starts by generating some data and then sends it's to a node asking to "collect" it. fix image related to this paragraph

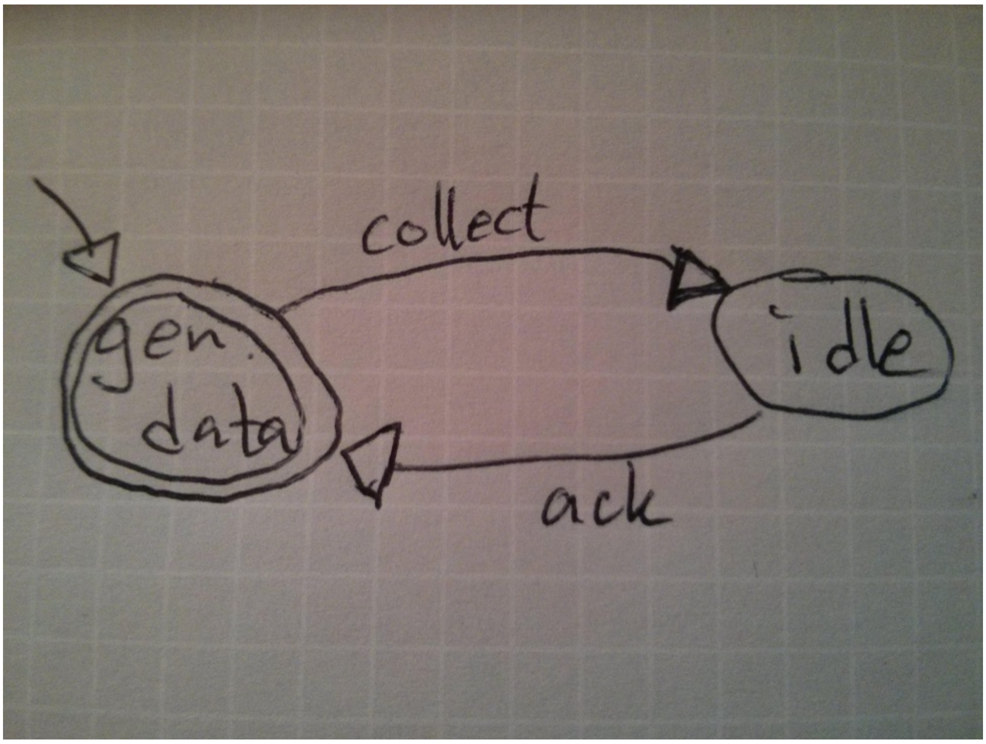

Figure 3.6: States for the Environment Process

3.2.3 Node Actor

As seen in the behavior model for the node actor (Figure 3.7), it captures the majority of a typical scenario for the entire system. That is intuitive since the node communicates with both of the other actors of the system and is a central part of the system. The behavior described is only a scenario where the data being sent is not causing over-collection. In Figure 3.8 instead, is a scenario where over-collection occurs.

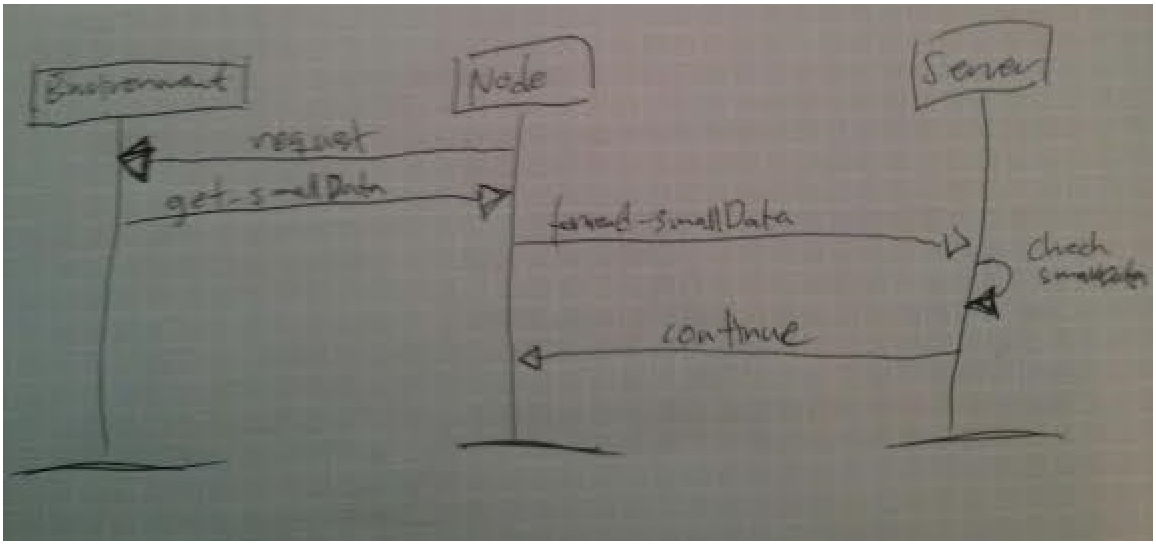

Figure 3.7: Behavior Model for a Node

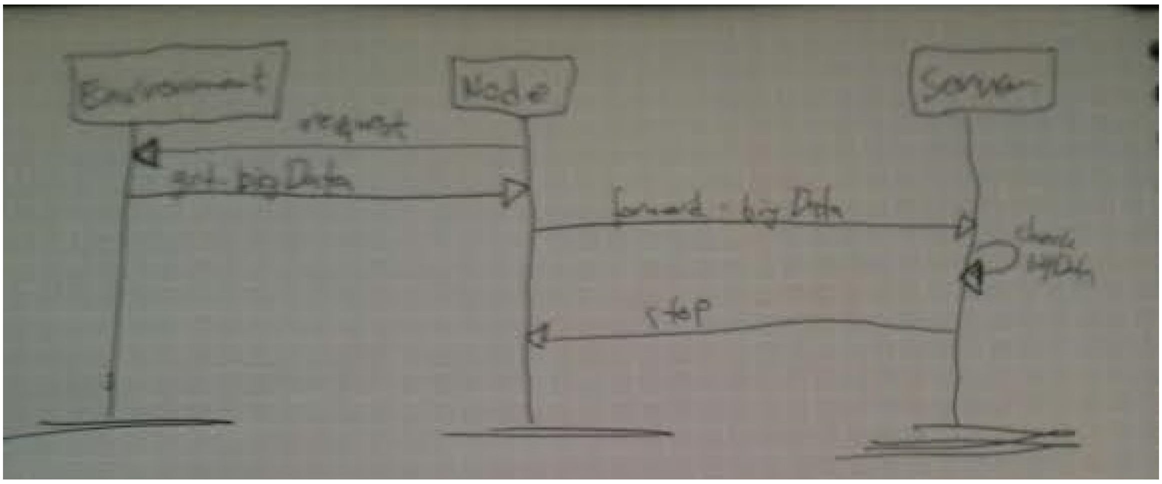

Figure 3.8: Behavior Model for a Node over-collecting

Describing this behavior using states, as in Figure 3.9,

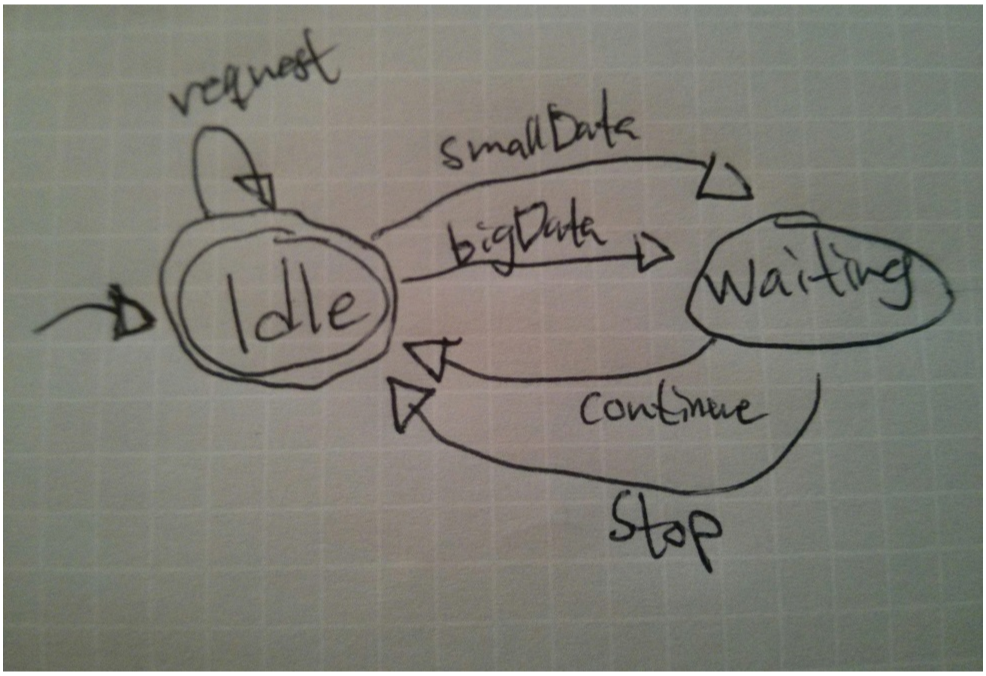

Figure 3.9: States for the Node Process

3.3 Specification

3.3.1 Properties

The first model had properties for **safety correctness** and **liveness**. Due to the simplicity of the model, made both of them also rather simple to express. The correctness property was stated as follows:

Definition 11. Safety Correctness
When over-collection has occurred, the system should stop collecting.

LTL: $\Box(O \rightarrow (\Diamond D))$

explain the last clause

Where **O** and **D** corresponds to the event that over-collection has occurred and the collection is stopped respectively. The liveness property was stated as:

Definition 12. Liveness
The program shall collect until over-collection has occurred.

LTL: $\Box(\neg D \text{ Until } (\Diamond O))$

until sign?

Where **D** and **O** are the same events as described previously.

3.3.2 Extensions

I'm saving this section for other properties that can be added.

4 Design

introduction-text: I seek to use SPIN/Promela for my models and first I need to justify why I did so and compare it to other tools...

Analysis of UPPAAL vs. TLC for verifying the WS-AT Protocol. [8]

Survey regarding the NuSVM "symbolic" model checker.[9]

Tool	**SPIN**	**UPPAAL**	**NuSVM**
specification language	promela	timed automata network with shared variables	...
necessary user's background	programming	programming	...
expressiveness of spec. language	...	restricted, communicating state machines, C-like (but finite) data structures, inductive approach	...
model checker characteristics	...	verifies the full specification language (with time)	...
modeling / verification speed	...	slower modeling, faster verification	...
verification of time/cost features	...	straightforward modeling and state-of-the-art verification support	...
parameterized reasoning	...	...	...

Table 4.1: Comparison between the model checkers SPIN, UPPAAL and NuSVM. [10]

4.1 System Description

first the proposed system will be described, with overlay of the architecture and how it's intended to work.

4.2 Modeling it in Promela

Explain the systematic translation to promela models, motivate that I don't introduce errors

FSA models contain States, Conditions, Steps? ...

explain GOTOs in promela?

are these the correct terms?

rewriting this part, stopped here since it only was a draft of how I could do it.

The translation to Promela from the FSA-models was made by the following steps:

States: Were translated into *labels* and and a move between two states were translated into *GOTO*-statements and corresponding message sent became *messages* sent in *message channels* between the two actors.

Conditions: Were translated into *if*-clauses.

4.3 Verification

Discuss the results from the verification, present modifications done to fix any errors that might occur (perhaps show a interesting case of this).

5

Implementation

Discuss different approaches to verify the implementation and argue for the one I decided on.

5.1 Code Generation

Decide on a tool, discuss it

5.2 Satisfaction

Explain how I used my approach to verify the implementation

5.3 Analysis

Analyze the result of the generation and discuss limitations on the current models. E.g. redundancy from the generation or weaknesses in terms of security

6

Discussion

Discuss different choices made and why they were made.

- discuss why I used formal verification & model checking instead of traditional approaches
- discuss why I didn't build all models from the start
- discuss why I made simplifications to the initial models
- discuss why I chosed to use SPIN/Promela as a tool

7
Conclusion

Conclude the results of the report, did it go as expected? What progress did you make and what didn't you achieve that you had hoped? Did you reach the aim stated and did you keep yourself in the scope & limitations?

8
Ethics

this section will discuss ethical aspects and what ethical impacts it can have.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] E. D. P. Supervisor, "Regulation (ec) no 45/2001," December 2000. [Online]. Available: https://secure.edps.europa.eu/EDPSWEB/edps/EDPS/Dataprotection/Glossary/pid/74

[3] P. Bjesse, "What is formal verification?" *ACM SIGDA Newsletter*, vol. 35, no. 24, p. 1, 2005.

[4] E. M. Clarke, O. Grumberg, and D. Peled, *Model checking*. MIT press, 1999.

[5] G. J. Holzmann, *The SPIN Model Checker - Primer and Reference Manual*. Lucent Technologies Inc., Bell Laboratories, 2004.

[6] Y. Li, W. Dai, Z. Ming, and M. Qiu, "Privacy protection for preventing data over-collection in smart city," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1339–1350, 2016.

[7] D. Kroening and O. Strichman, *Decision Procedures*. Springer, 2008.

[8] A. P. Ravn, J. Srba, and S. Vighio, "A formal analysis of the web services atomic transaction protocol with uppaal," in *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*. Springer, 2010, pp. 579–593.

[9] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "Nusmv: a new symbolic model checker," *International Journal on Software Tools for Technology Transfer*, vol. 2, no. 4, pp. 410–425, 2000.

[10] E. Bortnik, N. Trčka, A. J. Wijs, B. Luttik, J. M. van de Mortel-Fronczak, J. C. Baeten, W. J. Fokkink, and J. Rooda, "Analyzing a χ model of a turntable system using spin, cadp and uppaal," *The Journal of Logic and Algebraic Programming*, vol. 65, no. 2, pp. 51–104, 2005.