Distributed Intelligence-Assisted Autonomic
Context-Information Management
A context-based approach to handling vast amounts of heterogeneous IoT data

Hasibur Rahman

Academic dissertation for the Degree of Doctor of Philosophy in Computer and Systems Sciences at Stockholm University to be publicly defended on Wednesday 24 January 2018 at 13.00 in Lilla Hörsalen, Nod building, Borgarfjordsgatan 12.

Abstract
As an implication of rapid growth in Internet-of-Things (IoT) data, current focus has shifted towards utilizing and analysing the data in order to make sense of the data. The aim of which is to make instantaneous, automated, and informed decisions that will drive the future IoT. This corresponds to extracting and applying knowledge from IoT data which brings both a substantial challenge and high value. Context plays an important role in reaping value from data, and is capable of countering the IoT data challenges. The management of heterogeneous contextualized data is infeasible and insufficient with the existing solutions which mandates new solutions. Research until now has mostly concentrated on providing cloud-based IoT solutions; among other issues, this promotes real-time and faster decision-making issues. In view of this, this dissertation undertakes a study of a context-based approach entitled Distributed intelligence-assisted Autonomic Context Information Management (DACIM), the purpose of which is to efficiently (i) utilize and (ii) analyse IoT data.

To address the challenges and solutions with respect to enabling DACIM, the dissertation starts with proposing a logical-clustering approach for proper IoT data utilization. The environment that the number of Things immerse changes rapidly and becomes dynamic. To this end, self-organization has been supported by proposing self-* algorithms that resulted in 10 organized Things per second and high accuracy rate for Things joining. IoT contextualized data further requires scalable dissemination which has been addressed by a Publish/Subscribe model, and it has been shown that high publication rate and faster subscription matching are realisable. The dissertation ends with the proposal of a new approach which assists distribution of intelligence with regard to analysing context information to alleviate intelligence of things. The approach allows to bring few of the application of knowledge from the cloud to the edge; where edge based solution has been facilitated with intelligence that enables faster responses and reduced dependency on the rules by leveraging artificial intelligence techniques. To infer knowledge for different IoT applications closer to the Things, a multi-modal reasoner has been proposed which demonstrates faster response. The evaluations of the designed and developed DACIM gives promising results, which are distributed over seven publications; from this, it can be concluded that it is feasible to realize a distributed intelligence-assisted context-based approach that contribute towards autonomic context information management in the ever-expanding IoT realm.

Keywords: Internet of Things, Context information, Intelligence, Edge computing, Distributed computing.

Stockholm 2018
http://urn.kb.se/resolve?urn=urn:nbn:se:su:diva-149513

ISBN 978-91-7797-088-0
ISSN 1101-8526

Department of Computer and Systems Sciences
Stockholm University, 164 07 Kista
DISTRIBUTED INTELLIGENCE-ASSISTED AUTONOMIC CONTEXT-INFORMATION MANAGEMENT

Hasibur Rahman
Distributed Intelligence-Assisted Autonomic Context-Information Management
A context-based approach to handling vast amounts of heterogeneous IoT data

Hasibur Rahman
To my parents
Abstract

As an implication of rapid growth in Internet-of-Things (IoT) data, current focus has shifted towards utilizing and analysing the data in order to make sense of the data. The aim of which is to make instantaneous, automated, and informed decisions that will drive the future IoT. This corresponds to extracting and applying knowledge from IoT data which brings both a substantial challenge and high value. Context plays an important role in reaping value from data, and is capable of countering the IoT data challenges. The management of heterogeneous contextualized data is infeasible and insufficient with the existing solutions which mandates new solutions. Research until now has mostly concentrated on providing cloud-based IoT solutions; among other issues, this promotes real-time and faster decision-making issues. In view of this, this dissertation undertakes a study of a context-based approach entitled Distributed intelligence-assisted Autonomic Context Information Management (DACIM), the purpose of which is to efficiently (i) utilize and (ii) analyse IoT data.

To address the challenges and solutions with respect to enabling DACIM, the dissertation starts with proposing a logical-clustering approach for proper IoT data utilization. The environment that the number of Things immerse changes rapidly and becomes dynamic. To this end, self-organization has been supported by proposing self-* algorithms that resulted in ten organized Things per second and high accuracy rate for Things joining. IoT contextualized data further requires scalable dissemination which has been addressed by a Publish/Subscribe model, and it has been shown that high publication rate and faster subscription matching are realizable. The dissertation ends with the proposal of a new approach which assists distribution of intelligence with regard to analysing context information to alleviate intelligence of things. The approach allows to bring few of the application of knowledge from the cloud to the edge; where edge based solution has been facilitated with intelligence that enables faster responses and reduced dependency on the rules by leveraging artificial intelligence techniques. To infer knowledge for different IoT applications closer to the Things, a multi-modal reasoner has been proposed which demonstrates faster response. The evaluations of the designed and developed DACIM gives promising results, which are distributed over seven publications; from this, it can be concluded that it is feasible to realize a distributed intelligence-assisted context-based approach that contribute towards autonomic context information management in the ever-expanding IoT realm.
Sammanfattning


Acknowledgements

‘Read! In the name of thy Lord Who createth,
Createth man from a clot.
Read: And thy Lord is the Most Bounteous,
Who teacheth by the pen, Teacheth man that which he knew not.’

[Al-Qur’an 96:1-5 – interpretation of the meaning]

All praise belongs to the Lord of the worlds!

Back in January 2013, I was at my aunt’s house in my hometown with my elder brother, and I received a message on Skype; I assumed it was about the course I was involved in teaching at Mid Sweden University. When I saw the message, my first thought was that I was probably the wrong recipient. The message asked whether I was interested in doing a Ph.D. in wireless sensor networks, and the sender was my principal supervisor. Fresh from conducting a research as part of my Master’s thesis about improving system spectral efficiency within wireless communication, I thought it was a good match. More importantly, my burning desire to be involved in academia played a role in the decision, along with other personal reasons. This is how the road to this Ph.D. study began.

Naturally, I firstly would like to offer my sincere gratitude to my principal supervisor Associate Professor (Docent) Rahim Rahmani, for prompting me to undertake this interesting research and further supporting and encouraging me through the ups and downs. I am also grateful to my co-supervisor, Professor Theo G. Kanter, for allowing me to join the multicultural Immersive Networking research group and for supporting me during this research at several stages and in various capacities. I would also like to take the opportunity to earnestly thank my Master’s thesis supervisor, Mr. Magnus Eriksson (Mid Sweden University), who introduced me to the world of research and taught me how to carry out research. Special thanks go to Dr. Jamie Walters, my ex-teacher, ex-supervisor for a course project, ex-colleague and a good friend, who helped me to settle into the department, the research group (and who also interviewed me prior to joining), and now this great (yes, it is already great 😊) country.

I am also very grateful to three academics (Assoc. Prof. Shengnan Han, Prof. Athanasios Vasilakos, Assoc. Prof. Karl Andersson) who took the time to read my thesis (licentiate and doctoral), and gave constructive and valuable
feedback, which undoubtedly helped to improve the final thesis. I want to thank all the supporting staff and fellow Ph.D. students at DSV, my other group members, Dr. Bin Xiao, Irvin Homem (soon to be Dr.), Dr. KiM Nevelsteen, and Assoc. Prof. Yuhong Li (BUPT, China), for helping out when required. Many of my colleagues were always there to support and/or encourage me, including Dr. Aron, Marihan, Irma (of course, this refers to Irma Muñoz), Dr. Isak, Dr. Henrik, Workneh, Ranil, Ram, Dr. Javier, Sameer, Dr. Meshari, Jean-Claude, Dr. Thashmee, Xavier, Johan Eliasson and Dr. Stephane Junique (ACREO). No particular order is intended; I am truly thankful to each and every one of you in different ways. I would also like to thank the members of the EU FP7 MobiS project, and especially the technical team with whom I worked closely during the initial stages of this research, for their friendly and helpful manner in finishing the project and also for all the dinners together during the project meetings in different European cities. I will especially never forget the last dinner in Ljubljana.

I would like to thank my compatriot, Arif Mahmud (Ankur bhai), for his guidance and encouragement; he was a teacher at MIUN when I started my Master’s studies, and seeing him excelling gave me a great deal of motivation and courage. I would also like to thank several of my classmates, who were my study partners day and night in MIUN’s different lab rooms: Mezbah Uddin, Anas Siddiqui and Khair Zada. I will never forget Edmore Ndaba, Tariq Mahumud and Arif Hossain for your hospitality and friendship. You guys made my stay in Sundsvall enjoyable. I am also thankful to my classmates in B.Sc. Engineering, and particularly Fida Khattak, Muhammad Usman and Ijaz Ahmed, for helping me out in the Master’s programme admission process. Few others who were always there to help me during my Ph.D. studies in Stockholm. Suliman, Atiq bhai, Mukul bhai, Shuvo bhai, and Papon bhai at the beginning of the studies; Mashiur Rahman bhai, Khalid bhai, Moshouri Rahman bhai at the later stages were particularly helpful. Special mentioning of Seame mama and Sume mami; Azad bhai and Joyti bhabi; Jimmy and Hridi bhabi; Mashiur bhai and Ayesha bhabi for their respective friendship, and their adorable kids. You were family away from the family. I am very grateful to my friends, Ashiful Alam, Nazrul Islam, Shaheenur Rahman and Golam Rabbani, who helped me financially in the final stages of my Master’s studies when I needed help most. I am also thankful to my uncle, Shah Jafar Iqbal, and my aunt, Naznin Sultana, for their love and support.

Last but by no means least, I offer my gratitude and love to my beloved family for sacrifices made to support my education throughout my life. I will never forget the patience, support and dedication of my mother, Shamima Sultan, in our education and upbringing. My father, the late Sk Golam Rasul, was a school teacher who motivated me to pursue higher studies and inspired me to follow career in academia. His prayers and my mother’s never-ending love and support have helped to come this far. My only sibling, my elder brother, Advocate Saifur Rahman, not only helped me financially during my
higher studies abroad but also prayed and encouraged me all the way. Finally, thank you to my wife, Fiqa Faria, for your unconditional love and also for your patience and support during the pursuit of my doctoral studies. No thanks are enough for you all: you make my world. You are my strength to withstand any pain.

I am ever grateful to all of you! May God bless you all! Ameen.

Hasibur Rahman (Suzon)
Stockholm, December 2017
List of Papers

Included in this dissertation

The dissertation includes the following papers (arranged chronologically).


[Papers I, III, VI and VII] were published in peer-reviewed conferences, and [papers II, IV and V] were published in a peer-reviewed international journals. All papers were published in conferences and journals that ranked at **level 1** in the Norwegian ranking system.

I also contributed to the following papers which are not included in the dissertation. [Papers VIII - XI] are by-products of this study and contributed to some of the results in this dissertation.


Author Contributions

The main contribution of this dissertation to the research community is summarized as a list of publications in peer-reviewed conferences and journals as follows:

Publication I

Context-Based Logical Clustering of Flow Sensors - Exploiting Hyper-Flow and Hierarchical DHTs

This publication introduced the notion of logical-clustering, the aim of which was to cluster connected Things based on context similarity with the aim of fostering context information utilization (contextualized IoT data). This idea expressed the concept of grouping similar physically distributed context sources but logically grouped based on the similar context information. This is opposed to traditional physical clustering where grouping is usually done based on physical nearness. The proposed idea can also be perceived as filtering similar context information from heterogeneous context information. In view of this, the publication dealt with the feasibility of logical-clustering, and presented the technical representation and computational efficiency of the logical-clustering approach. A two-tier hierarchical distributed hash tables (H-DHTs) based system model was explored in this study, in order to decouple computation for the connected Things using the idea of several physically sinks distributed but synchronized centrally, i.e., a logical-sink. The author largely contributed to designing and developing the system model and further evaluated the approach based on a MATLAB simulation. The author also contributed the bulk of the writing.
Publication II

On Performance of Logical-Clustering of Flow Sensors

This work was an extension of the earlier work in publication I and involved the design of wireless sensor networks of logical- clustering using a network simulator-3 (ns-3) simulation tool. The study was done with the goal of verifying the scalability, reliability and reachability of the designed network for logical-clustering approach to reflect the real-world scenario. The paper also provided potential use cases for logical-clustering. The author’s main contribution in this paper comprised designing, developing and evaluating a logical-clustering approach in ns-3, and verifying the logical-clustering system model based on a combination of PROMELA and SPIN combination. The author also contributed the bulk of the writing.

Publication III

Enabling Scalable Publish/Subscribe for Logical-Clustering in Crowdsourcing via MediaSense

This publication studied the scalable distribution of context information, both to and from context sources. The study undertook the design, development and evaluation of a Publish/Subscribe (PubSub) model by extending the DCXP protocol. The outcome of this study demonstrated scalable and very fast subscription matching, along with high publication of items. In addition, the paper presented the properties and requirements that are expected to drive the future Internet of Things i.e. the Internet of Everything. By extending the DCXP protocol to realize distributed publication and subscription of clustering identification, its correctness has been proven using a distributed hash table-based MediaSense IoT platform. The key contributions of this study were the design and development of algorithms for publisher and subscriber by extending the DCXP protocol. Evaluations confirmed that the approach performed better than existing PubSub models such as ToPSS and PARDES. The author led the study, and proposed, designed, developed and evaluated the distributed Publish/Subscribe model. The author was also the chief contributor to the writing.
Publication IV

Supporting Self-Organization with Logical-Clustering Towards Autonomic Management of Internet-of-Things

This publication studied and proposed a solution for self-organization for controlling massive numbers of participating connected Things, that is, entities in an IoT scenario. This solution contributed to realize an organized edge controller in the logical-clustering approach with respect to managing entities in an IoT environment. Various self-* aspects of the autonomic computing vision were designed and developed which allow automatic seamless integration of entities into an IoT controller, discovery of other entities, compensation for and correction of entity failures and optimization of the clustering identification distribution. This study also outlines additional research challenges involved in a fully operational autonomic management of IoT. The author was the leader of this study and the primary contributor to this publication, and proposed, designed, developed and evaluated solutions to the self-organization algorithms, i.e. the self-* capabilities, and proved their correctness using the MediaSense platform. The author made the chief contribution to the writing.

Publication V

Enabling Distributed Intelligence Assisted Future Internet of Things Controller (FITC)

This publication introduced the idea of distributing intelligence by leveraging both edge- and cloud-based solutions. The idea behind this particular research is to extract low-level knowledge closer to the actual Things by exploiting the contextualized raw data, providing further higher-level intelligence in the cloud. This low-level intelligence enables faster decisions, actions and predictions. To achieve this, algorithms leveraging belief networks and similar to reinforcement learning were designed, developed and evaluated. The results obtained using a Raspberry Pi (as an edge controller) suggest that this approach is viable for achieving faster response times and reducing the requirements of rules, and that further learning is feasible based on experience. The author led the study, proposed, designed, developed and evaluated solutions to this study, and was the chief contributor to the writing.
Publication VI

*Multi-Modal Context-Aware reasoNer (CAN) at the Edge of IoT*

This publication described an approach to providing reasoning based on the contextualized raw data. More often than not, IoT requires the control of more than one IoT application at the edge by a single controller. This solution therefore proposed the provision of different reasoning techniques for different IoT applications. Reasoning is pivotal in the vision of extracting knowledge, and extraction of knowledge is also required to be done closer to the Things, that is, at the edge for each of the IoT applications. Two different reasoning techniques were employed for three IoT applications in order to demonstrate the feasibility of the approach. The paper described the proposal, design and development of a solution to counter this vision, where performance was verified in terms of latency. The author again led the study and was the chief contributor to the writing.

Publication VII

*Supporting IoT Data Similarity at the Edge towards Enabling Distributed Clustering*

This publication is connected with the first publication, in which the physically distributed clustering approach was proposed. This paper continued by proposing a solution that is capable of finding IoT data similarity. This was facilitated by extending the Jaro-Winkler and Jaccard-like similarity algorithms to the DCXP protocol. More specifically, in order to handle for both numerical and textual data, Jaro-Winkler is proposed and evaluated for textual data and Jaccard-like algorithm is designed and evaluated for numerical data. The feasibility of this approach was verified using an edge controller in terms of different IoT data acquisition scenarios; the results confirm the feasibility of the approach. The author was the sole contributor to this research.
## Contents

Abstract ......................................................................................................... vii

Sammanfattning ............................................................................................. ix

Acknowledgements ........................................................................................ xi

List of Papers ............................................................................................... xiv

Author Contributions .................................................................................... xvii

Abbreviations .............................................................................................. xxiii

List of Figures .............................................................................................. xxv

List of Tables .............................................................................................. xxvii

1 Introduction ............................................................................................ 1
  1.1 Motivation .......................................................................................... 5
  1.2 Problem Statement .............................................................................. 9
  1.3 Research Question .............................................................................. 11
  1.4 Research Methodology ....................................................................... 12
  1.5 Dissertation Disposition .................................................................... 12
  1.6 Summary ............................................................................................ 14

2 Background Study................................................................................ 15
  2.1 IoT Architectures .............................................................................. 15
  2.2 State-of-the-Art IoT Solutions ............................................................... 17
  2.3 Towards the Future IoT ...................................................................... 18
  2.4 Clustering ........................................................................................... 19
  2.5 IoT Autonomic Computing ................................................................... 21
  2.6 Context-Aware Computing in the IoT .................................................... 22
  2.7 IoT Intelligence .................................................................................... 22
  2.8 Enabling the Dissemination and Management of Distributed Context Information 24
  2.9 Summary ............................................................................................ 26

3 Research Method................................................................................. 27
  3.1 Philosophical Assumptions ................................................................. 27
  3.2 Methodological Overview ................................................................... 28
# DACIM Architecture

- **4 DACIM Architecture**
  - 4.1 System Model
  - 4.2 Clustering Heterogeneous Context Information
  - 4.3 The Publish/Subscribe Model
  - 4.4 Self-Organization Towards Enabling Autonomic Management of IoT
  - 4.5 Distributed Intelligence
  - 4.6 Multi-modal Context-Aware reasoner (CAN)
  - 4.7 The Architecture
  - 4.8 Summary

# Evaluation of DACIM

- **5 Evaluation of DACIM**
  - 5.1 Network Performance
  - 5.2 Distributed Clustering of Heterogeneous Context Information
  - 5.3 Publish/Subscribe evaluation
  - 5.4 Self-Organization Towards Enabling Autonomic Management of IoT
  - 5.5 Intelligence Evaluation at the Edge Controller
  - 5.6 Summary

# Conclusions

- **6 Conclusions**
  - 6.1 Discussion
  - 6.2 Future Work

# References
Abbreviations

4G    Fourth-generation
5G    Fifth-generation
AI    Artificial Intelligence
BLE   Bluetooth Low Energy
CI/ConIn  Context Information
Context-ID  Context-based clustering identification
DACIM Distributed intelligence-assisted Autonomic Context Information Management
DAP    Decision, Actions, and Predictions
DCXP   Distributed Context eXchange Protocol
D2D    Device-to-Device
DHT    Distributed Hash Table
DNS    Domain Name System
DoS    Denial of Service
E2S    Entity-to-Sink
FITC   Future Internet of Things Controller
H-DHT  Hierarchical Distributed Hash Table
HLI    High-level intelligence
IoE    Internet of Everything
IoT    Internet of Things
LEACH Low Energy Adaptive Clustering Hierarchy
LLI    Low-Level Intelligence
MAC    Media Access Control
MB     Megabytes
ms     Milliseconds
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFC</td>
<td>Near Field Communication</td>
</tr>
<tr>
<td>NFV</td>
<td>Network Function Virtualization</td>
</tr>
<tr>
<td>P2P</td>
<td>Peer-to-Peer</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PROMELA</td>
<td>Process or Protocol Meta Language</td>
</tr>
<tr>
<td>PubSub</td>
<td>Publish/Subscribe</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio Frequency Identification</td>
</tr>
<tr>
<td>RSS</td>
<td>Rich Site Summary</td>
</tr>
<tr>
<td>SDN</td>
<td>Software Defined Networking</td>
</tr>
<tr>
<td>S2E</td>
<td>Sink-to-Entity</td>
</tr>
<tr>
<td>S2S</td>
<td>Sink-to-Sink</td>
</tr>
<tr>
<td>ToPSS</td>
<td>Toronto Publish/Subscribe System</td>
</tr>
<tr>
<td>TTL</td>
<td>Time to Live</td>
</tr>
<tr>
<td>UCI</td>
<td>Universal Context Identifier</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>WPAN</td>
<td>Wireless Personal Area Network</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Network</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1: ConIn management (adapted from [16]) ...................... 3
Figure 2: A simplified IoT architecture ........................................ 4
Figure 3: Future IoT properties ......................................................... 6
Figure 4: Research problem and motivation map ..................... 9
Figure 5: Dissertation road map .................................................... 14
Figure 6: Cycle of future IoT properties ........................................ 21
Figure 7: Information-knowledge pyramid hierarchy (adapted from [92] and [93]) ................................................................. 23
Figure 8: Typical PubSub model (adapted from [56]) ................. 25
Figure 9: Applying the design science research method [63] ........ 30
Figure 10: Artefacts for the DACIM ................................................. 32
Figure 11: Two tier H-DHT concept (Adjusted from [Paper I] and [78]) ................................................................. 41
Figure 12: MediaSense and DCXP [78] ............................................. 42
Figure 13: Example of a two-tier network .................................... 44
Figure 14: E2S and S2E Communication ....................................... 45
Figure 15: S2S communication ....................................................... 45
Figure 16: Flow chart ................................................................. 46
Figure 17: Communication between edge and cloud controller ...... 49
Figure 18: DCXP-enabled MediaSense as PubSub model .......... 50
Figure 19: Approach to utilizing MediaSense ................. 50
Figure 20: Supporting self-organization with logical-clustering ....... 54
Figure 21: Discovering entities ................................................... 55
List of Tables

Table 1: Key concepts .............................................................................. 40
Table 2: Self-* capabilities of an autonomic entity [31] ............. 53
Table 3: Examples of knowledge at the edge ................................. 65
Table 4: Bayesian reasoning for sprinkler activation ............... 66
Table 5: 3T3C cluster size vs. % similarity ...................................... 74
Table 6: Time required for clustering textual data by three different similarity metrics .............................................................. 76
Table 7: Required time for publishing .............................................. 77
Table 8: PubSub messages/sec ......................................................... 79
Table 9: Subscription matching ........................................................ 79
Table 10: Entity-join performance .................................................. 81
Table 11: Entity discovery (dynamic) .............................................. 81
Table 12: Discovery accuracy (concurrent) ................................. 81
Table 13: Advantages of Belief-Network over rule-based ........ 82
1 Introduction

The Internet-of-Things (IoT) has lately witnessed a shift in focus, where handling for the vast amount of IoT data generated from the estimated hundreds of billions of Things involves both a substantial challenge and high value. This shift in focus has been due to the widespread adoption of IoT in everyday life. Recent advancements in the development of pervasive technologies such as Radio Frequency Identification (RFID)/Near Field Communication (NFC), Wireless Personal Area Network (WPAN), high speed communication (4G/5G), Bluetooth Low Energy (BLE), sensing and actuating, etc. have paved the way for the IoT pervasiveness. Every aspect of human life promises to be impacted which would drive the Future Internet by expanding the scope of IoT to many smart applications such as SmartHome, SmartHealth, SmartAgriculture, SmartEnvironment, SmartParking, Traffic, SmartManufacturing, Waste Management, etc. This also corresponds to the earlier vision of IoT, which was to connect any unconnected objects in order to collect and share data.

This widespread of IoT applicability enables the distributed acquisition and dissemination of heterogeneous IoT data in real-time communication, which faces the prospect of encountering a vast amount of data in the IoT realm. However, most of the data collected from the IoT is never utilized or analysed. In [1], Seth mentions that less than 1% of data from the IoT is ever analysed, while according to IBM, 90% of data generated by the IoT is never utilized or managed [2]. Earlier, it was predicted that an enormous amount of IoT data would remain underutilized if not managed properly [3, 4], and research into IoT requires efficient and effective data management [5]. The present IoT focus, and its main challenge, lies in handling the IoT data by making sense of current data rather than just past data [Paper VII]. Furthermore, this should be done with low-latency, in real-time and closer to the data origin [Paper V]. Making sense of IoT data, both in real-time and with low-latency, is envisioned for the future fifth-generation (5G) enabled IoT [6]. Context, which has been investigated in pervasive computing over recent decades and is one of the techniques for making raw-data useful, and context-aware computing, that is, context-based solutions, are proven to be successful in making sense of raw-data [5, 7]. One of the alternatives to context-based solutions could be semantic-based solutions as shown in [8]; however, semantics deals with the study of the meaning, and providing meaning is a pre-requisite which a context-based solution aims to offer. Moreover, context-based solutions
have been identified as a key enabler for IoT [5, 9, 10] and would support autonomous control and context information (ConIn) retrieval [7, 11].

However, existing solutions have been referred to as infeasible and inefficient; and earlier IoT solutions were primarily focused on the management of and communication between Things, where context-awareness was ignored and did not satisfy the demands of the IoT [5]. This underscores the need for research into enabling context-based solutions, and this study aims to undertake this. This dissertation explores a context-based approach to handling the ever-increasing distributed ConIn, the goal of which is to utilize ConIn by grouping similar ConIn [Papers I, VII] and analyse ConIn [Papers V, VI].

Herein, a context-based approach implies that the raw-data collected from the Things is first contextualized, in order to utilize (i.e. group together similar ConIn) and analyse (i.e. alleviate intelligence-of-things) [Paper V]. Contextualization in IoT plays a significant role with regard to providing intelligence, since the collected raw-data does not provide any usefulness [1, 5, 12]. The contextualized raw-data, that is, ConIn, provides meaning to the raw-data. An example of this contextualization is the provision of meaning to a temperature sensor value. For instance, let us assume that a temperature sensor gives a value of 20 °C; this does not tell anything else about this data. To put this value into context, if other contextual information needs to be added, for example: when (time: 07:00 AM) and where (origin: bedroom) this reading was recorded and who (id: indoor.temperature@bedroom) was the originator, etc.; this gives a better understanding of the situation. This can be inferred as “Current bedroom temperature is 20 °C (warm) at 07:00 AM (morning).” Thus, any application which is exploiting this particular sensor can make better decisions at the edge (see [Paper V] for more details). One example of decision-making at the edge based on this particular context information could be the turning off of heating and making breakfast without user intervention, or waiting for cloud to make the decisions, thus automating the process and achieving faster decision-making. Sending a notification to a caregiver in a health monitoring scenario is another example of application of knowledge based on context information at the edge. Therefore, ConIn is useful to make sense of data which requires efficient management in the IoT domain [5, 9]; this is expected to grow in IoT [5, 17].

Context-based solutions advocate the idea of determining actions based on the context and an application that actually corresponds to the context is known as context-aware [13]. On the other hand, context management deals with the idea of managing the contextualization process; this can be divided into the four steps of context acquisition, modelling, reasoning and dissemination [5]. Various approaches have been proposed to handle each step or variations of these steps, for example, the centralized Publish/Subscribe (PubSub) model or middleware for context acquisition and dissemination [5, 8, 14]; W4 Diary, XML, JSON, key-value pairing, 5Ws, etc. for context modelling [5, 15, 17]; and rules/ontology based, probabilistic approaches which, among others,
are context reasoning techniques [5, 15, 16, 17]. These steps also correspond to a cycle as shown in [5, 18] and further enhanced in [16]; this study explores this in terms of autonomic management with minimal outside intervention. Figure 1 illustrates how this dissertation exploits the context-based approach. Here, context reasoning has been replaced with context handling (adapted from [16]), which has subsequently been divided into clustering and reasoning phases. These two phases highlight the context-based approach to the utilization and analysis of ConIn. However, most of the current solutions use web-based protocol and/or middleware-based protocols [5, 8, 14, 15, 16]; among other issues, these create centralized-dependency, and susceptibility to attacks and Denial of Service issues [15].

![Figure 1: ConIn management (adapted from [16])]()

IoT has traditionally used cloud computing based solutions to perform computation and any other tasks related to IoT such as those mentioned above for context management including the contextualization process itself. Prior to this, a gateway is used to collect data from Things; these data are then forwarded to the cloud for further processing and/or actions [8, 14, 19]. Figure 2 gives an insight into how this is typically done. Centralized middleware solutions for context acquisition and dissemination are subject to a single point of failure, and reside mostly in the cloud; this causes delay with respect to context management such as context modelling, reasoning, etc. The PubSub approach needs to move away from a centralized approach to reflect to the heterogeneous and distributed nature of the IoT. Furthermore, the IoT is required to have a fast response with response time to be in milliseconds (ms) [9]. To counter this, an alternative to cloud computing is to bring computation closer to the Things, that is, to the edge of the IoT, and edge computing has been gaining tremendous attention lately [10, 20, 21]. Intelligence for the IoT has so far been conducted in the cloud or at the edge by employing only rules based reasoning. Rules fail to scale well with the proliferation of Things in IoT. Existing edge based IoT solutions do not adequately support efficient context management with respect to intelligence, learning, reasoning, clustering, etc.
This study, therefore, further aims to provide edge centred knowledge by means of distributed intelligence and reasoning; its goal of which is to contribute to autonomic ConIn management [Papers V, VI].

![A simplified IoT architecture](image)

Figure 2: A simplified IoT architecture

Clustering is a useful technique for utilizing (i.e. organizing and exploring) data, and is widely used in almost every discipline. This study exploits clustering to utilize ConIn in the IoT domain. Since ConIn is heterogeneous and distributed, similar ConIn may be produced remotely. Furthermore, a single sensing device may produce different ConIn at different times. In view of this, this dissertation proposes logical-clustering as opposed to physical clustering. Physical clustering is based on physical nearness but fails to cluster similar information remotely. Some examples of organizing and exploring context information in IoT via logical-clustering, in addition to the examples mentioned in [Paper III], can be: (i) a researcher/scientist investigating pollution levels in different parts of a city/country, and wanting to gather similar context information in real-time; (ii) monitoring of the real-time status of machines in factories; (iii) real-time waste management; etc.

This research is undertaken with the aim of realizing a context-based approach with regard to enabling Distributed-intelligence assisted Autonomic Context Information Management (DACIM) to manage the ever-increasing amount of ConIn. The necessity of moving towards a distributed approach in the IoT was earlier stressed in [5, 9, 15]. To this end, this dissertation presents an accumulation of work produced during the course of this study, including an architecture for a logical-clustering approach; a model enabling scalable and faster Publish/Subscribe; self-organization algorithms with high rates of discovery of Things and duplication checks; a distributed intelligence assisted IoT controller capable of faster decision-making, thereby reducing dependency on rules and the cloud, and learning and predicting missing context information; a multi-modal context-aware reasoner offering low latency knowledge extraction for different IoT applications; and a capable distributed clustering approach for both textual and numerical IoT data, of the based vision. The underlying vision aims to utilize and analyse IoT data with the goal
of enabling distributed context information clustering and contributing to distributed intelligence quickly, efficiently and closer to the data origin.

1.1 Motivation

The unprecedented proliferation of Things in the IoT has led to the current existence of almost two Things for each person [22]; this number is estimated at six per person in the Nordic countries [23]. Consequently, the IoT has moved on from its earlier definition and vision. IoT was defined as an interconnection of physical objects which is capable of collecting and sharing data anytime and anywhere [5,9]. The earlier IoT therefore focused predominantly on creating architecture and communication protocols to connect Things, and to collect and share data from the connected Things. In the fast-growing IoT, making sense of collected data involves both significant challenges and value [Papers V to VII], and this is further reflected in the vision for a 5G IoT [6]. Existing approaches to utilizing sensory data have been deemed infeasible and insufficient [5]; moreover, intelligent IoT data utilization needs to be addressed [1,9], which can be achieved by leveraging context-aware computing [5,17]. Context-aware computing is envisioned to play an integral role in driving the future IoT, and this combination will be inseparable along with artificial intelligence [5, Paper V]. However, current context management is limited to web- and/or middleware-based solutions, as outlined earlier. Furthermore, research into the IoT has until now mostly ignored the importance of context-based solutions in real-time, for example context reasoning and dissemination [5]. Moreover, existing solutions use context to provide services to users with little or no focus on the application itself. In addition to providing services to users based on the context, further challenges exist, for example the automation of IoT applications with respect to decision making, taking action, context sharing etc. The IoT is evolving towards a paradigm which integrates people, services, context information and Things [Paper IV], as illustrated in Figure 3 (see section 2.5 also). The approach proposed in this dissertation aims to contribute to a context-based IoT solution in order to handle massive, heterogeneous, distributed IoT ConIn. Mindful of earlier research with regard to the IoT vision, i.e., architectures and communication protocols, this dissertation builds upon an existing context-centric architecture [15,24] and further extends to handling foreseeable IoT challenges.

With the emergence of mobile computing, context has been the subject of extensive research in distributed computing. However, the use of context was initially limited to location, although it has the potential to offer more than this [25]. While there exist many definitions of context, the definition proposed by Dey and Abowd in [7] has been widely accepted [5]. Their definition of context is as follows: “Context is any information that can be used to characterize
the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves”. Motivated by this definition, Walters extended it in his dissertation [15] to an IoT scenario, which is more commonly regarded as context information: “Any subset of information that can be used to characterize the situation of an entity as well as its relationship with other entities including the entity itself”. Here, bearing in mind Walters’ definition of context information, this dissertation explores that subset of information which can be exploited for utilization and analysis. This dissertation therefore uses the following definition of IoT context information: “Any flow of subset of information that characterizes the current situation of an entity as well as its associated subset of information, including other entities’ subset of information, that can be explored for utilization and analysis towards enabling instantaneous, automated, and informed decisions”.

![Figure 3: Future IoT properties](image)

Recently, the landscape of anytime computing has been drastically transformed, and is now rapidly moving towards a seamlessly connected society. The realization of a future society where hundreds of billions of devices (i.e. Things) exist will give rise to a vast amount of ConIn. This will be facilitated, by and large, by the distributed dissemination and acquisition of ConIn from enormous numbers of entities, which points towards an Internet of Everything (IoE) as the future of the IoT. This dissertation therefore identifies the following as properties of the future IoT:

- People
- Pervasive devices
- Internet or Web-enabled services
- Surrounding things
- Context information
This can also be seen in Figure 3, which also portrays how ConIn is generated in the IoT. Heterogeneous ConIn from different IoT applications can be generated locally, while homogeneous ConIn from similar IoT applications can be generated remotely. This variant of IoT ConIn acquisition requires an intelligent solution; the management of the massive amount of heterogeneous ConIn from the future IoT (i.e. the IoE) is impractical with current approaches [3,5,26]. Clustering is one possibility that can aid in efficient management and exploration of ConIn; each cluster is a set of ConIn that exhibits similar characteristics and is dissimilar from other set(s) of ConIn [4]. However, most of the existing clustering approaches are centralized, cloud-based and based on location nearness. An approach which can group similar sets of ConIn both locally and globally (i.e. physically distributed) called logical-clustering is proposed in this dissertation [Papers I and VII]. Cloud has traditionally managed the ConIn, including the process of contextualization, as shown by the context broker in the FIWARE IoT solution [14]. As illustrated in Figure 2, when data has been collected by the gateway, it is forwarded to the cloud for further processing. While cloud computing has been the prevailing choice until now, it involved higher network loads and response time. For example, when raw data is collected from sensors installed in a SmartFarm or SmartHome and an actuation is needed based on the sensory data, the wait for a decision on when and how to perform the actuation from the cloud is time-consuming. In the era of real-time computing, it is fitting to make decisions as quickly as possible, as anticipated in 5G IoT visions [6]. This can be mitigated by bringing computing closer to the Things; edge computing has been suggested for this [10,21,27] and is constantly increasing in applicability and adoption. Recently, intelligence at the edge has also attracted a great deal of attention, and existing edge intelligence relies heavily on rules which can be countered by artificial intelligence (AI) techniques [Paper V]. Edge computing is also capable of providing faster response and data storage, and lower bandwidth and communication overhead etc. [28]. While edge computing is gaining in popularity and offers certain advantages, cloud computing cannot be disregarded, since edge computing offers only solutions for small data in local context. This necessitates the design of an IoT architecture capable of providing a solution by incorporating both edge- and cloud-based solutions [Papers I, V].

Distributed hash table (DHT)-based approaches, for instance MediaSense [24] and SCOPE [29], have been proposed to fulfil these requirements for pervasive computing, including but not limited to the scalable sharing of ConIn in real-time. Both of these solutions aim to offer unified APIs to deliver an efficient and scalable platform for creating applications based on ConIn. Distributed approaches are capable of providing certain advantages over centralized approaches; for example, liberating locating service portals from DNS, avoidance of central failure, and fewer configurations and errors [15].
trolling the ever-mounting number of Things in a diverse IoT scenario is unrealistic with a flat DHT structure, and hierarchical DHT (H-DHT) is capable of better management than flat DHT [30]. H-DHT reflects the idea of a two-tier architecture. Solutions such as collection, storing and forwarding small data, contextualization, low-level intelligence, reasoning, self-organization, the Publish/Subscribe process etc. can be controlled at the edge, and the cloud can take care of solutions such as large-scale storage, advanced data mining solutions, high-level intelligence, complex processing, and analysis and visualization tools. In light of the above, this study portrays a novel IoT solution capable of providing both edge- and cloud-based solution, thus enabling a Future Internet-of-Things Controller (FITC) that is capable of providing low-level intelligence [Paper V], multi-modal reasoning in order to infer knowledge [Paper VI], clustering algorithms to group similar ConIn [Paper VII], and support for self-organization [Paper IV] at the edge. The enabling of low-latency intelligence, automated decision-making, actionable insights and learning in real-time, and making sense of current data, are some of the challenges that 5G needs to take into consideration in its IoT vision, as stressed by D. M. West in [6]. Mindful of these challenges, FITC has been proposed to counter some of these challenges at the edge.

A future IoT would contain many different connected Things, with different capabilities and communication patterns. Such a complex and diverse IoT would require flexible, dynamic and efficient configuration and management. This vision of autonomic computing aims to mitigate the above challenges by responding to unexpected situations in organizing a system [31]. This should be done with minimal intervention from outside sources, a vision echoed in recent 5G IoT visions [6] which bring self-organization into consideration. Self-organization also implies that a system should evolve correctly in a dynamic environment. Self-organization is driven by self-* capabilities which execute embedded policies that further correspond to an autonomic loop. Some of these policies contribute to integration, discovery, compensation for and correction of failures, optimization etc. within a system. A controller usually handles the self-* capabilities and executes the appropriate policies for each of them. In an IoT scenario, this controller (for example, an edge controller) would control the Things connected to it [Paper IV]. Moreover, controlling Things which may number in the tens of thousands may be impossible via a single controller. To counter this, physically distributed but logically synchronized sinks or controllers are used ([Papers I to IV] use the term sink, and this is used interchangeably with the term controller in this dissertation. Chapter 2 describes the differences). To provide logically synchronized sinks, a PubSub model is required in which each sink publishes and subscribes to the other sinks, thus enabling context acquisition and dissemination [Paper III]. In addition, PubSub enables the dissemination of clustering information. Figure 4 portrays the research motivation and the problem this study undertakes,
where it is demonstrated how increase in the penetration of Things motivates this study.

![Figure 4: Research problem and motivation map](image)

### 1.2 Problem Statement

The penetration of Things in the IoT realm is increasing, and this gives rise to a dynamic environment from which enormous amounts of distributed and heterogeneous ConIn are collected. Since existing solutions are impracticable and inefficient in countering the challenge of managing and utilizing vast ConIn, the provision of solutions that can counter the challenges faced by the IoT is imperative. The grouping of similar ConIn can help in the better utilization of ConIn in the IoT, for example accessing similar automated and grouped remote ConIn by means of clustering. In this regard, the grouping of similar, physically distributed ConIn provides a better way of utilizing ConIn, and this is referred to as logical-clustering in this dissertation. However, this requires many research challenges to be addressed. The first of these is the provision of an architecture that reflects the real-world scenario and which does not cause unnecessary overload for the existing network [Papers I, II]. Logical-clustering also implies that a similarity algorithm that identifies distributed similarity is imperative to the vision of grouping similar IoT data, both for numerical and textual IoT data [Paper VII]. As a consequence of the proliferation of Things, the distributed environment in which the Things interact has become dynamic, and ConIn therefore changes and evolves over time. The next challenge is therefore to provide a mechanism which enables the scalable, dynamic and faster dissemination of ConIn in (near) real-time. A scalable Pub-Sub model is indispensable for these purposes. These challenges can be re-
garded on two fronts in terms of enabling autonomic context information management: (i) the provision of high publication rates and low subscription matching times, and (ii) a scalable PubSub model. Most of the existing PubSub models use centralized approaches, meaning that all publishers publish to a central broker and subscribers express their subscription intentions to this single broker. Similar to the management of connected Things in the IoT realm, PubSub is also considered to be impractical with a single centralized broker. A distributed approach employing several controllers is capable of addressing this issue, where more than one controller can be employed to register subscription interests and store published items. A PubSub model is further required to realize the synchronization of physically distributed controllers to enable DACIM [Paper III]. For example, when a subscriber expresses interest in a logical-cluster of ConIn, publishers from different controllers contribute to the clustering of these similar ConIn, which may be physically distributed. This process of expressing interest, clustering and then the publishers forwarding the clustered ConIn contributes to a certain degree towards the autonomic management of ConIn. This needs to be done with no or minimal intervention from outside sources. Furthermore, the interacting environment can frequently change and become dynamic; this requires awareness of these changes and dynamism, and the Things should be able to organize themselves under these conditions. Thus, the next challenge on which the dissertation focuses is the provision of self-organization to the proposed logical-clustering approach [Paper IV].

Furthermore, management of the enormous amount of ConIn would be infeasible through a centralized controller, due to network load and response time; therefore, this dissertation explores the development of a distributed intelligence-assisted controller that enables faster responses at the edge of the IoT. Moreover, with the ever-increasing number of Things in the IoT landscape, the provision of intelligence by means of rules becomes infeasible and fails to scale well. Rules also fail to provide intelligence in uncertain conditions and can only offer pre-assumed intelligence. This can be countered by incorporating Artificial Intelligence (AI)-based solutions. In the quest for faster responses in the IoT, the faster extraction of knowledge also becomes an important challenge at the edge, for example in real-time decision-making [Paper V]. Reasoning is usually employed in extracting knowledge, for instance in order to reap value from data to make informed decisions. However, a controller, for example an edge controller, is often required to control or manage more than one IoT application, and each of these would require different reasoning to extract efficient knowledge. This gives rise to the challenge of designing and developing a multi-modal context-aware reasoner [Paper VI].
1.3 Research Question

Based on the problem outlined, the research question that is central to the work carried out in this dissertation can be formulated as follows: **how can distributed intelligence-assisted autonomic context information management be realized for the efficient utilization and analysis of vast amount of distributed heterogeneous context information?**

In order to answer this research question, a set of criteria is established. These can be achieved by fulfilling two objectives, which are further divided into four parts; each of these is addressed via a sub-question. The first objective of the research is concerned with enabling a logical-clustering approach (i.e. the utilization of context information) and the second focuses on distributed intelligence (i.e. the analysis of context information). Each of the four parts is summarized below:

I. **What kind of system architecture can enable a logical-clustering approach, and how would it behave in a real-life scenario?** The issue of how physically distributed clustering can be achieved based on similar context, as opposed to physical clustering, can be understood by answering this question. This question is answered by incorporating a two-tier H-DHT model to verify the logical-clustering concept. Jaro-Winkler and Jaccard-like algorithms are extended to find numerical and textual similarity in IoT data, and further development scenarios are outlined for achieving real-time management of ConIn.

The published papers I, II, and VII address this question.

II. **How can the scalable and fast dissemination of clustering identification be facilitated?** The challenge of distributing heterogeneous ConIn is central to this question. The challenges of fast subscription matching and synchronization of sinks/controllers are also addressed under this question. The issue of whether a distributed Publish/Subscribe model can unravel these challenges is further examined.

The question is answered in publication III.

III. **How can the context entities and controllers be organized with minimal intervention from outside sources?** This question investigates methods of organizing connected Things and controllers with embedded policies so that self-organization can be supported in a logical-clustering approach. This would enable the automatic and seamless integration of entities within the system in the case of any failure when...
reconfiguring entities, and the optimization of the objectives of logical-clustering. The answer to this question allows further understanding of how the autonomic management of IoT can be achieved.

This question is addressed in publication IV.

IV. How can intelligence be distributed by extracting and applying knowledge at the edge? The question of the provision of distributed intelligence is answered by building upon the two-tier system architecture proposed in [Paper I]. Computing in IoT is distributed between the edge and cloud; the edge controller provides low-level intelligence while the cloud controller provides high-level intelligence. By leveraging belief networks and learning, similarly to the concept of reinforcement learning for an edge controller, low-level intelligence can be provided. A Multi-Modal Context-Awarereasoner (CAN) enables inferring knowledge for different IoT applications at the edge.

Papers V and VI deal with this question.

1.4 Research Methodology

Research is undertaken which follows a systematic approach in order to establish new facts, which are confirmed by analysing the obtained results. The results necessary to answer the research question, based on the above research problems, are obtained by adhering to a design science research method. To this regard, first of all a question was asked by reviewing existing literatures in the IoT domain where current limitations were explicated with respect to handling growing IoT data. A set of requirements was formulated to design and develop artefacts following agile method to address the explicated limitations. Simulations as experiments approach were employed to demonstrate and evaluate the set of solutions to assist the envisioned approach. Chapter 3 gives details of the research methodology followed in this dissertation, and Figure 5 and 9 also give an idea how this was applied.

1.5 Dissertation Disposition

Figure 5 shows the dissertation roadmap. The dissertation is organized into chapters as follows:

Chapter 1: INTRODUCTION. This chapter gives a description of the study undertaken and research problem based on earlier studies. A solution to the
research problem is also proposed. This chapter also formulates the problem statement and the research question addressed in the accompanying publications.

Chapter 2: BACKGROUND STUDY. This chapter describes the background of existing literature studies corresponding to this research, allowing the subject areas studied in this dissertation to be outlined.

Chapter 3: RESEARCH METHOD. This chapter presents the chosen scientific approach, which also includes the philosophical assumptions made in this study. It further describes how the method was applied within each publication to shape the overall research and thereby the final dissertation.

Chapter 4: DACIM ARCHITECTURE. This describes the architecture proposed for the efficient utilization and analysis of ConIn. The chapter begins by outlining the system model that enables logical-clustering [Paper I]. It also presents an implementation plan and further verifies the correctness of the model using combination of PROMELA and SPIN [Paper II], and describes an algorithm for finding similarity in both numerical and textual IoT data [Paper VII].

A distributed Publish/Subscribe (PubSub) model is presented which can handle the ever-increasing number of heterogeneous context sources. This section extends the Distributed Context eXchange Protocol (DCXP) to provide the proposed scalable PubSub on the MediaSense IoT platform, based on a peer-to-peer (P2P) infrastructure [Paper III].

Supporting self-organization towards autonomic management of IoT outlines the approach to countering the massive amount of participating entities with aim of bringing self-organization. This chapter continues by describing the self-* capabilities that can be applied to an IoT scenario, and illustrates the design and development of those self-* capabilities in order to enable autonomic management [Paper IV].

Distributed intelligence involves the design and development of a novel future IoT controller concept, where intelligence has been distributed and both cloud- and edge-based intelligence are proposed for the IoT domain. This chapter also presents an approach for reducing heavy dependency on rules and increasing response time by leveraging belief networks and reinforcement-like learning at the edge of the IoT [Paper V].

A description is given of a multi-modal context-aware reasoner (MM-CAN) which enables the extraction of knowledge for IoT applications at the edge by employing several reasoning techniques [Paper VI].

The chapter ends by presenting the overall architecture of DACIM.
Chapter 5: EVALUATION of DACIM. This chapter evaluates each of the approaches described in Chapter 4, in order to convey to the research community the novelty and usefulness of these approaches. Different approaches are evaluated to suit the respective evaluations.

Chapter 6: CONCLUSIONS. This chapter concludes the dissertation by reviewing the publications. The research sub-questions posed in Chapter 1 are revisited, and discussions and comparisons with prior work are given where applicable. The most interesting findings are discussed and future work is outlined.

Figure 5: Dissertation road map

1.6 Summary

This chapter first introduces to the research area and the research problem of implication of increase in the penetration of Things in IoT and thereby, the challenges of handling vast amount of IoT data. An insight is given to the research challenges and limits, that the current and future IoT are expected to counter the lack of utilization and analysis of heterogeneous IoT data in the dynamic environments, by reviewing existing research in the IoT domain. Following this, light is shed on the motivation of the study of enabling distributed intelligence-assisted autonomic management of contextualized IoT data. Based on the problem and the motivation, a research question is formulated, for which a design science research method is chosen. The chapter ends by presenting the research roadmap and the structure of this dissertation.
2 Background Study

This chapter highlights existing research, both within and related to the IoT, which provides solutions enabling the underlying vision of this dissertation, i.e. DACIM. Until now, IoT research has focused on creating architecture for connecting Things, and algorithms to provide collected data/context information [15]. This has generally been done by extending a wireless sensor network (WSN) perspective into IoT scenarios. e-SENSE is one such example, and involves organizing context information for dynamic WSNs [32]. This dissertation focuses on utilizing and analysing the context information provided by IoT entities (Things) and enabling DACIM, i.e. autonomic context information management. To this end, the goal of this chapter is to provide insight into earlier research and how each study has contributed to motivating the current research. Furthermore, this chapter corresponds to part of the design science research method, more specifically in terms of an explication of the problem and an outline of the requirements, as mentioned in Chapter 3. The explication of the problem starts with outlining the fundamental problem of managing context information.

2.1 IoT Architectures

The term IoT was first coined by Kevin Ashton in 1999. It was based on the vision of reducing human labour and mitigating human inaccuracy by collecting data from surrounding physical Things via pervasive devices [33]. The main objective behind this was to automate the collection of data. Since then, many solutions have been proposed to address the challenge of collecting data autonomously by designing IoT solutions, mostly via middleware solutions. Each middleware-based IoT architecture addresses different IoT problems, for example device management, ConIn modelling, acquisition and dissemination, context reasoning, context-awareness and interoperability. However, an IoT architecture capable of countering all these IoT challenges has yet to be designed [5]. Most earlier research more or less uses the typical three-layer IoT architecture, as depicted in Figure 2 [5,19,20]. The figure shows a simplified architecture for an IoT application, in which a gateway controls the connectivity of the Things and then collects and forwards data to the cloud for
further processing. The goal of this is to enable computing anytime, anywhere, for anything [5, 34].

**Things** are physical objects that can communicate, that is, can sense and share data. These must be identifiable, meaning that they should have IDs and therefore be addressable. However, the current IoT does not focus solely on physical objects; a virtual or logical object can also be considered a Thing in IoT, thus bringing smart objects into play [5,35]. For example, a tweet (virtual sensor) can be considered sensor data [36] and weather information from a web service (logical sensor) [5]. This also gives an idea of the increase of IoT data, which needs to be handled efficiently and intelligently and on which this dissertation focuses. Several terms for the Things in the IoT exist, for example nodes, connected devices, objects, entities etc., and these are used interchangeably in this dissertation.

**Gateways** lie at the middle of IoT architectures. This concept can be compared to the concept of the sink node in WSNs [5]. Gateways (sinks) have usually been employed to connect Things, to collect data from the connected Things and to forward these to a higher level, e.g. the cloud [5,22,23]. Recently, there has been interest in extending the capabilities of sinks to provide a wider range of functions. This expansion includes, for instance, local storage, low-latency communication, real-time local data processing, and the bridging of several sensor networks [Paper I, 20]. In [Paper V], it was shown how these sinks/gateways can be further employed to provide intelligence at the edge by exploring AI techniques, and such a sink/gateway is henceforth defined as a controller (this dissertation uses the terms sink and controller interchangeably, and these are defined as mentioned above and in Chapter 4).

The **cloud** resides at the top of almost every IoT architecture [3,5,19,20,36,93,94]. IoT has traditionally used cloud computing based solutions for almost all processing and analysing, i.e. computing in the IoT. This choice was obvious, since earlier IoT adoption was generally focused on collecting data in order to reduce reliance on solely human-entered data. For instance, all data collected from SmartFarming are sent to the cloud for storage, processing, analysis and decision making. At times, data are sent to the cloud without prior processing. A temperature sensor may generate new data every few seconds, or the application might be designed so that it forwards data every few seconds, and the amount of data to process therefore becomes massive, giving rise to unnecessary delay and bandwidth requirements. This creates a number of challenges in the IoT such as latency, bandwidth, communication overhead, geographic coverage and analytical dependency. While the advantages offered by cloud computing cannot be ignored, in the rapidly evolving IoT, over-dependency on the cloud simultaneously needs be addressed. This has been echoed in several earlier research studies, in which existing solutions were deemed infeasible and inefficient [3,5,26]. Naturally, the design of an IoT system would need to reduce response time to a minimum...
and avoid over-reliance on the cloud by extending cloud capabilities closer to the Things.

*Edge computing*, a fog computing approach, has therefore attracted attention from IoT researchers [20,28]. There have been recent advancements in powerful and computational capable devices such as smartphones, SmartHome devices, small cellular base stations, connected vehicles and edge controllers [21]. Computation utilizing these devices is referred as edge computing [21]. Even though the terms fog and edge are often used interchangeably, fog computing is a broader term, involving the use of similar cloud capabilities processed at the edge of networks [21,28]. On the other hand, edge computing brings some of the capabilities of the cloud to edge devices [21], which can also contribute to the envisioned 5G device-to-device (D2D) communication for the IoT, bypassing the cloud. Edge computing can offer several advantages, as mentioned earlier in this section, and provides timely and efficient alternatives to cloud computing at the edge. Lately, edge devices are also envisioned to infer new knowledge in order to act based on the IoT data [Paper V,23,94]. A combination of the IoT and edge computing with context-aware computing is assumed to be a key enabler for the future IoT. The next section discusses several existing IoT solutions.

### 2.2 State-of-the-Art IoT Solutions

This sub-section reviews some of the existing IoT solutions to provide an overview of the current state of the art and requirements for future solutions. OpenIoT is one of the more recent open source IoT platforms that enables cloud-based solutions for building and deploying IoT applications [8]. This solution involves semantically interoperable middleware which allows the unification of IoT applications in the cloud in order to deliver services to users based on requests. The objective is to collect and combine data, enabled via a Publish/Subscribe model and wrappers, into a virtual sensor to hide the underlying physical infrastructure from IoT applications in order to make use of data to provide services. This solution was proposed with scalability and performance in mind, for which the cloud has been the prevailing choice. Although this solution proposes the study of the meaning of data from different data sources by adding metadata to the actual data in the cloud, it does not focus on the IoT application itself. This proposal mostly deals with high-level intelligence and the provision of services, and does not explore making sense of data (e.g. for automating decision making and actions) for the actual IoT application. It does however allow access to high-level processing, knowledge discovery and evaluation of data understanding, with perceived cloud-based scalability, at the expense of a delayed response and central point of failure. The proposed middleware utilizes an extended Global Sensor Network
(XGSN) concept which itself is cloud-bound [8,35] and exploits a web-based protocol [37]. Its objective is to collect, forward and filter data; it can be compared to an IoT gateway, as in Figure 2, which lies at the middle of the OpenIoT architecture. It does not address how to use the collected data or what to do with it in terms of providing intelligence for the respective IoT application; however, it does emphasize the fact that raw data needs to be made useful. The solution does this by adding metadata in order to facilitate discovery and searches, by using, for example, an HTTP based-wrapper to allow the cloud to make use of the data. One example of an implementation is CA4IOT (Context Awareness for the Internet of Things) which proposes automation of the task of selecting sensors based on the context to help users [26]. This idea of enabling the communication of combined data to users based on their respective requirements is comparable to the idea of physically distributed clustering; furthermore, this solution advocates the need for contextualization and context reasoning to facilitate user requests and knowledge extraction.

However, the solution proposed by CA4IOT is linked with OpenIoT, meaning that the cloud is again exploited in the solution to make use of data. Another open-source IoT solution is a FIWARE-enabled IoT platform [14], which proposes a cloud-based IoT broker to enable communication with IoT devices and gateways. This solution can be compared to typical IoT solutions, as depicted in Figure 2. The proposed IoT broker is expected to be deployed at the data centre and serves as middleware to enable fast and centralized access to IoT data via a client/server-like approach, employing RESTful API and HTTP. The role of the FIWARE IoT is to provide generic enablers to retrieve and aggregate data from various sources on behalf of IoT applications. The problem of using a client/server approach in the IoT was earlier discussed in [15], necessitating alternative solutions such that undertaken in this study. Central to all of these IoT solutions is an over-reliance on the cloud and/or centralized solutions for making use of data. This is also reflected by a recent study where a survey of existing IoT platforms was conducted [94, 95]. The work also highlighted that while in some of existing solutions, gateway is employed to offer few intelligences but limited to system level intelligence. However, the current and future IoT require making use of data closer to the data origin, that is, at the edge; furthermore, making sense of data is also required to be as close and as fast as possible. The next section describes the focus of the future IoT.

2.3 Towards the Future IoT

The recent surge in mobile/pervasive devices, social networks, and Internet and web-enabled services has brought unprecedented human participation in real-time communication; due to this, it is estimated that hundreds of billions
of Things will be deployed. In addition, the recent advancements in pervasive technologies mentioned in the previous chapter have also contributed to the blossoming of the IoT. This development was previously predicted by Zam-bonelli in [34]; he envisioned that future pervasive computing would be driven by distributed collaboration, known as crowdsourcing, which would enrich urban networks via spontaneous human participation. Boulos et al. described this as human-in-the-loop-sensing [38], and Sheth describes it as citizen sensor networks [39]. This integration of people, services, pervasive devices, surrounding Things and context information will shape the future IoT, leading to a paradigm shift towards the Internet of Everything (IoE).

This paradigm shift will contribute to the evolution of an ultimately connected society in which an enormous amount of heterogeneous IoT data is generated from billions of Things. While the collection and sharing of these IoT data was once the challenge and goal of the IoT, as reflected in the current IoT solutions described in the previous sections, the challenge and goal lie today in making sense of data. This involves far greater challenges and value in the IoT, in this era of the connected society and for the future. Context-aware computing promises to play a crucial role in reaping value from collected data in the future IoT [5,7,10,26]. Earlier context-centric approaches have been used to enable context-information provisioning, creating context-driven applications and providing automatic connectivity between Things irrespective of their physical location [15,24]. However, none of the earlier context-based solutions discusses the management of the ever-growing volume of context information. Efficient and intelligent management of this influx of context information requires the utilization and analysis of context information. Clustering is one technique which can aid in minimizing the volume of context information [4,15,40], thus enabling better utilization; reasoning (that is, inferring knowledge to reap value) is another analysis technique [5,10] providing intelligence.

2.4 Clustering

The huge influx of context information in the future IoT corresponds not only to the sensing and analysis of any situation, anywhere and anytime [34], but also includes smart objects. For example, data from a Twitter feed can be considered as sensory data [36]. Data clustering is a popular technique and has been widely used for over half a century. The purpose of clustering is to explore data efficiently by grouping data that demonstrate similar characteristics and which are dissimilar from other groups of data [4]. According to Jain, such a grouping involves fundamental modes of understanding and learning [41]. Clustering is also used to analyse and organize the data.
Data analysis in IoT is still in its infancy [42]. Clustering in the IoT has not been extensively studied until now. It has been a popular technique for exploring data and is widely used in WSNs; however, most of these studies are centred around grouping sensors in order to increase network lifetime and to decrease energy dissemination. The first study of clustering protocols in sensor networks was conducted by Heinzelman et al. [43], in which the authors proposed the LEACH routing protocol. The goal was primarily to achieve network longevity and decrease energy dissemination. Other studies have either proposed new protocols centred on this or have further modified the LEACH protocol to achieve the same goals. The aggregation of data is also one of the usages of clustering, as described in [44-46]. However, this approach also aims to reduce energy consumption and increase the scalability and robustness of networks. The distributed processing of context for dynamic WSNs was presented by Lombriser et al. [32]; these authors proposed e-SENSE, which computes context information from sensor networks. In this approach, sensors are clustered according to similar context activity, restricted to adjacent sensors; however, e-SENSE does not solve large-scale sensor network issues. A concept known as the logical neighbourhood of sensor nodes, a substitute for the concept of the physical neighbourhood, was proposed by Mottola and Picco in [47]. However, the limitation of this solution is that it makes use of a programming language abstraction where nodes are logically assumed to be in the same neighbourhood if certain attributes are fulfilled. A system administrator usually defines the attributes of nodes and the data segment that can be part of a neighbourhood. This therefore fails to explicitly address the issue of real-time distributed context information distribution. The need for a real-time analysis of sensor data in the IoT by means of clustering was highlighted by Hromic et al. [42]. They proposed clustering data by employing an information server in the cloud environment, which groups data based on k-means clustering. However, it is unclear how this clustering helps in utilizing IoT data. Furthermore, the paper’s contribution or proposal for clustering appears to lie in data acquisition and grouping data into k-numbers which is pre-determined. Clustering for massive small IoT data was addressed in [48] to improve the utilization of system resources and efficiency of data processing in terms of execution time and memory usage. This work addresses the issue by dividing large datasets into small ones using k-means, which again is pre-determined.

The requirements of distributed clustering are not met by the aforementioned solutions, and there is no existing approach that is capable of solving the large-scale data utilization issue. Thus, there is a clear mandate for a solution to the large-scale distributed management of ConIn. This necessitates research into alternative approaches such as logical-clustering to mitigate the issue.
2.5 IoT Autonomic Computing

The properties of the future IoT (i.e. the IoE) correspond to a cycle, as illustrated in Figure 3. This figure is another version of Figure 6. Context information generated from the surrounding Things can be returned to people, pervasive devices and services. This cycle resembles the idea of an autonomic computing loop, and is also echoed in [95], implying that a controller executes policies embedded by an outsider, for example a system administrator. The goal of enabling autonomic computing is to ensure that the IoT system evolves correctly in dynamic and unpredictable situations, and structures itself in an organized way. Adaptation to fast-changing environments and being aware of changes is one of the challenges of the IoT. Stabilization of the system in uncertain situations and organization of the system so that it responds to dynamic conditions are crucial [3]. For example, network connectivity, bandwidth, insertion and deletion of information, and the joining and leaving of a node/device (i.e. Thing) are some of the dynamic aspects expected of the IoT [3,22]. Therefore, it is imperative that a system is capable of responding to such situations with minimal or no (current challenge) intervention from outside sources. To achieve this, an autonomous system requires several self-* capabilities. The future Internet will require autonomic computing, and this is one of the challenges of the future Internet [49] and the vision of 5G IoT [6]; the future Internet will be driven by the IoT. Therefore, the IoT also requires autonomic computing, which will allow it to evolve correctly. The clustering of context information can also be compared with an autonomic process. For example, when a researcher or system administrator expresses an interest in a particular cluster (one of the use cases of logical-clustering), the system should be able to cluster the relevant information (both locally and globally) and forward this clustering information to the requestors without their intervention, thus automating the process. The Publish/Subscribe method comes into play in this type of autonomic process.
2.6 Context-Aware Computing in the IoT

The collected raw data does not usually provide any usefulness [12]. To add value, the raw data need to be contextualized [5,26, Paper X], which aids in understanding the current situation in terms of the collected raw data. A system that uses and responds to the context in order to provide relevant information and/or services is defined as context-aware [7]. In an IoT scenario, an IoT application is said to be context-aware if the application makes use of context to add value to raw data and to provide services based on this contextualized raw data, for example the filtering of collected data before forwarding it to the cloud, making decisions, extracting knowledge, context information retrieval and automation of tasks. With this in mind, Kanter et al. [24] proposed and developed a context-based protocol known as Distributed Context eXchange Protocol (DCXP). DCXP aims to provide real-time context information between Things. This protocol was further employed to create an open source IoT platform, MediaSense, which enables the creation of interactive context-aware IoT applications [15,24]. This research originated within distributed context networks, with the aim of providing scalable and real-time solutions in the IoT. Such solutions aim to analyse, extract knowledge from and make sense of raw data collected from the IoT. In this regard, context-aware computing is expected to play a vital role in the IoT field, as it did earlier in mobile and pervasive computing [5,10,15,25,26,50].

Context-aware computing is expected to enable smart city applications and services [Paper X, 51]. Furthermore, context plays an important role in enabling autonomic computing [7], which can also be seen from the context cycle illustrated in Figure 1 and shown in [5,16,17]. Until now, cloud computing has been the prevailing choice for enabling context-aware computing in the IoT [5,51], and rules have been the most widely used method of extracting knowledge, i.e., reasoning [5]. New approaches are required to reduce the burden on cloud computing, and rules fail to scale with the ever-increasing amount of context information. Context-aware IoT computing therefore needs to respond faster to the context; contextualization is also required to be done closer to Things rather than in the cloud, and new reasoning techniques are required to handle the high volume of context information. The proposal of FITC can address these challenges, as demonstrated in [Papers V and VI].

2.7 IoT Intelligence

Intelligence is defined as the application of knowledge, and involves several steps to obtain knowledge. This can be seen in Figure 7, reproduced from the well-known information-knowledge hierarchy. The application of knowledge also corresponds to the question of *how*. This knowledge is extracted from the
collected data; between these two steps lies the information step, which is obtained by answering questions about data, such as those shown in Figure 7. In an IoT scenario, these steps can be compared with collecting raw data from Things; answering questions to find information is similar to contextualization. This contextualized data is then applied in order to infer knowledge [5, Paper V]. Research has until now focused on providing this intelligence in the cloud, while recent parallel approaches such as that in [23] also demonstrate system-level intelligence in the IoT while highlighting shortcomings in existing research into intelligence in IoT. It is demonstrated that existing intelligence is only limited to interoperability between Things, reconfigurability, or specific to a particular IoT application. The authors illustrate intelligence by enhancing data fusion, aggregation and interpretation in terms of local data processing and storage, notification etc. However, intelligence based on the collected data is limited to predefined rules at the edge [23, Paper V] and the cloud is consulted to define new rules; this is time-consuming. To address the current challenge, which is to act on the current data in real-time, intelligence needs to be realized at the edge, and should not rely solely on rules, due to the scalability issues pointed out earlier. Rules also fail in uncertain situations, for which AI-based techniques such as Bayesian approaches, hidden Markov models, artificial neural networks, support vector machines etc. can be considered [5].

![Figure 7: Information-knowledge pyramid hierarchy (adapted from [92] and [93])](image)

Learning plays an important role in providing intelligence. Most machine learning techniques are heavily dependent on historical data is required in order to learn from these data [12]. Recently, Bayesian belief network-based learning has been proposed to allow prediction, learning, decision making, etc. The reason that this technique is receiving attention is that it is both conceptually simple and effective in learning through experience. Reinforcement learning, a machine learning technique, allows an agent to learn from experience. This learning is very helpful in a situation where an agent, learner or decision maker (for example, a controller) can learn by giving rewards for each perceived action from any environment such as an IoT application [52, Paper V]. Both low-level and high-level learning can use this kind of approach. One of the main challenges within IoT with regard to intelligence is that none of the current approaches attempt to improve the intelligence by
making sense of current data. In view of future IoT requirements and current shortcomings, research towards supporting IoT with intelligence, overcoming dependency on rules, and learning from the current IoT data at the edge needs to be undertaken.

2.8 Enabling the Dissemination and Management of Distributed Context Information

The large-scale collection of dynamic context information is expected to contribute towards the future IoT [15]. This context information is also required scalable and flexible dissemination among the Things. The IoT also requires support for managing massive numbers of Things, and this has mostly been explored using centralized approaches or broker-based architectures [5,15]. Centralization approaches specify a single platform for data aggregation and distribution, but give rise to concerns over scalability in terms of large-scale entity management. Other approaches have tried to counteract for this challenge, but have again failed to promise real-time context information provisioning [15]. These solutions fail to counter the challenges that everywhere computing poses, as summarized by Hadim and Mohamed [53]. Moreover, centralized web service portals mostly employ a Domain Name System (DNS) [15]. This approach involves challenges such as the availability of DNS, vulnerability to Denial of Service (DoS) attacks, configuration errors, etc. In search of alternative solutions, Pappas et al. [54] proposed a Distributed Hash Table (DHT) overlay-based approach in order to support this enormous number of Things. DHTs offer certain advantages such as countering the centralization-dependency issue, providing scalability, and dynamic self-organization issues [15]. A dynamic, self-organizing and open IoT can be built upon DHTs [15]. A blend of DHT and WSN was first realized by Fersi et al. [55], and this allows efficient management of location-agnostic data and node identification; Walters [15] further demonstrated the application of DHTs in the IoT domain to realize a dynamic and decentralized IoT for real-time communication.

DHTs can be further divided into flat and hierarchical DHTs (H-DHTs). Entities (Things) are organized in layers in DHTs. This division of entities into tiers aids in the efficient organization of the ever-increasing number of entities; furthermore, H-DHTs provide better fault separation, more effective bandwidth utilization, better adaptation to the underlying physical network and a reduction of the lookup path length [30]. Furthermore, Zoels et al. [30] illustrated that two-tier H-DHTs can be used as an optimal alternative approach for the efficient management of cost-effective entities. Two-tier H-DHTs divide the Things into super-entities and regular entities. A two-tier H-
DHT therefore controls the super-entities on the top tier, and the bottom tier controls other regular entities. Various DHT approaches such as Chord, Pastry, P-Grid etc. can be explored in each of the tiers [30]. The benefits offered by H-DHTs can be employed in order to realize a distributed context information management approach (see Chapter 4 for details). The scalability and flexibility essential for the IoT can be obtained by DHT-based approaches.

RSS or ATOM feeds are typically used to distribute news or event notifications on the Internet, while Publish/Subscribe (PubSub) systems are perceived as enabling the sharing of distributed context information [56]. Fabret et al. [57] created an extremely fast web-based PubSub model, Le Subscribe, with dynamic web content in mind. Their proposal was mainly driven by the fact that most of the data would be on the web in future. Although the PubSub model accommodates dynamic web content, it does not satisfy distributed requirements. Moreover, mobility requirements need to be met, and this aspect was investigated by Zarko et al. [58]. Their proposed PubSub model explored real-time data delivery and energy saving in mobile crowdsensing. The feasibility of scalability and mobility was further explored in [59,60] with the ToPSS PubSub model; however, this model does not satisfy the heterogeneity issue. A large memory is also required by ToPSS to store the event notifications, which is a shortcoming in resource-constrained devices such as the Raspberry Pi. Figure 8 displays a typical illustration of a PubSub model.

![Figure 8: Typical PubSub model (adapted from [56])](image)

PubSub models can be divided into two types: (1) topic-based and (2) content-based. Subscribers to a topic-based model cannot choose the type of events that are of interest to them; all topics connected to a subject are notified to a subscriber. In content-based systems, subscribers have the freedom to choose the events of interest, thus enjoying more flexibility and usefulness than the topic-based model can offer. Tootoonchian et al. [61] exploited this PubSub model to enable physically distributed but logically synchronized controllers for OpenFlow, in a model known as HyperFlow.

PubSub models can also be employed in context dissemination automation. For example, when a subscriber expresses an interest in a topic or content, the event dispatcher registers this interest, and whenever context information related to the interest is available, the event dispatcher forwards it to the subscriber without the subscriber’s intervention. This provides a certain degree of automation.
2.9 Summary

In order to realize distributed intelligence-assisted automatic context information management (DACIM), a fusion of architectures, methods and algorithms is required. This chapter has described the background to and related work in efficient and intelligent context information management for the future IoT, starting from context modelling (corresponding to the contextualization and representation of IoT raw data), ConIn acquisition and dissemination (relating to ConIn distribution) and context handling by means of low-level intelligence, where clustering and reasoning are employed. With respect to an evident paradigm shift leading to the IoE and the challenges associated with this shift, the need for the design of a new system was discussed. This chapter then examined the literature on the utilization of context information by means of clustering, followed by the autonomic management of context information in the IoT. To comply with current and future IoT requirements, the need for faster response and intelligent context information analysis by means of intelligence closer to the Things was discussed. This chapter also illustrated an approach to the organization of dynamic entities (Things) with minimal outside help, i.e. the scope of the autonomic management of entities was examined. The chapter concluded by describing the efficient dissemination and acquisition of context information by means of a PubSub model. Issues such as the development of an architecture to realize DACIM, its probable performance in real-world scenarios, the dissemination of distributed information, the development of mechanisms to support the correct evolution of massive numbers of dynamic Things, algorithms to provide distributed intelligence at the edge, reasoning based on current IoT data and the grouping of similar context information were also discussed. Chapter 4 examines the aforementioned issues, and Chapter 5 describes performance metrics for these issues.
3 Research Method

This chapter introduces the methodological and philosophical assumptions underpinning the research and experimental methods used here. This chapter first presents the philosophical assumptions made in this study, followed by the research methods applied in this work. The chapter ends by discussing the ethical issues concerned with this particular study.

3.1 Philosophical Assumptions

Scientific inquiry is composed of a collection of philosophical assumptions that are employed by a research community. In the quest for the knowledge needed to ultimately answer the research question, this study makes inquiries based on philosophical assumptions. These assumptions involve the existing reality, how it should evolve and how it needs to be seen and interpreted, thus constituting the ontological nature of research. In order to understand the existing theory and to become an insider in the reality that is being studied, observations are used to obtain knowledge which correspond to epistemological assumptions. Following this, the axiology factor is addressed by reporting quantitative values via testing; this might involve certain biases, as the simulated hypothesis model, often considered to be perfect, does not always reflect real-life scenarios. Finally, the methodological aspect is addressed by applying methods to acquire knowledge and to report the outcome of the research. These philosophical beliefs guide the researcher in terms of the methods selected and applied; logical reasoning is used to deduce the implications of this research for the community [62]. Furthermore, the philosophical assumptions that constitute research within computer science are generally considered to be either Positivist or Interpretivist [63].

Positivism is the philosophical view that there exists a single reality in human minds, and that this is ontologically common for every observation. Epistemologically, knowledge about this single reality is objectively gained via observations and experimentations. Hypothesis testing and prediction of values relate to axiology, while methodology concerns the approach by which experiments are employed to attain this knowledge, through analysis and observation of the experiments.
This research is based on positivism, and aims to develop a model that will allow the management of heterogeneous distributed ConIn in the IoT domain. Research within IoT mostly relies on verifying hypotheses based on existing theory and tested empirically. Hypotheses regarding the creation of a distributed intelligence-assisted context-based approach which aims to aid in the autonomous management of IoT, as undertaken in this research, are tested empirically through experiments. Experimental research allows observations to be made in an environment which can be controlled, in order to verify the designed and developed model based on certain variables. The development of such a model therefore begins by understanding the existing problem within the IoT domain, through reviewing the existing theory. This research was undertaken with objectivity to propose and test hypotheses about certain known or unknown aspects, for example the network performance of the model and different performance metrics for the algorithms. The model itself is a combination of several other artefacts. Each of these artefacts is tested empirically by observing its performance. In order to observe this performance, an empirical model is designed, developed and evaluated, which involves the creation of algorithms and the development of software methods and simulation models. Experiments are carried out using these algorithms, software methods and simulation models in order to observe improvements and performance; this offers evidence for accepting or rejecting the hypothesis and/or for determining the contribution of the current research, thereby verifying and generalizing the research results to the single objective reality. The performance is measured using a standard approach and/or earlier similar approaches; this generally follows a formula which tries to verify whether a model is more effective than another model by comparing the improvement in terms of a percentage for various performance metrics relative to the specific artefact. This kind of deductive performance measurement, i.e. the deduction of a quantitative result, helps to verify whether or not a model is effective (that is, whether the model is to be accepted or rejected), with a certain degree of confidence.

3.2 Methodological Overview

The Oxford English Dictionary defines research\(^1\) as:

*The systematic investigation into and study of materials and sources in order to establish facts and reach new conclusions.*

\(^1\)http://www.oxforddictionaries.com/definition/english/research
This definition consists of four parts: (i) systematic investigation; (ii) study of sources; (iii) establishment of facts; and (iv) reaching new conclusions. According to Logan, much of this systematic investigation is about getting to know the unknowns [64] by following research methods; Demeyer [65] states that research in computer science involves studying artefacts that are designed by humans. A practical problem which humans perceive is addressed by an artefact [63]. Humans study the sources of problems, try to reach new conclusions about the problem by establishing the facts, and finally design artefacts to address the problems. An artefact may be, for example, a physical object e.g. a hammer, a car or a software system such as logic for designing databases, algorithms, design guidelines etc. [63,66]. One strand of design research is design science, which as a research method attempts to create and realize artefacts; it is similar to the approach used in the hypothetico-deductive scientific method, through which research problems can be solved [63].

Design science seeks to develop working solutions that are able to produce and communicate new knowledge. This process follows a series of steps and iterations, as illustrated in Figure 9 (adapted from [63]), from explication of the problem to the definition of requirements and the evaluation of the designed and developed artefact. The order of the design science steps resembles the sequential waterfall model, where each step is dependent on output from the previous step(s); however, it essentially follows an iterative and agile approach to developing a solution. It can be seen in terms of dividing the process into phases and disciplines, where in each phase almost all disciplines are consulted. The current study also follows this approach. One alternative to the design science research method which might have been considered is the constructive research approach (CRA), which tries to bridge the gap between academia and industry [67]. This method involves the following steps: (i) selecting a practically relevant problem; (ii) obtaining a comprehensive understanding of the study area; (iii) designing one or more applicable solutions to the problem; (iv) demonstrating the feasibility of the solution; (v) linking the results back to the theory and demonstrating their practical contribution; and (vi) examining the generalizability [67]. These steps correspond to the steps in the design science research (DSR) method, except for step (v), which involves consulting the theory and demonstrating the solution’s novelty and usefulness. This particular step is important in answering any research question where the researcher needs to verify whether the results correspond to the identified requirements and whether it provides any usefulness in practice, thus bringing the iterative approach into consideration, which is also advocated by DSR. Moreover, steps (iv) to (vi) of the CRA essentially correspond to the last two steps of the DSR. This shows that there is little difference between the two methods; however, the research question of this study is based on positivism and can be answered via deductive reasoning. DSR is able to apply positivism and deductive reasoning to answer a research question [63] while CRA is
aligned with pragmatism and abductive reasoning [67,68]. Furthermore, constructivism brings personal values into a study via the interpretation of data, while positivism, which is supported by DSR, uses standards of reliability and validity through observing and measuring information numerically [69]. This study aims to develop a model, that is, an artefact, which can be evaluated based on quantified data with respect to existing solutions and to whether, and how, the developed model provides any usefulness and novelty. The final artefact to be developed is itself a combination of different sub-artefacts that are defined in the requirements. Each of the developed sub-artefacts applies deductive reasoning, and is demonstrated and evaluated to determine whether it can be accepted or rejected with a certain degree of confidence by comparing with existing solutions or standards whenever applicable.

![Diagram](image.png)

Figure 9: Applying the design science research method [63]

To comply with the requirements of positivism and deductive reasoning, the reporting strategy of the research results is predominantly quantitative. DSR provides the flexibility to choose any suitable research strategy to validate a research question. The research objective in this study needs to be validated by evaluating the artefact, and experiments are well suited to this. The designed and developed artefact is therefore investigated by employing simulations as an experimental approach, as outlined in Sections 3.2.4 and 3.2.5. The performance of the resultant artefact is verified using several tools such as a network simulator tool called ns-3, and a scalable and versatile IoT platform called MediaSense. More specifically, the network performance is verified using ns-3 [Paper II] and the performance and/or feasibility of other artefacts (e.g. the Publish/Subscribe model, self-organizing algorithms, clustering algorithms etc.) are verified using MediaSense by extending the current version [Papers III, IV, VII].
3.2.1 Explication of Problems

The first step in DSR is to explicate the research problem; this is similar to that of any scientific research, that is, asking questions. This has been carried out predominantly in the Introduction and Background sections, and the problem is subsequently detailed in each of the accompanying publications. A literature study of the existing solutions for managing and utilizing ConIn and clustering is undertaken, and further current solutions related to the increase in ConIn within the IoT are documented in order to highlight the gap between current solutions and future challenges. A system based on current standards, tests and results is found for building the artefact. More specifically, an architecture for managing ConIn is demonstrated in [Paper I]. As the number of Things in the IoT is experiencing an unprecedented rise, centralized approaches fail to manage these ever-proliferating Things, as earlier research has also shown. A distributed approach, such as a distributed hash table (DHT)-based approach has been proposed to address this challenge. However, the expected number of hundreds of billions of Things gives rise to further challenges for flat DHTs to meet, and scalability becomes an important issue. It would be challenging for a flat DHT-based system to provide better scalability in the IoT. Cloud computing has until now been the prevailing choice for handling computing in the IoT; however, the IoT requires real-time computing with minimal latency. The IoT consists of different IoT applications, including SmartHome, SmartHealth, SmartFarming/SmartAgriculture, SmartParking, SmartRetail etc., and these applications require rapid decisions in order to act on based on collected raw data. For example, when sensors collect raw data from a SmartFarm or a SmartHome, and an actuation is needed based on the sensory data, waiting for a decision on when and how to perform the actuation from the cloud can be time-consuming. It is more suitable in the era of real-time computing to make decisions as quickly as possible, closer to the Things, i.e. at the edge. Solutions which can provide real-time computing with low latency in the IoT are therefore necessary, and this has been a challenge [Papers I, V]. Once a model is designed to meet the above challenges, the model needs to be tested. The testing also corresponds to the reliability check [Paper II].

Following this, various sub-artefacts are combined to form the overall model developed in this research, as shown in Figure 10. Another challenging issue in the IoT is to share ConIn with a high published rate and low subscription-matching time. A scalable Publish/Subscribe model is necessary for such purposes. There are challenges on two fronts: (i) the achievement of a high publication rate and low subscription matching time; and (ii) a scalable Publish/Subscribe model to synchronize several edge controllers. Most of the existing Publish/Subscribe models are centralized, meaning that all publishers publish to a central broker and subscribers express their subscription inten-
tions to this single broker. As for the management of Things in the IoT, Publish/Subscribe is also infeasible for a single centralized broker [Paper III]. The Things in IoT applications interact within a rapidly changing environment, and in each application need to be aware and respond to these changes with minimal intervention from outside sources. Therefore, each Thing in the IoT needs to be self-organized [Paper IV]. Once the Things are organized, with minimal intervention from outside sources, they need to act on the collected raw data. Raw data does not generally provide any insight unless contextualized or processed. Again, cloud computing has been the primary choice for processing, providing insight into and reasoning with raw data. Intelligence is also provided via the cloud. Edge computing has recently become a popular research topic, especially with regard to incorporation in the IoT. It is envisioned that edge computing and context-aware computing will be the key enablers for the IoT. Solutions for processing raw data, providing intelligence and clustering based on the collected raw data are some of the current challenges at the edge of the IoT [Papers V, VI, VII].

![Figure 10: Artefacts for the DACIM](image)

### 3.2.2 Definition of Requirements

The requirements for the artefact, which further explicate the research problem, are a set of approaches, methods, algorithms, and relationships for the identified problem in the environment, that is, the IoT. Various requirements are further highlighted in the accompanying publications. The first requirement is an architecture that helps to realize the logical-clustering approach and distributed intelligence, and thus the DACIM. As mentioned in the explication of the problem, both centralized and flat DHT-based approaches would be infeasible in the IoT; therefore, alternative approaches such as a hierarchical DHT-based solution offer suitable alternatives. To comply with the low-latency and real-time computing requirements of the IoT, computing should be done as close to the Things as possible, that is, at the edge of the IoT. However, bringing full cloud computing capabilities closer to the Things may not be feasible due to the high requirements for computational capabilities. Hence, a
solution is required that is capable of dividing and delegating computational tasks from cloud computing to the edge, for which a software-defined approach such as an OpenFlow-based approach is required [Paper I]. The next challenge is to share the clustering identification and ConIn with the distributed context entities (Things) using a Publish/Subscribe model. To address these problems requires a scalable Publish/Subscribe model, and new algorithms are essential to enable such a model. Existing solutions such as Le Subscribe, PARDES [95] and ToPSS offer PubSub models; however, most of these are centralized and/or slow in terms of publishing rates, and/or require high subscription matching times. To overcome these shortcomings and to enable a distributed approach, new algorithms are needed based on a distributed protocol [Paper III].

To integrate self-organization with IoT, it is necessary to propose, design, and develop new algorithms, which are then tested in terms of their effectiveness based on the standard evaluation. Autonomic computing, i.e. self-organization, corresponds to a loop which executes policies embedded into the self-* algorithms. Each IoT application may have different policies; however, there are certain policies that are common to the achievement of self-configuration, self-optimization, self-healing, self-protection, etc. for the Things. A controller based on cloud and/or edge computing can execute the self-* algorithms to enable such self-organization [Paper IV]. The policies that control the Things in the IoT realm are mostly rules-based, both in the cloud and at the edge. Rules fail to scale well with an increase in the number of Things, and a new approach to address this is required by employing artificial intelligence techniques. A controller capable of providing intelligence both in the cloud and at the edge is therefore essential. This distributed intelligence-assisted IoT controller would execute the self-* algorithms and provide intelligence to enable autonomic management of the IoT with respect to the Things and its ConIn. The controller would provide intelligence based on the processed raw data, that is, ConIn [Paper V]; furthermore, the controller is required to infer knowledge before applying it, for which a context-aware reasoner is necessary. The reasoner also needs to handle several IoT applications, meaning that a multi-modal solution involving a reasoning technique is vital at the IoT edge [Paper VI]. Grouping similar ConIn, which may be stored remotely, requires the use of a similarity calculation algorithm. IoT data consists of both textual and numeric data, and an algorithm needs to be able to handle both types. Existing similarity algorithms such as Jaro-Winkler and Jaccard-like algorithms are examined and extended to find a distributed clustering algorithm which can work on the proposed distributed controller, thus enabling clustering on a logical basis as opposed to a physical one [Paper VII].
3.2.3 Designing and Developing the Artefact

This stage of DSR is based on the earlier two activities, and involves the design and development of an artefact to address the research challenge. The study presented in this dissertation is based on a compilation of research papers; each paper focuses on the design and development of artefacts corresponding to the problems set out in the problem explication section and the further requirements described in the previous section. The final artefact is a summation of the other sub-artefacts, as shown in Figure 9, and the further division of this artefact is illustrated in Figure 10. Following this, an agile development for each of the artefacts was carried out over several iterations. This iterative approach helped to improve the functionalities of each artefact, and earlier steps were consulted to ensure that the developed sub-artefact was aligned with both the specific problem and the defined requirements. This also involved tracking the overall artefact. Mindful of the requirements described in [Papers I and II] and the earlier sections, this thesis first presents the design of a system model that enables the DACIM. More specifically, this is a two-tier hierarchical DHT (H-DHT)-based architecture, in which edge computing is employed at the bottom tier and the upper tier corresponds to cloud computing. Scalable MediaSense is utilized as the DHT-based system. The system model is verified with a versatile and useful combination of PROMELA and SPIN. This combination was explored earlier for the simulation and verification of the system model in [70-72]; it offers versatility and is very valuable for model checking, used extensively for modelling and verifying communication protocols [73]. Following this, the model is verified in terms of the reliability of its performance. Software called network simulator 3 (ns-3) is used to verify the research model with realistic real-time performance. Three different wireless networks are employed, where each network is controlled by a gateway and OpenFlow is used to manage these sensor networks. To enable the flow of data between clustered sensor nodes, multicasting support for wireless networks within ns-3 is developed, and similar sensor nodes are clustered on a logical basis. Following this stage, in order to design a scalable Publish/Subscribe model and self-organized algorithms, DCXP is extended, which ensures faster responses. Then, several algorithms are developed for publishing items, subscription matching, self-configuration, self-optimization, self-healing etc. as outlined in [Papers III and IV].

Next, a distributed intelligence-assisted IoT controller is designed by proposing new algorithms to delegate some of the cloud computing tasks to the edge computing. Algorithms are designed based on the knowledge pyramid, where low-level intelligence is carried out at the edge, and high-level intelligence is implemented in the cloud. Furthermore, Bayesian-based reasoning, that is, a belief network, is proposed and developed at the edge to reduce dependence solely on a rules-based approach [Paper V]. The IoT domain involves various different IoT applications and data types; a reasoning model
that can provide multi-modal reasoning for such IoT applications is therefore designed and developed [Paper VI], and a clustering algorithm is designed by extending the Jaro-Winkler and Jaccard-like algorithms to cluster different IoT data types [Paper VII]. An iterative approach is employed to design, develop and demonstrate the abovementioned algorithms.

3.2.4 Demonstration of the Artefact

The purpose of this demonstration is to convey to the audience the novelty and usefulness of the study. The demonstration of the designed artefact started with a verification performance analysis using MATLAB and ns-3 simulations. Real-world experiments are often expensive and time-consuming [65, 74], and ns-3 can aid researchers in testing a research model, giving realistic real-time results. The reliability of the designed ns-3 model is tested and demonstrated for several different scenarios [Paper II]. Reliability involves the delay, jitter, and packet loss ratio performance of the developed model; the developed model should not introduce unnecessary delay or jitter. Its performance is demonstrated for several scenarios using varying numbers of nodes per cluster and varying rate of data flow. A physically distributed clustering approach is implemented in MATLAB to demonstrate the computational efficiency of the designed model [Paper I]. Following this, the Publish/Subscribe model is verified; this is usually evaluated by measuring the published items, subscription matching and Publish/Subscribe messages per unit time. Although research into the IoT has progressed substantially since it was first introduced in 1999, the IoT is still regarded as being in its infancy due to the lack of large-scale implementation; however, this is expected to change within the next few years. This means that there is a lack of both data for the IoT and implementation scenarios that can be consulted to extract data. Simulated data is therefore used to test and evaluate the remaining algorithms and models. The Publish/Subscribe model is evaluated using the performance metrics mentioned above, and this is further discussed in [Paper III]. These metrics are compared with earlier approaches and the effectiveness is confirmed. Self-* algorithms are demonstrated for various performance metrics such as the joining rate, discovery rate, discovery accuracy etc. for different numbers of Things. Things are randomly generated on the MediaSense platform and varying numbers of Things are used in each simulation. The simulated Things or data do not affect the performance of the demonstration. The reason for this is that this research is concerned with managing the data rather than how to collect data; however, the type of data is important, and papers related to earlier IoT applications such as SmartHome, SmartFarm and SmartHealth are taken as references and similar data is simulated in order to carry out the experiments by consulting relevant sources, as mentioned in [Paper VII].
The way in which the Things in the IoT are connected, for example to the edge gateway (i.e. via IoT controller), is beyond the scope of this study. The collected raw data is contextualized at the edge rather than in the cloud, and the contextualized data (ConIn) is then utilized to provide low-level intelligence by reasoning. Reasoning is evaluated by calculating how quickly it responds, i.e. the latency is measured. Furthermore, reasoning scalability is demonstrated by showing how the dependency on rules can be reduced. This dissertation also evaluates the clustering performance for various scenarios such as clustering size, and similarity percentages for different numbers of controllers. The developed artefacts are compared using various simulation scenarios in order to verify the correctness and feasibility of each artefact and in order to generalize the results to reflect probable real-world scenarios. This comparison helps to determine whether the developed artefact contributes to the research community and can solve real-world problems.

3.2.5 Evaluation of the Artefact

The experimental setup is based on earlier studies, as described in the previous section and in [Papers I-VII] for the respective artefact(s); these are carried out during the demonstration stage to evaluate the model against quantitative data, generated using simulations as experiments [74,75]. Simulations generate new data for empirical systems [75]; this is also a type of experiment and allows scientific inquiry into a computer-based model to be carried out more quickly and with lower cost [74]. For example, if temperature sensor values on a farm need to be observed, it may take a very long time to gather data for a particular season; however, it is possible to simulate the expected temperature values on the farm (a sensor usually reports temperature values with a maximum and minimum value already suggested) and to validate the developed model based on these simulated values. This gives new data about the developed model for the probable scenarios for which the model was developed.

This evaluation is reported using a simple formula:

Model \( M_1 \) will perform better compared to model \( M_2 \) based on the performance metrics \( P_m \) given the \( x \% \) of improvement in \( P_m \).

To calculate the \( x \% \) improvement, the following formula is used:

\[
x \% = \frac{V_1 - V_2}{V_2} \times 100 \% \quad \{ V_1 > V_2 \}
\]

(1)

\( V_1 \) and \( V_2 \) are the observed values from models \( M_1 \) and \( M_2 \) respectively for performance metric \( P_m \). To observe how an artefact behaves in different sce-
narios, several tests of the developed artefacts are carried out in different sce-
narios, thus giving data to measure the performance. Mean values are reported
over several simulations for each test scenario, and/or standard deviation are
measured to evaluate the artefact whenever applicable. Performance measure-
ments are compared with the standard approaches if applicable, or with similar
approaches, to confirm their usefulness and novelty. The evaluation of the ar-
tefact does not rely on a single, unique platform. Furthermore, different ap-
proaches are evaluated in the research outcome by means of publications.
Each publication describes the evaluation of each approach, and compares this
with similar past and current approaches whenever applicable. These compar-
isons and the other results allow the observation of evidence about any im-
provement or standard behaviour, which allows conclusions about the devel-
oped model to be deduced.

3.2.6 Research Ethics

This study is largely concerned with ConIn for distributed entities, and does
not deal with private and sensitive data. Schmidt [25] advises that ConIn
should be obtained when users explicitly participate in the provision of data,
or if users agree to allow a system to acquire data. Therefore, this study only
considers ConIn that is made available to the systems by the users or simulated
data, to conform with earlier studies. This means that the data used to test the
developed model is simulated, but not the data used to evaluate the developed
model; this is new data which is obtained from test results. Furthermore, by
no means does it use private and sensitive data. Moreover, this dissertation
does not involve any human participation, nor does it acquire any data from
outside sources. All data are gathered by experiments on personal computing
devices such as PCs or Raspberry Pies. The experiments are mostly carried
out using the free software ns-3 [76] and the open source platform MediaSense
[24]. A student license from MATLAB was available to the author during
these experiments. The other tools used in this thesis, namely PREMOLA and
SPIN, are also freely available [77].

3.3 Summary

This chapter presented the methodological approach driving the research un-
dertaken in this study, and the philosophical assumptions made. The chapter
started by describing the philosophical assumptions underpinning the study
and the way in which such assumptions can lead to the acquisition of
knowledge to answer the research question. The chapter then presented the
research method used to acquire this knowledge. In particular, a discussion
was presented as to why the applied method was chosen and how a comparison of this method with other methods allows validation of the research question. Following this, each of the steps involved in the research were discussed. The chapter also briefly mentions the research ethics involved in the study.
4 DACIM Architecture

The motivation behind the development of a new approach, called here DACIM, through an investigation of current and probable future scenarios has been discussed in Chapters 1 and 2. This chapter describes the design and development step of the DSR method. The realization of the envisioned DACIM requires an architecture that is capable of real-time and scalable communication. This chapter therefore presents an architecture that is capable of enabling distributed intelligence-assisted autonomic context information management in the IoT realm. This chapter builds upon the requirements defined in earlier chapters, particularly those discussed in Section 3.2.2 and illustrated in Figure 1. The chapter begins by presenting a DHT-based system model that assists in decoupling computation such as context-based intelligence, the management of Things and context information, communication etc. (see [Paper I] for details). A distributed model requires synchronization, and a PubSub model is designed and developed for this purpose (see [Paper III] for details) followed by a self-organized approach in order to address the proliferation of Things (see [Paper IV] for details). Distributed intelligence in IoT is a new concept, for which AI techniques such as belief networks and reinforcement learning have been used within the designed architecture (see [Paper V] for details). The architecture also includes context-based algorithms with regard to distributed clustering (see [Paper VII] for details) and a multi-modal reasoning (see [Paper VI] for details). See also the “Author Contributions” section for a description of the author’s contributions towards the realization of the DACIM architecture.

4.1 System Model

This section describes the system model which helps realize distributed context information management; [Papers I and II] largely contribute to this section, as does [78], which also corresponds to part of the first research sub-question. In the beginning of this section, some of the key concepts that are central to the design of DACIM are defined. Table 1 presents their definitions. The section continues by discussing distributed hash tables (DHTs), DCXP, logical sinks and finally the model that allows a physically distributed clustering approach.
4.1.1 Key concepts

Table 1: Key concepts

<table>
<thead>
<tr>
<th>Concept</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thing</td>
<td>A Thing in the IoT landscape can be, e.g., a sensor/actuator or a physical/virtual/logical object which is capable of contributing context. Other names of this include entity, node, connected device, objects, etc.</td>
</tr>
<tr>
<td>Flow</td>
<td>Short for flow of context from a Thing</td>
</tr>
<tr>
<td>Flow entity</td>
<td>OpenFlow/SDN-enabled entity, i.e., Thing</td>
</tr>
<tr>
<td>Virtual entity</td>
<td>A logical representation of a cluster (e.g. cluster head on the H-DHTs top-tier)</td>
</tr>
<tr>
<td>Sink/gateway</td>
<td>A (physical) sink that manages underlying Things in terms of connecting and collecting data from Things</td>
</tr>
<tr>
<td>Logical-sink</td>
<td>A physically distributed but logically synchronized physical sinks</td>
</tr>
<tr>
<td>Controller</td>
<td>A sink or gateway that is capable of providing intelligence and reasoning based on collected raw-data, in addition to sink’s capabilities</td>
</tr>
<tr>
<td>Edge controller</td>
<td>A controller that carries out few of the earlier cloud-based capabilities closer to the Things, usually, at the gateway for example those proposed in [Paper V] and in [28].</td>
</tr>
<tr>
<td>Cloud controller</td>
<td>A cloud-based controller that carries out high-level computing, for example, those proposed in [19, 79].</td>
</tr>
<tr>
<td>Distributed intelligence</td>
<td>Distributing intelligence by utilizing edge and cloud controllers to provide low-level and high-level intelligence respectively such as those mentioned in [Paper V].</td>
</tr>
<tr>
<td>Context-Aware reasoner</td>
<td>A reasoner capable of extracting knowledge by reasoning based on context.</td>
</tr>
</tbody>
</table>

4.1.2 DHTs

The development of the IoE (i.e. the future IoT) requires the control of enormous numbers of Things. Scalability becomes a challenging issue with the rising numbers of Things; centralized approaches fail to address this, as outlined in Chapter 2, and a DHT-based approach has been proposed to address
these shortcomings due to the advantages it offers [54]. Furthermore, hierarchical DHTs (H-DHTs) have been proven to be a better solution than flat DHTs [30]. Layering is used by H-DHTs to organize entities (Things). This study therefore exploits a two-tier H-DHT-based architecture to manage the ever-increasing number of Things and amount of context information in the IoT. More specifically, this dissertation proposes the management of the controller (sink) and clustering identifications (e.g. cluster heads) in the top tier, and context information and Things in the bottom tier, as illustrated in Figure 11 [78].

![Two tier H-DHT concept](image)

Figure 11: Two tier H-DHT concept (Adjusted from [Paper I] and [78])

4.1.3 DCXP

One of the DHT-based solutions is MediaSense; this can offer a scalable, decentralized, fast, flexible, lightweight and seamless solution which further exploits the Distributed Context eXchange Protocol (DCXP) [24]. DCXP was originally developed for the Ambient Networks project [24] and enables real-time context dissemination between DCXP-capable entities. It exploits an idea called register/resolve, akin to request/response, in which an entity is registered using a Universal Context Identifier (UCI) which other entities resolve to fetch context information. The UCIs are stored on the overlay using a DHT [15,78]. An entity can register several UCIs, which offers an advantage in situations where an entity is required to be identified in several ways; for example, a controller is responsible for storing several clustering identifications and the controller itself is regarded as the virtual entity for each cluster. All contributing entities create a DHT logically [24, 78]. Identification for each entity is created by hashing the IP address. Based on the requirements, the entity can immediately fetch context information or subscribe to forthcoming and continuous context information. The protocol is very simple to use.

One implementation of DCXP is the MediaSense IoT platform, which uses a peer-to-peer (P2P) concept as its underlying infrastructure. Like most P2P
systems, it also uses a rendezvous, i.e. a bootstrapping node, to initiate communication between entities whereby entities can join or leave. Primitive messages in DCXP are used for joining entities, setting, fetching and subscribing to context information etc. Figure 12 shows an example of such a scenario with three entities.

Figure 12: MediaSense and DCXP [78]

Through the use of DCXP, MediaSense removes any reliance on the IP address and underlying physical network infrastructure. This allows the system to run on a diverse collection of heterogeneous networks. Support for heterogeneous networks is one of the essential requirements for the present and future IoT. The capability of a single entity to register numerous UCIs provides a way of realizing a logical sink (see the following sub-section). Moreover, the register and resolve primitives allow DCXP-enabled MediaSense to be utilized as a PubSub model (see Section 4.3). This work extends the DCXP primitives to realize a scalable PubSub model, thus facilitating scalable and faster dissemination of context information among distributed context entities.

4.1.4 Logical-sink

In an IoT scenario, a single sink (controller) might need to control hundreds or thousands of Things, depending on the application. The penetration of Things is consistently on the rise, and this trend is expected to continue. Further, this penetration causes complexities within the network and difficulties in managing existing networks. These issues can be solved by employing an approach similar to Software Defined Networking (SDN) [78,80,81]. Mahmud and Rahmani have shown that FlowSensors perform better than regular sensors [73]. The control of heterogeneous Things and networks such as those in the IoT raises many challenges that can be resolved via SDN, as described by Martinez-Julia and Skarmeta [80]. However, it would be infeasible for a single controller to manage the ever-mounting number of Things. Tootoonchian and Ganjali explored several physically distributed controllers, with regard to an OpenFlow scenario [61] which was also used by Oliveira et
al. [82]. Similar to these proposals, this dissertation defines a logical sink as a physically distributed but logically synchronized controller (sink). Another advantage of a logical sink is that each controller carries out computing locally, and other controllers are then notified of the local event changes and thereby synchronized; this is achieved using a Publish/Subscribe model (see Section 4.3). The idea of employing DHTs with an SDN controller was also proposed in [83]. This idea underpins the concept of controllers (sinks in Figures 11 and 13) in this dissertation, which are responsible for controlling the underlying networks and Things. The logical sink is also responsible for other tasks, such as contextualizing IoT data [Paper V]; finding context-based similarity and making decisions about clustering [Paper VII]; assigning the respective ID for each Thing, flow of context information or cluster, etc. [Paper I]; determining tasks, finding experiences, learning beliefs and predictions [Paper V]; executing self-* capabilities [Paper IV] and multi-modal context-aware reasoning [Paper VI], etc. These tasks are elaborated in detail in the following sub-section and Sections 4.2 onwards in this chapter.

4.1.5 The Model

Prior to presenting the system model for utilizing and analysing context information, some definitions of key terms that used in the system model are provided ([Paper I] and [78] primarily contribute to this sub-section).

**Entity-ID**: Each Thing (entity) should be uniquely identified, meaning Things should have distinctive IDs. There are various techniques for obtaining the IDs; for example, it can be chosen randomly or can be obtained by hashing the sensor IP or MAC address [55]. The P2P infrastructure on which MediaSense is built upon supports both random IDs and hashing of IP addresses. Moreover, it can support and scale up to $2^{160}$ entities, which is more than IPv6 addressing can provide, and is therefore more than able to support the hundreds of billions of Things envisioned in the future IoT landscape.

**Flow-ID**: Flow from each entity is logically identified by the flow-ID. Flow can be defined depending on the requirements of a particular application [81]. The flow-ID is the flow of context information from a particular entity to the controller. The flow-ID will remain unchanged if an entity is responsible for the same flow of information. OpenFlow defines flow-tables, which consist of match-fields, action sets and statistics; flow is defined by the match-fields and flow-ID is defined by the action field.

**Context-ID**: In the logical-clustering approach, a cluster is identified by a context ID. The logical sink publishes the context ID to the Internet, and any
interested entity (researchers, service providers, analysts, administrators etc.) can subscribe to the context IDs.

**Virtual flow-entity**: A sink (controller) that is part of a cluster acts as a virtual flow entity, with very high computational capabilities compared to the Things to which the controller is connected. This eliminates the problem of choosing or electing a cluster head. This virtual flow entity can be thought of as the cluster head (one for each cluster). These virtual flow entities (i.e. cluster heads) are organized in the top-tier overlay and form a DHT.

**Context flow-table**: The match fields of the OpenFlow flow tables can be programmed (i.e. defined) according to a particular research requirement [81]. On this basis, a new flow table called the context flow table is introduced, which consists of the flow entity’s entity ID, flow ID and context ID.

Using a two-tier DHT network, the top-tier overlay stores the clustering identification (context IDs) and the flow entities form another DHT in the bottom tier. Figure 13 portrays a sample communication between the logical sink and three clusters. The following sub-section describes the communication process.

![Figure 13: Example of a two-tier network](image)

### 4.1.5.1 Communication

Things in the IoT domain can either be mobile or fixed, physical or virtual/logical. Things usually communicate with the nearby controller either directly or via an overlay hop. Communication between fixed Things and a controller is simple, since fixed Things usually communicate using the same physical sink; however, Things may need to communicate with different sinks occasionally. Figure 14 illustrates the communication between a flow entity and a sink. Three different types of communication can take place: sink-to-sink (S2S), sink-to-entity (S2E) and entity-to-sink (E2S). S2E communication takes place
in the forward path, and E2S in the reverse path. S2E and E2S communication are relatively established, and the challenges related to these have been addressed in many WSN studies; these are beyond the scope of this study. This dissertation further envisages another type of communication, namely entity-to-entity (E2E, or Thing-to-Thing) communication, which takes place via the logical sink. This study is primarily concerned with S2S communication, which is facilitated by designing and developing a PubSub model (Figure 15), thus enabling synchronization between sinks (see Section 4.3). Moreover, if there are Sn (≥2) physical sinks inside a logical sink and the total number of logical sinks is L, the total number of exchanged messages can be calculated using the following formula:

\[(S_n - 1) \cdot N_C \leq M\]  

(2)

where \(N_C\) is the total number of clusters and \(M\) is the total messages required per second. Details can be found in [Paper I].

![Figure 14: E2S and S2E Communication](image)

**Figure 14: E2S and S2E Communication**

![Figure 15: S2S communication](image)

**Figure 15: S2S communication**

### 4.1.5.2 Implementation

All types of flow entities are taken into consideration in the design of the DACIM. The logical sink lies in the middle of the three-tier IoT architecture, and communicates with both the Things (locally) and the cloud (globally via...
the Internet), and controls and manages the flow-entity traffic. The following steps summarize the implementation plan for utilizing context information by means of clustering to enable DACIM. The workflow is also shown in Figure 16.

- Flow-entity match-fields define the flow and the action defines the flow-ID
- Flow-ID is forwarded to the nearby physical sink S1
- S1 resolves flow-ID and returns corresponding context-ID
- S1 returns the entity-ID if not already assigned
- S1 forwards the request to other physical sinks (S2, S3… Sn) if no match found in S1
- If no context-ID is found, then a new context-ID is defined and published to other networks
- Logical-sink returns the context-ID to the requested flow-entity
- Regular and context flow-tables are updated by the logical-sink
- Statistics check for any flow mismatch, new flow-ID is defined in case of any mismatch

Here, similarity is central to finding or defining a cluster, as detailed in Section 4.2. The following figure shows a flow chart for the proposed model.

![Flow chart](image.png)

Figure 16: Flow chart
4.2 Clustering Heterogeneous Context Information

Making sense of current IoT data requires a new solution. Section 2.3 described how clustering can be useful in exploring data, whereby each cluster of data exhibits similar characteristics and is dissimilar from other clusters ([Paper VII] primarily contributes to this section and part of the first research sub-question). Here, the concept of similarity plays a central part in deciding which groups of data exhibit similar characteristics. There are many different techniques for extracting the similarity between data. One such technique is the Jaro-Winkler approach, which has been shown earlier to be successful in finding similarity in text and sentences [84,85] and performs better than other comparable techniques; for example, Cosine increases the time complexity and Jaccard fails to deal with the reverse order (Chapter 5 presents a performance analysis of these three techniques). The Jaro-Winkler algorithm was therefore chosen over Cosine and Jaccard as the similarity algorithm for textual data clustering. However, IoT generates not only textual data but also numerical data. Thus, a clustering approach in IoT requires the provision of a solution which can handle both types of data. To address this, a variant of the Jaccard algorithm is explored to find similarity in numerical data. Jaccard finds similarity defined by the size of the intersection divided by the union of the sample sets. Thus, in this approach, all the available data from a particular IoT Thing (e.g. a sensor or an actuator) is first fetched; this is equivalent to the union of the sample set and is denoted as the union of IoT data, $U_{data}$. New data, when it arrives, is denoted as $N_{data}$. Hence, the similarity can be obtained as follows:

$$S_n = \frac{N_{data}}{U_{data}}$$

(3)

Here, $S_n$ refers to the numeric similarity, and $N_{data} = \{\mathbb{R} : \text{new data is a particular thing}\}$, $U_{data} = \{\mathbb{R} : \text{all data is a particular thing}\}$. 
Algorithm 1 Cluster IoT data

1. Fetch current TConIn
2. if thing is already registered then
3.     fetch existing ConIn
4.     add current ConIn to existing ConIn
5. else if thing is not registered then
6.     register thing
7.     add ConIn
8. end if
9. Resolve thing
10. for all TConIn do
11.     while TConIn is not empty do
12.         if ConIn is numeric then
13.             calculate Jaccard-like similarity (reference value, ConIn)
14.         else if ConIn is textual then
15.             calculate Jaro-Winkler similarity (reference value, ConIn)
16.         end if
17.         if similarity is greater than threshold (e.g. 90%) then
18.             if scanning existing cluster found true then
19.                 add ConIn to the cluster’s ConIn
20.             else if no existing cluster found then
21.                 define a new cluster
22.                 add ConIn
23.             end if
24.         end if
25.     end while
26.     publish cluster information for subscriptions
27. end for

4.2.1 Clustering Algorithm

This sub-section summarizes the proposed approach with respect to supporting clustering distributed heterogeneous context information. The clustering approach designed and developed here extends the DCXP algorithm to find similarity in distributed IoT data. The logical sink, that is, a cluster of edge controllers, fetches ConIn from the Things it is responsible for; then, this ConIn is fed into the similarity algorithm to find the respective cluster. After fetching a Thing’s context information (TConIn), this TConIn is then merged with the existing ConIn. Depending on the type, similarity in the available ConIn is calculated using either a Jaro-Winkler or Jaccard-like approach. Since this study uses instantaneous clustering, the obtained similarity is
matched, as mentioned in Section 4.1, based on a specific requirement from the requestor. The matched ConIn is then either added to an already available cluster or a new cluster is created. Once the cluster is matched or created, the update is published to other edge controllers and to the cloud controller, as shown in Figure 17. Algorithm 1 further clarifies each step [Paper VII].

Figure 17: Communication between edge and cloud controller

4.3 The Publish/Subscribe Model

As illustrated earlier in Figures 15 and 17 and in Chapters 2 and 4, there is a need to enable dissemination of context information. Furthermore, it is evident that S2S communication, that is, enabling a logical sink, poses a challenge. In response, this section presents a Publish/Subscribe (PubSub) model (contributions to this section are primarily made by [Paper III] and [78]). This challenge is solved by extending the DCXP protocol on the MediaSense platform. The remainder of this section describes how this is achieved, thus answering part of the second research sub-question.

A typical PubSub model is illustrated in Figure 8. Typically, a publisher publishes events in which subscribers can express interest via an event dispatcher. Subscribers are notified by the event dispatcher of any changes to the subscribed items, e.g., addition or deletion of values. In a logical sink scenario, this implies that any changes in a sink or controller need to be conveyed to the other sinks or controllers.

4.3.1 Approach

Section 4.1 introduced DCXP and MediaSense, whereby each entity in DCXP-enabled MediaSense registers via a UCI, which other entities can re-
solve. This concept is further extended, whereby each publisher registers using a UCI and other subscribers resolve the UCI in order to fetch context information. Logical sink synchronization, that is, S2S communication, is illustrated in Figure 15. Figure 18 shows communication between different logical sinks via MediaSense. In this approach, a logical sink registers itself using a UCI, and the context IDs (clustering identifications) associated with the logical sink are registered as UCI data. Another logical sink, residing remotely, resolves the UCI and fetches the context ID, as shown in Figure 19. The logical sink is responsible for gathering context information from the distributed Things, which generate context information and are responsible for creating the context IDs based on the context similarity, as discussed above (see Figure 18). This approach can be evaluated in terms of both these purposes, as illustrated in Figures 15 and 18. The existing DCXP-enabled MediaSense implementation does not support the registration of context information and the UCI at the same time. After registering the UCI, context information is collected using GET and SET messages, and this incurs delays. However, an extended DCXP approach enables a faster response, as shown in Section 5.3.1.

Figure 18: DCXP-enabled MediaSense as PubSub model

Figure 19: Approach to utilizing MediaSense
4.3.1.1 Algorithms

Algorithm 2 and 3 are the publisher and subscriber algorithms, respectively. The publishing algorithm begins by starting the MediaSense bootstrap node (see Figure 12). Following this, the algorithm updates the UCI with current and historical context information if it is already registered. Otherwise, the UCI is registered, along with its associated context information. The registered UCI can be deleted, and a logical sink can essentially register several UCIs at the same time, allowing multiple identities for single entity. This offers versatility, for example, an entity representing both a physical sink (part of the logical sink) and a logical sink (communicating with other logical sinks) can communicate with other entities using different identifications. The registered UCIs are stored on the overlay, meaning that the context information is not lost as long as the UCI is not deleted when an entity is down or fails. This means there is no risk of central point of failure. The resolution of the UCI for subscription is described in Algorithm 2. After resolving the UCI, the algorithm then fetches context IDs until these are empty. The context ID to be subscribed is then checked against the fetched context IDs, and a notification message is sent to the subscription requestor when a match is established.

Algorithm 2 UCIRegistration for publishing

1. Initialize MediaSense platform
2. Run the MediaSense bootstrap node
3. if UCI is not registered then
4. Invoke Registrator class
5. Initialize registration and add UCI invoking Media-Sense platform’s registerUCI method
6. Add context information
7. else if
8. Invoke Updater class
9. Initialize Updating and update UCI invoking Media-Sense platform’s update method
10. Update context information
11. end if
12. publish cluster information for subscriptions
13. end UCIRegistration for publishing
4.4 Self-Organization Towards Enabling Autonomic Management of IoT

Chapter 2 introduced the concept of self-organization and its role in enabling autonomic computing. The proliferation of Things in the IoT requires suitable management via a controller to enable autonomic computing. This corresponds to providing the self-* capabilities envisioned by autonomic computing [31]. To address this, this section describes the design and development of algorithms to assist an edge controller with self-organization in DACIM [Paper IV, 78]; this also corresponds to the third research sub-question, the eventual goal of which is to enable autonomic management with minimal or no intervention from outside sources.

This section begins by describing self-organization with the logical-clustering approach using DACIM. Table 2 shows each of the self-* capabilities of an autonomic computing entity.
Table 2: Self-* capabilities of an autonomic entity [31]

<table>
<thead>
<tr>
<th>Aspects</th>
<th>Capabilities</th>
</tr>
</thead>
</table>
| Self-configuration | A new entity joins, advertises itself and discovers other entities  
|                 | Adjusts and integrates automatically and seamlessly according to the high-level objectives (policies) set by outside sources                   |
| Self-optimization  | An entity should be able to optimize the local operation parameters according to global policies set by outside sources  
|                 | Learning and altering policies adapted by others  
|                 | Should be able to adjust in the case of policy conflict                                                                                  |
| Self-healing     | Reconfigurations of the entities in case of failures  
|                 | Redeem for configuration and optimization failures                                                                                       |
| Self-protection  | An entity should be able to protect itself from outside or other undesirable attack  
|                 | Ensure security for communication between entities                                                                                       |

Typically, a manager (e.g. a sink or a controller) executes the self-* capabilities. This aids cooperation with other joined, organized entities and/or outside sources (e.g. a human administrator or another sink). A manager facilitates the analysis, planning, and execution of the policies set by outside sources and permits an organized entity to collaborate with another organized entity inside the system (after having joined). This joining of entity and execution of policies can be regarded as a control loop. This sums up the relation between self-* capabilities, autonomic computing and the execution of policies (the details of which are given in [Paper IV] and [78]). The policies are responsible for the implementation of self-* capabilities. These policies are first incorporated within a system by outside sources, and new policies are added and adopted as the system evolves and encounters new problems. Addition and adoption usually require learning, i.e. awareness (through obtaining knowledge) of each organized entity. An example of this learning through extracted knowledge is given in the next section and in [Paper V].

This section focuses on describing the design and development of the self-* capabilities with one limitation, in that it does not explore the self-protection aspect, which is beyond the scope of this study; an approach to handling security is discussed in the Future Work section of Chapter 6. The idea is to extend DCXP-enabled MediaSense in order to support an IoT controller with respect to self-organization. Three self-* aspects have therefore been designed and developed, and are included as extended primitive functions. The next
sub-section starts by showing how self-* capabilities enable support for self-organization in logical-clustering; their contribution to distributed intelligence is then discussed.

4.4.1 Self-Organization Support for Logical-Clustering

Distributed context-based clustering relies heavily on logical sinks, as described in earlier sections. Here, clustering tasks such as creation, insertion and deletion are dependent on controllers (sinks). In addition to this, controllers need to discover and collaborate with other available controllers, and should also organize themselves. It is of great importance that controllers are able to organize themselves with minimal or no support from outside sources in a dynamic and real-time IoT. Likewise, logical-clustering involves real-time communication, and it is vital that this evolves accurately and automatically in real-time. As in any other system, a controller initially joins the system, and this is also the first operation in the autonomic control loop. This process also includes analysing the policies associated with the join request. During this operation, the self-* algorithms are implemented on the extended MediaSense platform; based on the outcome of these algorithms, the controller is adapted, after which the execution is carried out. The platform has awareness of all these actions, as shown in Figure 20; this process can be summed up as follows:

- A (sink) controller joins
- The platform analyses the policies, i.e. evaluates the self-* algorithms
- The platform also adapts the controller based on the policies’ outcome
- The controller is ready for execution (after this stage, controller is said to be organized)
- The platform has awareness of all these actions

![Figure 20: Supporting self-organization with logical-clustering](image-url)
4.4.2 Self-* Capable Algorithms

The three self-* capabilities of self-protection, self-healing and self-organization are described below. Further details of these capabilities can be found in [Paper IV] and [78].

4.4.2.1 Self-configuration

The task of this self-* capability is to ensure automatic and seamless integration of a joined entity within any system, e.g. in an IoT application. This was achieved by designing and developing a new primitive message for DCXP called joinedUCI. The bootstrapping node was used to define a global publisher, i.e. global_ucci in MediaSense, and each entity trying to join the platform using the extended joinUCI primitive function automatically subscribes to the publisher without knowing which entity currently holds the global_ucci. This is illustrated in Figure 21, and the procedure for achieving this is given in Algorithms 4 and 5.

The self-configuration algorithm comprises several steps as listed below:

- Controller joins
- Configure global_ucci (at beginning on MediaSense bootstrap entity)
- Controller configuration
- Check for self-healing, i.e., re-configuration
- Discover other controllers

![Figure 21: Discovering entities](image-url)
Algorithm 4 sink_join

1. \textbf{begin} \\
2. create an instance of MediaSensePlatform \\
3. initialize platform with network settings (Bootstrap IP address, bootport, local port) \\
4. \textbf{while} MediaSense is bootstrapped \\
5. \hspace{1em} declare UCI (i.e. identity of the node) \\
6. \hspace{1em} invoke MediaSensePlatform’s joinUCI \\
7. \hspace{1em} \textbf{if} MediaSensePlatform’s selfHealing is true \\
8. \hspace{2em} \textbf{if} the UCI is not listed on global_ucci \\
9. \hspace{3em} publish on global_ucci by invoking MediaSensePlatform’s config method \\
10. \hspace{3em} return the UCI’s current configuration status \\
11. \hspace{2em} \textbf{end if} \\
12. \hspace{1em} \textbf{else if} \\
13. \hspace{2em} register the UCI on MediaSensePlatform \\
14. \hspace{3em} publish on global_ucci by invoking MediaSensePlatform’s config function \\
15. \hspace{2em} \textbf{end if} \\
16. \hspace{1em} invoke MediaSensePlatform’s selfConfiguration \\
17. \hspace{1em} synchronizes with the existing UCIs every T seconds \\
18. \hspace{1em} \textbf{end while} \\
19. \textbf{end}

Algorithm 5 sink_discovery

1. \textbf{begin} \\
2. resolve global_ucci (based on PubSub’s subscription algorithm) \\
3. read the subscribeable UCIs \\
4. store the UCI to the global_ucci \\
5. update global_ucci \\
6. merge or renew depending on the configuration \\
7. subscription is synchronized whenever this method is invoked \\
8. \textbf{end}

4.4.2.2 Self-healing

Algorithm 6 presents the self-healing algorithm, which primarily handles entity reconfiguration in the case of a failure. This algorithm achieves the reconfiguration by checking two parameters, UCI and sink ID. If an entity is already
present, the sink reconfigures using the previous configuration; and if not, a new configuration is created by invoking the self-configuration.

**Algorithm 6 self-healing**

1. i. sink_duplication_check
2. begin
3. resolves the UCI
4. fetches associated information
5. if the sink id is found with the fetched information
6. returns true
7. else if
8. returns false
9. end if
10. end

11. ii. sink_duplication_check
12. begin
13. resolves the UCI
14. fetches associated information
15. if uci exists in the current MediaSense
16. reconfigure the node and fetch previously existing data
17. else if
18. start uci registration
19. while register
20. invoke selfConfiguration
21. end while
22. end if
23. end

### 4.4.2.3 Self-optimization

Self-optimization implies that an entity, i.e. a Thing, should optimize itself according to the policies set by the outside sources. It also ensures that a Thing is able to perform efficiently and to the best of its ability. One example is the optimization of clustering information associated with logical-clustering. Each controller is responsible for creating a cluster in real-time; therefore, each controller should publish these newly created clusters to other edge controllers and to the cloud controller at a specified interval. Algorithm 7 demonstrates this procedure, including the automatic and periodical insertion of the new context IDs, seamless integration with existing context IDs and synchro-
organization with other edge controllers in the case of logical-clustering. Depending on requirements, a specific cluster (e.g. with a timestamp for time to live (TTL) attached) can be removed, or all clustering information may be deleted; the algorithms also deal with these events. This self-* capability executes these policies and optimizes the sink.

Algorithm 7 sink_optimization

1. I. Insert context-ID
2. begin
3. resolves the UCI
4. checks for new context-IDs
5. if new context-IDs are found
6. insert new context-IDs in the UCI and adjusts seamlessly with existing context-IDs
7. invoke Insert ContextID Policies
8. end if
9. end

10. II. Delete context-ID
11. begin
12. resolves the UCI
13. checks for context-IDs to be deleted
14. if context-IDs need to deleted
15. delete existing context-IDs in the UCI and adjusts seamlessly with existing context-IDs
16. invoke Delete ContextID Policies
17. if single context-ID with a TTL
18. delete context-ID
19. else if
20. delete context-IDs
21. end if
22. else if
23. end

4.5 Distributed Intelligence

As intelligence implies the application of knowledge, distributed intelligence implies that the application of knowledge is distributed, that is, decentralized. This section first describes the significance of distributed intelligence, followed by an approach that enables this vision [Paper V]. Section 2.6 discussed the need for and important of intelligence in the IoT domain. Intelligence in the IoT can also be realized as a fully autonomous IoT controller which would
make decisions, actions and predictions (DAPs) independently based on the context information at its disposal. Furthermore, these operations should be carried out as close as possible to the Things, in order to comply with a real-time IoT vision. In response, a Future Internet of Things Controller (FITC) is proposed and designed, which is able to provide DAPs and low-level context-based intelligence by employing edge controller(s), while a cloud controller deals with high-level intelligence; this corresponds to the fourth research sub-question. Details of this can be found in [Papers V and VI]. These two publications contribute predominantly to this section.

4.5.1 Approach

Reaching the full potential of IoT requires two other aspects, namely Information of Things and Intelligence of Things. Information of Things involves reaping value from the collected IoT data, while Intelligence of Things means making sense of the Information of Things. An improvement in the Intelligence of Things requires exploring a two-level form of intelligence, whereby small data at the edge can provide low-level intelligence and big data can provide high-level intelligence in the cloud. This division of intelligence can be seen in Figure 22, and was inspired by the information-knowledge hierarchy. Knowledge lies in the middle of the hierarchy, and deals with questions of how. The application of answers to questions of how is more commonly known as intelligence. An edge controller can answer questions related to how that are specific to each IoT application by reaping value from the contextualized IoT data. Knowledge should be distributed between the edge and cloud controllers, since the edge controller may not be able to deal with all the intelligence due to a lack of computational capability. For example, the discovery of patterns in the IoT data and the understanding and evaluation of IoT data require high-level data mining algorithms, and a cloud controller is best suited to deal with this, as shown in earlier studies. Some examples of decisions and actions in edge intelligence include turning lighting and heating on/off in a SmartHome or SmartFarm, automated harvesting and water sprinklers, and caregiver notifications.

![Figure 22: Distributed intelligence for IoT](image)

Figure 22: Distributed intelligence for IoT
4.5.2 Leveraging AI Techniques

The examples of edge intelligence given in Section 4.5.1 have so far relied only on rules. Furthermore, the policies involved in self-organization also depend heavily on human administered rules. Although rules have historically been prominent in the IoT, these fail to scale well with an increase in the number of Things and in uncertain situations. To address this problem and to move on from pre-assumed intelligence at the edge, AI techniques such as belief networks can help; each piece of context information in an IoT application is given a prior belief, and based on the belief calculation, a task can be executed.

A belief network follows two simple rules involving sums and products, as shown in [Paper V]. An example of a belief network-based action is the preparation of dinner in a SmartHome. The SmartHome edge controller knows that the user usually has dinner at around 19:00, and it therefore activates the “turn on oven” action to a desired temperature at 18:15 every day. On a non-regular day, when the user plans to eat at a friend’s house or to go out for dinner, the controller makes a prediction as to whether or not the “turn on oven” function should be activated. Here, intelligence (that is, the application of knowledge) is not pre-programmed into the controller, and this allows the controller to learn not to take the usual actions when the user is not around. This offers another example of autonomic management of context information in an IoT application.

Learning at the edge can be difficult and time-consuming, since most learning algorithms require a great deal of data before they can start learning. Reinforcement learning is a relatively new learning technique which allows learning from experience, and this flexibility makes it well suited for learning at the edge in IoT. This kind of learning can help learning policies or rules to reduce time and complexity. Using reinforcement learning at the edge, a controller can predict an outcome by learning from experience. Figure 23 illustrates this. Once an edge controller learns from experience, thereby, belief through the frequency of each ConIn tuple and the total frequency of each set of ConIn, the learning algorithm learns from the experience to take a particular action.

![Figure 23: Learning experiences based on reinforcement learning](image-url)
4.5.3 Algorithms

The algorithms that enable intelligence at the edge to execute DAPs and to contextualize and learn from IoT data are summarized below.

4.5.3.1 Contextualization

Algorithm 8 provides more meaning to the collected raw data by answering the fundamental questions in the knowledge pyramid. Its input is the inserted raw data and its output is contextualized raw data.

<table>
<thead>
<tr>
<th>Algorithm 8 Contextualization</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Initialize Connection/communication to collect data</td>
</tr>
<tr>
<td>2. while there is sensed data do</td>
</tr>
<tr>
<td>3. if sensed data is not contextualized then</td>
</tr>
<tr>
<td>4. what: raw-data</td>
</tr>
<tr>
<td>5. where: origin of the raw-data</td>
</tr>
<tr>
<td>6. when: time of occurrence</td>
</tr>
<tr>
<td>7. who: originator of the raw-data</td>
</tr>
<tr>
<td>8. else if sensed data is contextualized then</td>
</tr>
<tr>
<td>9. find its frequency</td>
</tr>
<tr>
<td>10. frequency at the given ConIn</td>
</tr>
<tr>
<td>11. what other ConIn it is related to, e.g. some actuation</td>
</tr>
<tr>
<td>12. update ConIn</td>
</tr>
<tr>
<td>13. end if</td>
</tr>
<tr>
<td>14. determineTask(TConIn);</td>
</tr>
<tr>
<td>15. addToAConIn();</td>
</tr>
<tr>
<td>16. addToTConIn();</td>
</tr>
<tr>
<td>17. end while</td>
</tr>
</tbody>
</table>

4.5.3.2 Determine tasks

Algorithm 9 is responsible for making a decision to execute a task once experience is obtained. The algorithm first fetches a Thing’s context information (TConIn) and based on the calculated belief, it executes tasks, finds associated context information (AConIn), and inputs TConIn and AConIn for learning.
Algorithm 9 Determine Task

1. Fetch TConIn
2. for all TConIn do
3.      while TConIn is not empty do
4.          Look up the prior-belief for each ConIn
5.          Calculate probability of tasks related to TConIn (Eq. 1)
6.          if probability of task > threshold then
7.              Execute task(s), i.e. decision
8.          end if
9.      end while
10.     while TConIn is not empty do
11.        findAConIn(TConIn)
12.        predict (TConIn);
13.     end while
14.     findExperience(TConIn, AConIn);
15. end for

4.5.3.3 Find experience
Algorithm 10 concerns the finding of experiences. It takes TConIn and AConIn as its input, and outputs experiences. This contributes to learning, as shown in Figure 23 and Algorithm 11. In the case where an edge controller fails to take an action, the cloud is consulted.

4.5.3.4 Prediction
Algorithm 12 makes predictions in the case of missing values in a set of ConIn. This algorithm takes TConIn with missing values as input, and finds other ConIn (OConIn) for the available ConIn. Next, the algorithm finds the frequency for each OConIn. It then checks whether the frequency is the highest among OConIn frequencies; when the frequency is the highest, it identifies this as the most probable missing ConIn.
Algorithm 10 Find Experiences

1. Fetch TConIn
2. for each what (sensed data) do
3.      for each when (time) do
4.          add to tuple
5.          increase frequency
6.        for each where (location) do
7.            add to tuple
8.            increase frequency
9.        for each who (originator) do
10.           add to tuple
11.           increase frequency
12.          find AConIn
13.          add to tuple (TConIn, AConIn) to learn experiences
14.          learnBelief(TConIn, AConIn);
15.      end for
16.  end for
17. end for

Algorithm 11 Learn Belief

1. Fetch TConIn, AConIn
2. for i = size of TConIn do
3.      for j = size of AConIn do
4.          if TConIn(i) is associated with AConIn(j) then
5.              fetch frequency of (TConIn(i),AConIn(j))
6.              calculate probability of (TConIn(i),AConIn(j))
7.          if probability > threshold (Eq. 2) then
8.              new belief is obtained
9.          end if
10. else if
11.          forward for higher level action (to cloud)
12.      end if
13.  end for
14. end for
4.6 Multi-modal Context-Aware Reasoner (CAN)

The previous section described the application of knowledge in order to provide intelligence. Prior to applying knowledge in the IoT, an IoT application needs to infer this knowledge. Reasoning is usually employed to harvest knowledge. Since context-aware computing has been deemed one of the key enablers of future IoT, context-aware reasoning in the IoT is also gaining attention [Paper X]. Context-aware reasoning can also be used to fill gaps in the collected raw data and provide services, which corresponds to the fourth research sub-question. However, the IoT consists of several different IoT applications, and more often than not, an IoT edge controller needs to control more than one IoT application, meaning that a CAN must be able to harvest knowledge for each of these IoT applications. Such a controller therefore needs to offer different context-aware reasoning to improve the Intelligence of Things. In response, a multi-modal CAN has been designed and developed at the edge; it is presented in this section, and its contribution is explored in [Paper VI].

4.6.1 Approach

A Context-Aware Reasoner (CAN) is defined as one which is able to reason based on the context information fed into it. The CAN should therefore be able both to react to the context, thus being context-aware, and to reason based on the context. Some of the more popular reasoning techniques include rule-based, ontology and Bayesian/statistical reasoning. Figure 24 illustrates a multi-modal CAN (MM-CAN) in which different CANs can be used for different IoT applications. The proposed MM-CAN can be inserted between the information and knowledge, as shown in Figure 22. The contextualized raw data is filtered to decide which CAN is needed by a particular ConIn, and the relevant CAN is then used. Often, a set of ConIn may be required by different

---

**Algorithm 12 Prediction**

1. for each available tuple of ConIn do
2. find OConIn for this tuple
3. for each available tuple of ConIn do
4. fetch frequency
5. if frequency of OConIn equals highest frequency then
6. ConIn predicted
7. end if
8. end for
9. end for
CANs, and the MM-CAN should be able to handle this. The reasoned ConIn (i.e. harvested knowledge) is then stored in the IoT edge controller. This harvesting of knowledge at the edge is helpful, for example, in giving faster responses and reducing dependency on a cloud-based solution. Some examples of the knowledge that an edge controller can offer are shown in Table 3.

![Diagram of different CANs for different IoT applications]

**Figure 24: Different CANs for different IoT applications**

<table>
<thead>
<tr>
<th>Knowledge</th>
<th>Application(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turning on/off light</td>
<td>SmartHome/Garden</td>
</tr>
<tr>
<td>Turning on/off heating</td>
<td>SmartHome</td>
</tr>
<tr>
<td>Preparing food</td>
<td>SmartHome</td>
</tr>
<tr>
<td>Controlling air pressure</td>
<td>SmartGarden</td>
</tr>
<tr>
<td>Dimming light</td>
<td>SmartGarden</td>
</tr>
<tr>
<td>Automated harvesting</td>
<td>SmartGarden</td>
</tr>
<tr>
<td>Controlling water sprinklers</td>
<td>SmartGarden</td>
</tr>
<tr>
<td>Prescription</td>
<td>Elderly care (SmartHealth)</td>
</tr>
<tr>
<td>Requesting an appointment</td>
<td>Elderly care (SmartHealth)</td>
</tr>
<tr>
<td>Notifying caregiver</td>
<td>Elderly care (SmartHealth)</td>
</tr>
</tbody>
</table>

Reasoning is a prerequisite for offering such knowledge; however, earlier studies were mostly limited to rules- and/or ontology-based reasoning. In the previous section, it was shown that Bayesian reasoning (i.e. statistical reasoning) may help in reducing dependence on the rules, and the following chapter confirms its performance. However, the goal of MM-CAN is not to determine...
which reasoning technique demonstrates better results (although this may be a good research study for the future). This work uses two reasoning techniques (rules and statistical reasoning) for three IoT applications, in order to demonstrate the feasibility of harvesting knowledge at the edge and to fulfil the vision of distributed intelligence and autonomic management of context information. An earlier study found that no prior IoT solution has provided a different model of reasoning [5], and due to the computationally constrained nature of any IoT edge controller, the provision of intelligence (i.e., knowledge) at the edge is limited to low-level intelligence in the IoT domain, as discussed above. Table 4 shows the statistical reasoning needed to execute a task at the edge controller.

Table 4: Bayesian reasoning for sprinkler activation

<table>
<thead>
<tr>
<th>Temperature (0.1)</th>
<th>Rain (0.5)</th>
<th>Soil moisture (0.4)</th>
<th>Sprinkler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low (0.2)</td>
<td>Low (0.6)</td>
<td>Low (0.65)</td>
<td>0.58</td>
</tr>
<tr>
<td>Low (0.2)</td>
<td>Medium (0.35)</td>
<td>High (0.1)</td>
<td>0.24</td>
</tr>
<tr>
<td>Medium (0.3)</td>
<td>High (0.05)</td>
<td>Medium (0.25)</td>
<td>0.16</td>
</tr>
<tr>
<td>High (0.5)</td>
<td>High (0.05)</td>
<td>High (0.1)</td>
<td>0.12</td>
</tr>
<tr>
<td>Medium (0.3)</td>
<td>Low (0.6)</td>
<td>Medium (0.25)</td>
<td>0.43</td>
</tr>
<tr>
<td>High (0.5)</td>
<td>Medium (0.35)</td>
<td>Low (0.65)</td>
<td>0.49</td>
</tr>
</tbody>
</table>

4.7 The Architecture

The above sections have described how DACIM can be achieved. Based on these descriptions, the following Figure 25 illustrates the architecture that enables the implementation of DACIM. It demonstrates how raw data from Things are managed by edge controllers using clustering and low-level intelligence (LLI); high-level intelligence (HLI) is taken care of by the cloud controller.
4.8 Summary

This chapter has presented the vision of the DACIM architecture, corresponding to the design and development stage of the DSR method. The design of a context-based approach for a future IoT in order to utilize and analyse context information efficiently requires many research challenges to be met. In view of these requirements, this chapter first described the system model, followed by an approach to utilizing clustering information by means of a distributed clustering approach. The chapter continued by illustrating the scalable dissemination of context information by means of a PubSub model using an extended version of distributed IoT protocol, DCXP. Following this, an approach to organizing the IoT controller with minimal outside intervention was described, paving the way for autonomic computing by bringing system intelligence to the edge of the IoT. This vision of autonomic computing was further extended to enable distributed intelligence using contextualized IoT data, and a FITC was proposed and designed. An edge controller capable of providing low-level intelligence (LLI) based on context information and AI technique was designed and developed. High-level intelligence (HLI) was proposed to be used in the cloud, due to the computationally constrained nature of edge controllers. Furthermore, the application of knowledge requires inferring knowledge, and a CAN was therefore proposed, designed and developed for IoT applications. In order to handle a range of IoT applications, a multi-modal CAN was proposed, designed and developed. The overall architecture of DACIM is shown.
in Figure 25. Several algorithms for DACIM were also presented in this chapter. The following chapter explores the performance of each of the described approaches involved in DACIM.
5 Evaluation of DACIM

An architecture called DACIM enabling a context-based approach to handling heterogeneous context information was designed and developed in the previous chapter by examining present and future scenarios. This chapter presents the evaluation of DACIM, the aim of which is to convey to the research community the novelty and usefulness of the study. The chapter starts by carrying out a network performance analysis of the logical-clustering approach by means of ns-3 simulations; this is followed by an analysis of each of the remaining approaches: clustering, PubSub, self-* capabilities, distributed intelligence and MM-CAN. Each section describes the experimental setup, in which earlier studies are consulted where appropriate. More on this evaluation and the experimental setups can be found in Sections 3.2.4 and 3.2.5, and the corresponding publication for each scenario.

5.1 Network Performance

With the increase of the number of Things in the IoT domain, the network performance of real-time communication has become an important issue which needs to be taken into consideration. This section investigates some of the important network performance metrics of the logical-clustering approach. The main contribution to this section can be found in [Paper II] and in [78]. The ns-3 simulation tool was used in the design and development of an Open-Flow-enabled network of wireless sensor networks (WSNs), as shown in Figure 26. WSNs are integral to the IoT landscape and were therefore employed for demonstration purposes. The flows from the entities (flow entities) were measured in terms of mean delay, jitter and packet loss ratio in order to verify the reliability, scalability and reachability of the designed network. Performance was measured with FlowMonitor, a network monitoring framework for ns-3. The main contributions of this section are summarized as follows:

- Design of a WSN of logical-clustering of flow-entities in ns-3
- Verify reliability
- Verify reachability
- Examine scalability of the network for increased number of entities and clusters
5.1.1 Simulated Network

Figure 26 shows the network designed and developed in ns-3. Three WSNs are designed; network 1 has only fixed nodes, while the nodes in networks 2 and 3 are mobile (randomly moving). Each network has one gateway, an OpenFlow controller which connects the gateways, and 20 sensor nodes (the terms ‘node’ and ‘entity’ are used interchangeably).

5.1.2 Performance Evaluation

Evaluation of the performance of logical-clustering corresponds to the second part of the first research sub-question, and involves verifying how the model behaves in a real-life scenario. Evaluation is carried out by employing different simulation scenarios for scalability, reliability and reachability in real-time communication. Some of the performance metrics (mean delay and packet loss ratio) are evaluated below for different situations such as varying rates of information flow from entities and varying number of entities in a cluster (the terms ‘cluster’ and ‘group’ are used interchangeably in this section). Paper II gives details of these.

5.1.2.1 Varying Information Rate

Figure 27 illustrates performance measurements for varying information flows from entities. The information flow in ns-3 relates to packets per second (p/s) sent from one entity. For this particular evaluation, three clusters (groups) were used, and each cluster was assisted by nine entities. Entities in each cluster are distributed over the three networks, and are thus logically clustered.

Figure 27(a) and (b) show the mean delay and the packet loss ratio for each cluster, respectively, for varying rates of information flow. Each cluster
demonstrates qualitatively similar performance for information flow rates of up to 10 p/s for both metrics. The mean delay for clusters 1, 2 and 3 increased by 0.3172s, 0.2629s and 0.2166s, respectively, for an increase in information flow rate of 83%, indicating that the mean delay did not display a great deal of fluctuation. However, the first cluster demonstrated a rise in information reachability for a flow rate of 11 p/s in terms of packet loss ratio. For information flow rates of up to 10 p/s, reachability did not show high fluctuations. Further confirmation of the high reachability of information can be seen from the fact that clusters 1, 2 and 3 experience increase in packet loss ratios of 0.0635, 0.0285 and 0.0205 respectively compared with a flow rate of 6 p/s. The rise is about 20%, as compared with an 83% increase in the flow rate in terms of percentage. This high reachability would be very advantageous in real-time context information sharing, and would allow the sharing of a large amount of context information.

![Figure 27: (a) Mean delay and (b) packet loss ratio for varying information flow](image)

### 5.1.3 Effect of Increasing Nodes per Cluster and Cluster Size

Reliability and reachability for different information flow rates were discussed in the previous section. The assessment was conducted for a constant number of entities per cluster, and the cluster size was fixed. Information flow rate will always fluctuate in current and future IoT scenarios, and the entities responsible for data acquisition will also change; context information will therefore be generated which requires different clustering of contexts, i.e. groups of data. Scalability therefore becomes an important issue with respect to the growing number of entities and clusters for real-time context information management. This section therefore explores the effect of increasing the number of entities per cluster and cluster size. Figures 28(a) and (b) show the mean delay for varying entities per cluster and the mean delay for different cluster sizes, respectively. When the entities per cluster were increased by 100% and 200%, the mean delay increased by 15% and 22%, respectively, indicating that the logical-clustering approach is able to scale well for an increased number of entities per cluster. On the other hand, a cluster size of six gave results...
of 6% and 19% compared with a cluster size of three, for information flow rates of 5 p/s and 8 p/s, respectively. The fluctuation in delay was only 13% for an increase of 60% in the information flow rate and a doubling of cluster size. However, the mean delay decreased for a cluster size of six with a flow rate of 9 p/s, probably due to wireless interference and the random movement of the nodes of the chosen Random Walk model. Packet losses for this particular scenario were higher. The results reported in this section suggest that the proposed logical-clustering approach can provide scalability for delay in terms of increases in entity numbers per cluster and cluster sizes.

![Figure 28](image)

Figure 28: Mean delay for varying (a) entities per cluster and (b) cluster size

For reliable context information distribution, scalability and high distribution of context information are also vital. In view of this, Figure 29(a) shows the packet loss ratio (PLR) performance for different numbers of entities per cluster. PLR increased by 22% for a 100% increase in the number of entities per cluster; when the number of entities per cluster was increased by 200% (to the maximum volume of the designed network), the packet loss ratio increased by 48% in cluster 2. Furthermore, Figure 29(b) demonstrates that a cluster size of six compared with a size of three (a 100% increase) shows PLR degradation of only 26% and 33% for flow rates of 5 p/s and 8 p/s, respectively. This demonstrates the scalability in terms of reachability.

![Figure 29](image)

Figure 29: PLR for varying (a) entities per cluster and (b) cluster size
5.1.4 Packet Size vs. Information Flow Rate

The results reported so far in this section correspond to a fixed packet size of 512 bytes. The performance metrics showed better performance when the information flow rate was below or equal to 10 p/s for this packet size. However, when the packet size was halved, i.e. to 256 bytes, the information flow rate increased by 338%; that is, it increased to 35 p/s. Due to this upsurge in the information flow rate, the mean delay increased, as a result of the increased number of received packets. Notably, the information flow reachability decreased by 15%. The goal of this measurement was to check the effect of packet size on information flow rate. It is clear that packet size plays a role in information flow rate and reachability.

Recalling the second part of the first research sub-question, which concerned verification of the real-world behaviour of the proposed model, this section evaluates the proposed concept for different simulation scenarios. The results suggest that physically distributed clustering is feasible without greatly affecting scalability, reliability and reachability.

5.2 Distributed Clustering of Heterogeneous Context Information

The first research sub-question also involves a problem whereby clustering heterogeneous IoT data and finding similarities in numerical and textual data remain challenging. Enabling such clustering requires a new solution, which is discussed in Section 4.2 and [Paper VII]. This section highlights some of the interesting results for the solution designed and developed here.

5.2.1 Experimental Setup

Details of the experimental setup are given in [Paper VII]. In summary, a Raspberry Pi 2 (Model B) was employed as an edge controller, and included the DCXP-enabled MediaSense. This was done in order to extend a Jaro-Winkler and Jaccard-like approach to DCXP. An arithmetic mean value is used for each metric. Time is measured in milliseconds (ms) and an interval of 10 ms is used for concurrent fetching of ConIn. Two scenarios for numerical data were chosen; the first included one controller, and ConIn values were randomized in each simulation, which was then fed into the algorithm, as described in Algorithm 1. In the second scenario, three edge controllers were used, and each controller fetched ConIn concurrently. Various similarity threshold values (referred to here as % similarity) are employed in different scenarios. Textual data are mostly found in virtual sensors, human sensing, and contextualized numeric raw data (e.g. temperature can be categorized as cold, warm,
etc.). In order to verify the feasibility of the Jaro-Winkler algorithm for an IoT edge controller, various text strings such as *temperature, light, blood pressure* etc. are used; these are then randomized, and similarity is calculated based on the list of available text strings as ConIn. The similar text strings are then grouped into clusters, and time is measured for different cluster sizes and % similarity. Furthermore, three different textual similarity algorithms are compared in order to evaluate their suitability at the edge of IoT. [Paper VII] evaluates numerical data clustering for both single and several controllers; the results for several controllers, that is, distributed clustering, are presented below, since this study concerns distributed clustering.

Table 5 presents the results for 50, 100, and 150 ConIn combined from three controllers with a 10 ms interval. At 70% similarity, when the ConIn has a greater spread, the standard deviation seems a little greater, but is constant at 90% similarity. With a greater spread of ConIn, cluster size varies, as seen in Figure 30. This is as projected for a real IoT implementation, since ConIn is distributed, and different ConIn is anticipated from different simulation runs. Based on the results for the proposed approach, it can be concluded that clustering the similar distributed ConIn from several controllers is realizable.

A performance comparison between a single edge controller and several edge controllers is shown in Figure 31. The behaviour of each scenario exhibits a similar pattern, with cluster size decreasing with higher % similarity as expected.

<table>
<thead>
<tr>
<th>% similarity</th>
<th>70</th>
<th>80</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster size</td>
<td>NConIn</td>
<td>µ</td>
<td>σ</td>
</tr>
<tr>
<td>50</td>
<td>16</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>100</td>
<td>34</td>
<td>2</td>
<td>17</td>
</tr>
<tr>
<td>150</td>
<td>35</td>
<td>2</td>
<td>19</td>
</tr>
</tbody>
</table>
The performance of textual data clustering is shown in Table 6, where three textual data clustering techniques were compared to determine the suitability of the proposed Jaro-Winkler approach. The results for similarity calculation times and cluster matching are shown in Table 6, which shows that Cosine has a similar performance to that of Jaro-Winkler, although with a higher standard deviation. On the other hand, Jaro-Winkler at times demonstrated a performance that was 19% better than that of Jaccard (cluster size of 30 ConIn) and for cluster size of 20 ConIn; on average, Jaro-Winkler showed better performance. Cosine showed a 50% higher standard deviation compared to Jaro-Winkler.
This section reports the evaluation of the proposed algorithms for clustering distributed context information with logically synchronized controllers in the IoT. The feasibility of this approach is verified for the Jaro-Winkler and Jaccard-like approaches for textual and numerical data, respectively, on Raspberry Pi and DCXP. The results suggest that the developed algorithm is able to cluster both numerical and textual distributed data on edge controllers. The elapsed time for clustering is stable, with 1 ms standard deviation irrespective of % similarity. These results further justify the use of the Jaro-Winkler similarity algorithms rather than other similarity algorithms.

### 5.3 Publish/Subscribe evaluation

Section 4.3 presented the design and development of the PubSub model proposed to enable the dissemination of context information and the logical sink. This section presents the performance measurements for this approach, which is evaluated and compared with previous approaches in terms of mostly quantitative aspects such as the number of event messages published, time required for subscription matching etc. This reflects one of the goals of this study, which is to disseminate context information fast and in real-time. The evaluation starts by reporting the performance of the extended MediaSense in relation to the existing MediaSense. The performance of the PubSub model can be divided into two parts: (i) PubSub for the context IDs shared in logical-clustering, for which each published context ID is matched for subscription; and (ii) PubSub for logical sink synchronization, for which all the changes are published to the other physical sinks. Performance is evaluated using three peers, with one peer acting as the host sink and the remaining two as recipient sinks. The mean measurements are presented below, and time is shown in milliseconds (ms).
5.3.1 Current vs. Extended MediaSense

In the current DCXP-enabled MediaSense, each context ID needs to be registered as a UCI for sharing context IDs, which incurs delay. To address this, MediaSense was extended by allowing the registration of several UCIs at the same time; this demonstrates a faster response, as can be seen from Table 7. It is therefore effective to register context IDs as context information and the sink as a UCI, which gives an improvement of nearly 74%, as demonstrated in Figure 32.

<table>
<thead>
<tr>
<th># of published context-IDs</th>
<th>Current MediaSense</th>
<th>Extended MediaSense</th>
<th>% improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>7.34 ms</td>
<td>4.17 ms</td>
<td>76</td>
</tr>
<tr>
<td>10000</td>
<td>8.93 ms</td>
<td>5.37 ms</td>
<td>66</td>
</tr>
<tr>
<td>100000</td>
<td>10.74 ms</td>
<td>6.23 ms</td>
<td>72</td>
</tr>
<tr>
<td>200000</td>
<td>11.65 ms</td>
<td>6.69 ms</td>
<td>74</td>
</tr>
</tbody>
</table>

Table 7: Required time for publishing

![Figure 32: Publishing time difference in MediaSense (current vs. extended)](image)

5.3.2 Publishing and subscription

After running the PubSub model for one second, the average message rate obtained is 3537 messages/sec for PubSub events. This high number of PubSub events is very useful in fulfilling the real-time and fast response requirements of the IoT. With an increase in the number of published items, the number of PubSub messages/sec decreases, as shown in earlier studies such as Le Subscribe [57]. However, the proposed approach with DCXP-enabled MediaSense outperforms Le Subscribe’s (Counting) approach, as shown in Table 8. Furthermore, the PubSub model reaches rates of around 2911, 1789, and 931 PubSub messages/sec for context ID sizes of 10K, 50K, and 100K, respectively. The number of PubSub messages/sec is reduced only by one third (33%), whereas the magnitude of the context ID increases tenfold (900%). This shows that the proposed approach exhibits the required scalability and
faster response for a high number of context IDs in real-time for the IoT. Figures 33(a) and (b) show the results for published items on two peers (controllers), and performance in terms of subscription matching, respectively. Figure 33(a) shows that average number of PubSub context-IDs/second decreases when the number of published items increases; however, when the number of published items increases from 10K to 20K - a 100% increase, average number of PubSub context-IDs/second increases by merely 3% for this particular increase. Number-wise, this is an increase of around 89 (from 2911 to 3000) when published items increases by 10K. Similar trend is seen when published items increased from 2K to 5K, and on both sinks. On the other hand, subscription matching increases by 86% for a hundred-fold increase in published items, which is nominal.

Figure 33: (a) PubSub messages per second and (b) subscription matching

Figure 34 shows the results of subscription matching for a single context ID, where the $i^{th}$ context ID is matched from $i$-size of the context ID. It took 8.76 ms to match the millionth context ID. Figure 34 (right) shows that for PARDES PubSub system [86], when subscriptions increase from 25K to 50K, 50K to 100K, and 100K to 200K, subscription matching increases by 54%, 89%, and 125% respectively. The rate of increase is large compared to that of the PubSub model designed and developed here, which increases by only about 7%. PARDES did not give results beyond 200K; if the smallest rate increase is taken into consideration and plotted, then PARDES overtakes MediaSense at 500K. The PubSub model in this work shows a 99% improvement over PARDES for two millionth context ID matchings. This outcome indicates that the PubSub model works well in a large-scale distributed scenario. Furthermore, scalability efficiency can be seen in Tables 8 and 9; the proposed PubSub model achieves increases of 2058% and 1200% in subscription matching and PubSub messages/sec, respectively.
Figure 34: Subscription time comparison

Table 8: PubSub messages/sec

<table>
<thead>
<tr>
<th># of context-IDs</th>
<th>Le Subscribe (Counting)</th>
<th>MediaSense</th>
<th>% improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 K</td>
<td>621</td>
<td>3151</td>
<td>407</td>
</tr>
<tr>
<td>1 million</td>
<td>7</td>
<td>91</td>
<td>1200</td>
</tr>
</tbody>
</table>

Table 9: Subscription matching

<table>
<thead>
<tr>
<th># of context-IDs</th>
<th>Le Subscribe (Counting)</th>
<th>MediaSense</th>
<th>% improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>500 K</td>
<td>85 ms</td>
<td>14.76 ms</td>
<td>476</td>
</tr>
<tr>
<td>1 million</td>
<td>350 ms</td>
<td>16.22 ms</td>
<td>2058</td>
</tr>
</tbody>
</table>

Matching for published items is not necessary for logical sink synchronization, although S2S communication and synchronization are necessary. If no matching operation is required, as many as 9032 event changes/second can be achieved. This is a further increase of about 155% in percentage terms (a factor of nearly three) compared to PubSub messages/sec. This number corresponds to M in Equation 1, which determines how many sinks and clusters can be processed per second. This very high number means that the proposed PubSub model is a very competent and efficient tool for the current and future IoT, particularly for the purposes of physically distributed clustering (logical-clustering).

This section has reported results for a PubSub model which is capable of satisfying the requirements of disseminating scalable and fast context information in real-time. The performance measurements are compared with earlier approaches, and this verifies the feasibility of the approach. Dynamism and prediction in PubSub are two vital characteristics of the current and future IoT, and these are explored in [Paper XI], although they are excluded from this dissertation. It was shown that the proposed PubSub model can address the challenges of dynamism and prediction.
5.4 Self-Organization Towards Enabling Autonomic Management of IoT

Section 4.4 described the design and development of self-organization support, with the aim of enabling the autonomic management of IoT to reflect the third research sub-question. This section presents a performance evaluation of the self-* capabilities that help in supporting the vision of autonomic computing. As described in Section 4.4 and [Paper IV], several algorithms are designed and developed here to achieve the goal of this section; furthermore, the DCXP protocol is extended using two new primitive functions, namely joinUCI and DISCOVER, which are deployed in the extended DCXP-enabled MediaSense platform. Further details can be found in [Paper IV] and [78].

5.4.1 Performance of self-* capabilities

The measured results are shown on a logarithmic scale in milliseconds; the mean $\mu$ and standard deviation $\sigma$ values are shown in the tables below. In terms of self-organization, the evaluation is divided into two parts; in the first, the time needed for self-healing (i.e. duplication checking and reconfiguration) is not considered, and only the time needed for self-configuration is considered. The second part includes the time required for self-healing. MediaSense incurs a delay if self-* algorithms are employed; this is understandable and expected of self-* algorithms, since an entity goes through the lifecycle of autonomic computing en route to becoming organized, i.e. a managed entity. The performance, however, remains on a par with that of the current MediaSense, as can be seen in Table 10.

Performance measurements of the algorithms for different scenarios are shown in Tables 10 to 12. Table 11 shows that discovery accuracy is very high (close to 100%) for all three cases simulated; however, this is measured when entities join serially, i.e. one after another. Simultaneous entity joining plays a role in obtaining the accuracy. Table 12 displays the discovery accuracy for 3000 and 2000 concurrently joining entities. The arithmetical mean values for this scenario are 1699 and 1195, respectively, and the standard deviations are 110.4513 and 97.4664; the discovery accuracy drops to 56.63% and 59.75% respectively. This fall in the discovery accuracy necessitates the design and development of a load-balancing and scheduling algorithm, which is left for future work. However, self-healing provides 100% accuracy for duplication checks. More performance evaluations are discussed in detail in [Paper IV].
Table 10: Entity-join performance

<table>
<thead>
<tr>
<th>Nodes discovered</th>
<th>MediaSense (Current)</th>
<th>MediaSense (Self-Configuration)</th>
<th>MediaSense (Self-Organization)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \mu )</td>
<td>( \sigma )</td>
<td>( \mu )</td>
</tr>
<tr>
<td>5000 nodes</td>
<td>1.59</td>
<td>0.0522</td>
<td>1.69</td>
</tr>
<tr>
<td>3000 nodes</td>
<td>1.69</td>
<td>0.0459</td>
<td>1.88</td>
</tr>
<tr>
<td>1000 nodes</td>
<td>1.88</td>
<td>0.0338</td>
<td>( \sigma )</td>
</tr>
</tbody>
</table>

Table 11: Entity discovery (dynamic)

<table>
<thead>
<tr>
<th>Nodes discovered</th>
<th>5000 nodes</th>
<th>3000 nodes</th>
<th>1000 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>99.8 %</td>
<td>99.9 %</td>
<td>100 %</td>
</tr>
</tbody>
</table>

Table 12: Discovery accuracy (concurrent)

<table>
<thead>
<tr>
<th></th>
<th>( \mu )</th>
<th>( \sigma )</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000 nodes</td>
<td>1699</td>
<td>110.4513</td>
<td>56.63 %</td>
</tr>
<tr>
<td>2000 nodes</td>
<td>1195</td>
<td>97.4664</td>
<td>59.75 %</td>
</tr>
</tbody>
</table>

5.5 Intelligence Evaluation at the Edge Controller

Section 4.5 introduced the concept of distributed intelligence in the IoT realm, where an edge controller was proposed to provide LLI while a cloud controller offers HLI. This section provides a performance evaluation of the proposed intelligence at the edge controller, with the aim of enabling distributed intelligence by employing AI techniques such as belief networks and reinforcement learning-like approaches. The following sub-sections report the performance of DAPs by applying belief networks and reinforcement learning to predict or learn actions; these were measured on a Raspberry Pi using the simulations as experiments approach. The simulation scenarios are detailed in [Paper V].

5.5.1 Applying Belief-Network

As new Things are connected every day in the ever-expanding IoT domain, the reliance on rules for providing intelligence, as has been the case until now,
faces a considerable scalability issue. Furthermore, rules only offer pre-assumed intelligence and break under uncertain conditions. For instance, in a SmartHome scenario, if rules are relied on for events related to a temperature sensor in a bedroom, 36(3x2x3x2) rules would be required to initiate any action, comparing every condition based on 10 ConIn values (see [Paper V] for details). If three more ConIn values are added, the number of rules required would become 108(3x3x4x3), a substantial 200% rise. This gives an idea of how a reliance on rules would be unviable for a particular controller to provide intelligence for thousands (or more) of Things. Table 13 and Figure 35(a) further illustrate this dependency on rules.

It can be seen from Figure 35(a) and Table 13 that the number of rules required rises by approximately 733% for an increase of only eight sets of ConIn (from 14 to 22). Rules usually follow the product rule (see Equation 4), while belief networks follow the sum rule (see Equation 5). The total number of rules required for each set of ConIn is equal to the total of number of sets of ConIn, plus one rule for each action taken by assigning prior belief. This shows how a belief network is more advantageous than a rules-based approach; this is also apparent from Figure 35(a).

Table 13: Advantages of Belief-Network over rule-based

<table>
<thead>
<tr>
<th>What</th>
<th>When</th>
<th>Motion</th>
<th>Location</th>
<th>Heating</th>
<th>Lighting</th>
<th>Total ConIn</th>
<th>Increase in ConIn</th>
<th>Total rules</th>
<th>% increase in rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>14</td>
<td>-</td>
<td>144</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>16</td>
<td>2</td>
<td>288</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>18</td>
<td>4</td>
<td>512</td>
<td>255</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>20</td>
<td>6</td>
<td>800</td>
<td>456</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>22</td>
<td>8</td>
<td>1200</td>
<td>733</td>
</tr>
</tbody>
</table>

\[ N_r = \prod_{i=1}^{n} N_{ConIn_i} \]  
\[ N_r = \sum_{i=1}^{n} N_{ConIn_i} + N_A \]

Where \( N_r \) = total number of rules, \( N_{ConIn} \) = number of ConIn, \( n \) = total number of context states, and \( N_A \) refers to the number of actions to be taken for \( N_{ConIn} \).

Moreover, Figure 35(b) demonstrates that a belief network exhibits faster response times compared to a rule-based approach. The results illustrate that
as the amount of ConIn increases, the belief network shows a low response time compared to a solely rule-based approach. The proposed approach achieves faster response times, which is one of the significant challenges in edge computing and in real-time IoT communication. The evaluation therefore confirms that the realization of a belief network for edge computing is both feasible and advantageous.

![Figure 35: (a) Rules increase (top); (b) Response time (bottom)](image)

Although it is clear that the belief network reduces the dependency on rules, it requires the assignment of prior belief, which the rule-based approach does not necessarily require. This is where learning comes into play; the assignment...
Learning and prediction are particularly useful when one or more of the beliefs and/or the ConIn required to execute a certain task are missing. In such scenarios, the edge controller needs to predict the missing values, possibly through learned experience. Figures 36 and 37 show how learning is achieved, as detailed in [paper V]. Figure 36 shows the results when no ConIn values are missing; the figure demonstrates how collected raw data is contextualized (TConIn). In the third of the four examples, the TConIn is \{warm, home, morning, yes\}, which can be interpreted as “the temperature is warm in the morning at home, and the user is awake”. The heating does not need to be activated in this example, since the heating probability is low at 0.36; however, the light should be turned on, since the lighting probability is high at 0.77. Based on the following TConIn examples, the following actuations that would be activated are lighting and breakfast (second and third TConIn), and lighting and dinner (fourth TConIn).

Figure 36: Simulated ConIn examples and probability for actuations

Figure 37 demonstrates the prediction result when a piece of ConIn is missing. For the example shown in Figure 36, the stored frequency and experience
(Algorithms 8 and 10) for each specific piece of ConIn are used to predict the missing values. Figure 37 shows that the available ConIn for the first tuple is: {home, bedroom, morning}. The probabilities of the missing temperature value are 0.41, 0.28 and 0.31 respectively for cold, comfort, and warm. Since “cold” has the highest probability value, the prediction algorithm predicts “cold” as being the most probable missing temperature for this specific ConIn tuple. Later, based on the available ConIn and the predicted values, actions are taken. Heating, lighting and breakfast actions are taken for the first ConIn tuple; this seems to be an accurate prediction since the user is at home in the morning. Thus, these actions are the most probable for activation. The result confirms that the algorithm learns to improve its tasks through experiences at the edge. The probabilities for prediction may be very close due to the randomness, and with more simulated values the probabilities may show a greater spread, as shown in [Paper V].

![Figure 37: Results of prediction for missing value](image)

5.5.3 MM-CAN Performance

Providing intelligence means applying knowledge, and this knowledge is harvested through reasoning. Section 4.6 introduced the concept of the MM-CAN, which is able to infer knowledge for different IoT applications at the edge controller instead of the cloud controller. A performance comparison between the cloud and edge controllers is therefore given below to demonstrate
the usefulness of this approach. The figure below confirms that MM-CAN responds faster at the edge controller than at the cloud controller. This again confirms the usefulness and novelty of the proposed approach by enabling the inferred knowledge to generate LLI at the edge with low latency.

Figure 38: Reasoning latency at the edge and in the cloud

5.6 Summary

This chapter has evaluated each of the proposed approaches for enabling DACIM. This evaluation demonstrated the usefulness and novelty of the proposed approaches, and it was shown, in line with the future IoT, that logical-clustering is realizable without encountering significant network degradation of scalability, reliability and reachability. The performance of IoT data clustering for both numerical and textual data was then reported for different cluster sizes and % similarity. The chapter then evaluated the PubSub model, for which fast subscription matching and high publishing rates were achieved to enable the scalable dissemination of IoT context information and to synchronize physically distributed controllers. The environment of the IoT becomes dynamic, requiring the system to organize itself, and self-* capabilities were therefore evaluated for both the concurrent and non-concurrent joining of Things. It was demonstrated that the system was able to stabilize itself and evolve correctly without a central point of failure, providing network stability and resilience, two of the main components of a system. Moving forwards, the IoT requires making sense of data, meaning that knowledge needs to be ap-
plied to reap value from the data. This can be addressed by providing intelligence at the edge. AI techniques such as Bayesian networks or belief networks were evaluated, and this confirmed that dependence on both rules-based intelligence and the cloud can be reduced with a faster response. Learning and prediction can also be provided at the IoT edge. The chapter ends by evaluating the approach to extraction of knowledge by means of MM-CAN; this implies that before knowledge is applied, it is feasible to infer this knowledge closer to the Things. The combined use of these approaches makes it possible to realize the proposed context-based DACIM.
6 Conclusions

This chapter summarizes the contributions of this dissertation in terms of addressing the challenges associated with the future IoT. Firstly, the research objectives are revisited in order to answer the research questions and reflect on the achievements made in this study. The chapter ends by suggesting future research directions which would further contribute to enabling fully functional autonomic computing in the IoT.

6.1 Discussion

Central to this dissertation is addressing the stumbling block which the IoT is expected to encounter as a result of the rapid proliferation of Things and the subsequent increase in context information. The fast-evolving IoT is expected to involve hundreds of billions of connected Things, which would constitute vast amounts of IoT data. As described in the Introduction, most of the ever-expanding IoT data is never analysed and utilized, and most of these data would remain underutilized if not managed properly; context-based solutions attempt to mitigate this problem. However, existing solutions have been deemed inefficient and infeasible to address current and future IoT challenges, and this mandates a new approach with regard to context-based solutions. Motivated by this, this dissertation addresses some of these challenges in order to close the gap in existing limitations. A DACIM approach is proposed by establishing two objectives: (i) efficiently utilizing and (ii) analysing context information. Thus, several artefacts based on the DSR method are designed and developed here, as described in Chapter 4, and these are spread over seven publications. Contributions from these publications help to realize the study undertaken in this dissertation, and address the research question. The following sub-sections give a summary of the achievements made in this dissertation by reflecting on each of the research sub-questions before addressing the overall research question.

The first research sub-question, which relates to the system model enabling a logical-clustering approach and reflecting performance in real-life scenarios, is answered primarily in Sections 4.1, 4.2, 5.1 and 5.2. A two-tier hierarchical distributed hash table (H-DHT)-based model, one of the contributions of this
dissertation, is proposed in which both edge and cloud computing are explored, as prior research has shown that centralized approaches fail to counter the scalability issue, among many other aspects. An implementation blueprint was also being laid down in [Paper I] and is verified in [Paper II], which investigates its performance. [Paper II] also evaluates a logical-clustering approach by means of ns-3 simulation in terms of mean delay and packet loss ratio, in order to assess the scalability, reliability and reachability issues. This particular evaluation helped to realize the feasibility of the proposed physically distributed clustering approach for real-life scenarios. This was followed by the demonstration of an algorithm capable of grouping similar distributed context information by means of Jaro-Winkler and Jaccard-like approaches; this was designed, developed and evaluated in [Paper VII]. This approach enables the clustering of both numerical and textual data, and is not dependent on a particular clustering similarity. It is able to cluster based on a choice given by a researcher, a system administrator or any service in which an IoT application might be interested. Furthermore, it has been shown that the approach not only permits clustering on a single edge controller but also on several controllers. The clustering algorithm demonstrated stability with a standard deviation of about 1 ms for elapsed time, irrespective of % similarity.

The second sub-question relates to the dissemination of scalable context information and is mainly answered in Sections 4.3 and 5.3. [Paper III] contributes largely towards the design and development of a PubSub model which addresses this particular sub-question. In terms of answering the sub-question, the artefact is designed and developed by extending a distributed protocol called DCXP. This allows us to move away from the central point of contact with respect to publishing and expressing subscription interest, which to some degree allows distributed PubSub model. The developed artefact demonstrates scalable (verified using two million PubSub items), fast real-time context information dissemination (taking 8.76 ms to match the millionth published item) and high publishing rates (3537 messages/sec average message obtained for PubSub event). The artefact is also capable of handling dynamism in the PubSub model. The evaluation of the model uses a comparison with prior approaches and the results indicate that the approach presented in this dissertation outperforms prior approaches in terms of fast subscription matching and high numbers of PubSub messages per second, as demonstrated in Section 5.3. Based on these results, it is evident that this approach helps to answer the second sub-question by enabling scalable and fast ConIn dissemination, as well as facilitating S2S synchronization with the help of a PubSub model verified on the extended DCXP-enabled MediaSense IoT platform.

The third research sub-question that this thesis addresses is answered in [Paper IV] and Sections 4.4 and 5.4 of this dissertation. The question of how to organize Things and controllers with minimal outside intervention is answered by exploiting the self-organization concept of autonomic computing, and three of the four self-* capabilities of autonomic computing are designed,
developed and evaluated here in order to support self-organization. Self-organization implies that an entity, that is, a Thing, should optimize itself according to policies set by outside sources. The results suggest that it is possible to structure a system such as the logical-clustering approach in an organized way, and that further correct evolution can be ensured with minimal intervention from outside sources. These self-* capabilities execute policies and organize a controller. This can be compared with the 5G vision whereby each subsystem self-organizes automatically and locally with distributed policies [87]. The DCXP protocol is further been extended here, and two new primitive functions are introduced to the existing DCXP. Its performance is on a par with the existing DCXP, and demonstrates very high accuracy rates for the non-concurrent joining of Things and duplication checks. The algorithms designed and developed here and these results help in realizing a self-organized system and can be used as a template to enable autonomic computing in the IoT.

The final research sub-question corresponds to the application of extracted knowledge, which is addressed in [Papers V and VI] and Sections 4.5, 4.6 and 5.5. Applying knowledge means providing intelligence. To comply with the current IoT focus, any IoT application needs to be capable of making sense of real-time IoT data, thus improving intelligence. This intelligence should be provided with low latency and should be able to handle uncertain situations. In response to this research sub-question, an FITC is proposed, whereby edge computing is used to address the low latency issue by distributing intelligence to the edge and the cloud; this also helps in reducing the dependency on cloud computing. This is also addressed by leveraging a belief network and a technique similar to reinforcement learning with the edge controller, and low-level intelligence is provided by inferring knowledge. Knowledge is inferred by designing and developing a multi-modal CAN, which is capable of reasoning various IoT applications. The performance evaluation demonstrates the feasibility and usefulness of the FITC approach with reduced requirements for rules and prediction of missing context information, as discussed in Section 5.5. The results further reaffirm that responses that are faster than those in cloud computing can be achieved by decoupling tasks such as contextualization, decision making, actions and predictions (DAPs), making sense of data, and harvesting knowledge from the cloud to the edge; this can aid in fulfilling the real-time requirements of the future IoT. These results can be useful for the vision of the 5G IoT [6], such as making sense of current data in real-time to personalize each IoT application, discovering actionable insights, automated decision making, and learning in real-time, as successfully demonstrated in this dissertation.
6.1.1 Summary of key achievements

The following is a summary of key achievements found in this study:

- The proposal of physically distributed clustering, i.e. logical-clustering to utilize context information, as opposed to physical clustering [Paper I]
- An algorithm capable of finding similarity for both textual and numerical data [Paper VII]
- Faster subscription matching and higher publishing rates [Paper III]
- Very high discovery accuracy for the non-concurrent joining of Things [Paper IV]
- A template for IoT autonomic computing [Paper IV]
- The concept of distributed intelligence, which uses both edge and cloud computing [Paper V]
- The leveraging of AI techniques to aid intelligence and learning at the edge [Paper V]
- A reduction in dependency on rules and the cloud, and faster responses [Paper V]
- Low-latency knowledge extraction [Paper VI]

6.1.2 Relations with Existing Solutions

This sub-section relates the achievements of this study with existing approaches. The proposed logical-clustering approach clusters based on similar context that are not pre-defined as compared to the earlier logical neighbourhood [47] approach. The logical-clustering approach also enables remote access to a cluster, i.e., grouped context information based on a requestor choice as opposed to ca4iot [26] which would return all the available context information to its requestor. The clustering information sharing has been achieved via the proposed PubSub model. This model can provide about 99% improvement compared to the PARDES model [86] with respect to subscription matching from 500K subscriptions and beyond. Furthermore, a distributed approach by employing both edge and cloud based solutions can help reducing over-reliance on the cloud based solutions that were proposed in most of the earlier IoT solutions. It is shown that in addition to Things management and system level intelligence at the edge [20], intelligence can also be extended to analysing data. Earlier lite data processing was explored but limited to data validation and notifications primarily [20]; and only rules were employed to extract knowledge at the edge. This study has demonstrated that AI techniques are realizable at the edge of the IoT which enable low-latency knowledge extraction compared to cloud, faster response and reduced dependency on the rules. This also enables automating several tasks at the edge such as those mentioned earlier, thus reducing human labour.
6.2 Future Work

The study in this dissertation has laid the foundation for the vision of autonomic computing in IoT, and the research thus far constitutes a step towards fulfilling this vision. However, the implementation of fully functional distributed intelligence and autonomic computing requires some additional research to be undertaken. This section aims to outline some future directions for the research presented here.

This dissertation has not discussed learning policies in the organization of controllers. A fully self-optimized system requires the adaptation of new policies, possibly via learning, and optimization of the system accordingly. Moreover, the autonomic management of the IoT would only be possible by exploring further policies. The integration of policies into the manager would require a more flexible and concrete manager. This could be achieved by deploying the Software-Defined Networking (SDN) concept and by incorporating a SDN-supported cloud controller with the presented IoT edge controller.

Furthermore, the penetration of Things is constantly increasing in the Internet of Things domain, and this trend is expected to persist, causing complexities in the network and difficulties in managing the current and future IoT network. Things should be expected to join and leave the network at any time, and the network needs to adapt and to develop network-wide policies for adoption [50]. SDN aims to address the above mentioned challenges by providing a logically synchronized view of the overall network. The existing and future IoT invite heterogeneity; this heterogeneity is not limited to Things, but also involves the network in which entities are immersed. Connecting these heterogeneous Things and networks gives rise to many complexities and difficulties, although these could be addressed by exploring SDN in the IoT [80]. The use of SDN in the IoT was also outlined by Zhang [88]; an SDN-supported IoT is therefore an interesting topic for study.

5G wireless/mobile broadband is poised to incorporate SDN in the cloud and further enrich the IoT [6, 87]. The FITC cloud controller proposed in this dissertation could therefore incorporate 5G-supported SDN with the aim of offering high-level intelligence along with network function virtualization (NFV). This incorporation of SDN and NFV in the cloud controller would further enable programming of the heterogeneous IoT network to help realize autonomic computing, by deciding who needs what and when with respect to the varying capabilities and data demands of the future IoT. One such variable capability involves deciding when to invoke the cloud controller, since the edge controller cannot provide all the capabilities an IoT application might require. For example, in view of the billions of Things that are expected to be connected, not all IoT data should be forwarded to the cloud. A cloud controller, by virtue of its higher capabilities, can decide by itself when an IoT appli-
cation should forward data. This automatic determination of data traffic control and decisions is a driving factor of the 5G IoT [6] and should be explored further.

5G not only requires instantaneous learning and actionable insights, which this dissertation has already demonstrated, but also necessitates an assessment of data to extract understanding and value using data science algorithms, and this is an attractive area of study [6]. Another 5G vision that is expected to play a role in the IoT is D2D communication, bypassing any central server for which the proposed logical sink (S2S) communication can be extended in order to reflect the 5G concept.

The dissertation introduces the concepts of the MM-CAN and FITC. In future, to complement these, it would be interesting to investigate reasoning accuracy by exploring different reasoning techniques for a particular IoT application and for each DAP operation. The accuracy can be further investigated to identify which reasoning technique suits which IoT application. It would also be interesting to investigate the efficiency and soundness of the interoperability between the edge and cloud controllers, to determine the cross-platform behaviour of the envisioned FITC. The dynamic behaviour of the controllers could also be investigated. A CONtext LEAmer (CONLEA) which would enable a controller, for example an edge controller, to learn context by exploiting Thing’s context information and its associated context information may be an interesting research topic.

The results for the discovery of concurrent Things, as illustrated in Section 5.4, imply that the concurrent joining of Things incurs a performance penalty due to competition for resources within the DCXP-enabled MediaSense platform. This can be countered by exploring load-balancing and scheduling algorithms for controllers using SDN support.

The unique identification of each Thing is another important and open research issue in the IoT domain which needs to be addressed. However, the unique identification billions of Things is not easy to implement [89]. The P2P infrastructure of the MediaSense platform adopted in this dissertation exploits $2^{160}$ unique IDs; this is more than IPv6 can provide, and this number is sufficient for hundreds of billions of unique IDs. However, these IDs are very difficult for a human to remember, and a UCI can be a potential solution. Moreover, in logical-clustering, context IDs need to be unique and humans should be able to access these context IDs easily. It is therefore necessary to define a naming scheme (potentially a hierarchical scheme) in conjunction with the UCI, which will ensure the unique identification of entities in the IoT realm.

Security, which is beyond the scope of this study, is one of the other challenging issues in the current and future IoT; similar to intelligence, this cannot be solved by cloud computing alone, and must be addressed for a successful worldwide adoption of the IoT. Over-reliance on the cloud for security gives rise to the challenges of a central point of failure and scalability issues, which
can affect authorization and authentication [90]. Security in the IoT needs to work for resource-constrained devices as well as data security without user intervention; this can be handled by extending the proposed FITC to use local security at the edge for Things and local data, and global security in the cloud for edge controllers and global data. The proposal in [90,91] using Auth is appropriate for the DACIM proposal and can be leveraged to respond to the security challenges both at the edge and in the cloud. Auth also supports secure communication for the Publish/Subscribe process, which is in line with the DACIM proposal. Secure S2S and edge-to-cloud communication (and vice versa) can be investigated by extending existing security approaches such as Auth to enable heterogeneous and scalable security. Furthermore, dynamic authorization and authentication for Things can be supported by extending the proposed self-organization approach.
References


[69] [online] Constructivism Research Philosophy: https://research-methodology.net/research-philosophy/epistemology/constructivism/ [Last Accessed: September 2017].


[76] Ns-3 discrete-event network simulator for Internet systems Link: https://www.nsnam.org/.


