

TECHNICAL REPORT

# Montreal Smart-City Pilot System (MSCPS)

*submitted to*

Ville de Montréal,  
275, rue Notre-Dame Est, Montréal, Québec, H2Y 1C6

*by*

Broadband Communications Research Laboratory (BCRL)  
Department of Electrical & Computer Engineering  
McGill University  
3480 University Street, Montreal, Quebec H3A 0E9

**July 2017**

# Table of Contents

Table of Contents .....	i
List of tables.....	v
List of Figures .....	vi
Chapter 1 Executive Summary .....	1
Chapter 2 Smart-City/IoT Architectures and Issues .....	3
2.1 Motivations and Current Trends of the Internet of Things for Smart Cities .....	3
2.1.1 Motivations .....	3
2.1.2 Current trends of the Internet of Things for Smart Cities .....	4
2.1.3 Notable Examples of Smart City Applications around the World .....	4
2.2 IoT System Architecture and Enabling Technologies for Smart City .....	9
2.2.1 Data Acquisition Layer .....	10
2.2.2 Connectivity Layer.....	11
2.2.3 Data Management Layer .....	12
2.2.4 Application Layer .....	13
2.3 Open issues of Smart City based on IoT.....	13
2.3.1 Scalability .....	13
2.3.2 Virtualization .....	14
2.3.3 Security and Privacy .....	14
2.4 Conclusion .....	14
Chapter 3 Montréal Smart City Pilot System (MSCPS) .....	15
3.1 MSCPS deployment summary (Lot 1, 3, 4).....	15
3.1.1 Lot 1 – Intelligent traffic.....	15
3.1.2 Lot 3 – Urban Asset Management .....	17
3.1.3 Lot 4 – Public Security .....	18
3.2 VdM pilot deployment network structure.....	19
3.2.1 Network connectivity diagram.....	19
3.2.1 Physical deployment network structure .....	21
3.3 Device capabilities .....	22
3.3.2 Communication devices .....	22
3.3.3 Sensing devices .....	26

---

3.3.4 Mobile sensors .....	28
3.4 Configuration summary .....	33
3.4.1 Time synchronization.....	33
3.4.2 IP addressing and NAT .....	33
3.4.3 WiFi radios.....	34
3.4.4 Cameras.....	35
3.4.5 RFID .....	35
3.5 Throughput and device reliability test results .....	35
3.5.1 WiFi Radios throughput tests.....	35
3.5.2 Zigbee Radios throughput tests.....	38
3.5.3 Device Reliability .....	39
3.5.4 Road surface and tree inspection vehicle field test .....	40
3.6 Planning estimation for Montreal downtown and the whole Montreal.....	41
3.6.1 Assumptions and the investigated scenarios .....	41
3.6.2 Simulation results and discussions.....	43
3.7 Conclusion .....	45
Chapter 4 Data Flow, Storage and Control Structures.....	46
4.1 The data management structure .....	46
4.2 On-premises management structure for the MSCPS .....	49
4.2.1 Data Plane .....	49
4.2.2 Control Plane .....	59
4.3 Cloud-based data management structure for the MSCPS .....	66
4.3.1 Introduction to Microsoft Azure .....	66
4.3.2 Data Plane .....	69
4.3.3 Control Plane .....	75
4.4 Discussions and challenges.....	76
4.4.1 Comments of current deployment.....	76
4.4.2 Challenges.....	77
4.5 Chapter summary .....	77
Chapter 5 Security.....	79
5.1 The importance of IoT security, examples and common threats .....	79
5.1.1 The importance of IoT security:.....	79
5.1.2 IoT security vulnerability examples.....	80
5.2 Guidelines and best practices for Smart City planning and implementation with regard to security aspects.....	83

---

5.2.1 Key security requirements .....	83
5.2.2 Security considerations in Implementation.....	84
5.2.3 Security considerations in Operation and Maintenance.....	85
5.2.4 Security considerations in disposal of Smart City devices .....	85
5.3 Summary of security features that are enabled in the pilot deployment .....	86
5.3.1 Radios .....	86
5.3.2 Cameras.....	87
5.3.3 Adapters .....	89
5.3.4 LTE Gateway .....	89
5.3.5 Non-IP Devices .....	90
5.3.6 Data at rest Security .....	90
5.4 Challenges.....	91
5.4 Conclusion .....	92
Chapter 6 Data Analytics – Example Functions and Results.....	93
6.1 Motivations, benefits, and applications.....	93
6.1.1 Motivations and benefits of data analytics for Smart City.....	93
6.1.2 Promising applications from data analytic applications.....	94
6.2 Example data analytic functions and results .....	94
6.2.1 Parking place occupancy and vehicle counting application.....	94
6.2.2 Crowd counting.....	101
6.3 Challenges.....	104
6.4 Conclusion .....	105
Chapter 7 Conclusions and Recommendations.....	106
7.1 Data Acquisition Layer .....	108
7.2 Connection/Communication Layer:.....	109
7.3 Management Layer .....	110
7.4 Application Layer .....	110
7.5 Security Issues .....	111
References.....	113
Appendix A: Replies to Questions & Comments from VdM.....	A1
Appendix B: Montréal Smart City Pilot Deployment Implementation details.....	B1
Appendix C: Datasheets of Deployed Sensors and Devices.....	C1
Appendix D: Details on Device Selection, Deployment Procedures, and Guidelines.....	D1



---

Appendix E: Database Integration.....	E1
Appendix F: Monitoring and Control of the IoT network.....	F1
Appendix G: IoT Security Implementation in MSCPS.....	G1
Appendix H: Data Analytic Application – Parking place occupancy and vehicle/pedestrian counting.....	H1
Appendix I: Data Analytic Application – Crowd Counting.....	I1
Appendix J: VdM WiFi Network Planning.....	J1
Appendix K: Commercial Data Analytic Trials .....	L1

# List of tables

<b>Table 2.1:</b> IoT Applications in Smart Cities. ....	8
<b>Table 3.1:</b> Summary of proposed tasks and achievements in Lot 1- Intelligent traffic. ....	16
<b>Table 3.2:</b> Summary of proposed tasks and achievements in Lot 3- Urban Asset Management.....	17
<b>Table 3.3:</b> Summary of proposed tasks and achievements in Lot 4 – Public Security.....	18
<b>Table 3.4:</b> Device naming convention codes .....	20
<b>Table 3.5:</b> Location code and real deployed location.....	20
<b>Table 3.6:</b> Ubiquiti radios used in the deployment project.....	23
<b>Table 3.7:</b> Zigbee radio used in the deployment project.....	23
<b>Table 3.8:</b> Serial adapters used in the deployment project .....	24
<b>Table 3.9:</b> Cisco Miraki MX84 device. ....	24
<b>Table 3.10:</b> Camera functional specifications.....	26
<b>Table 3.11:</b> Traffic radar specifications.....	27
<b>Table 3.12:</b> RFID reader specifications .....	27
<b>Table 3.13:</b> Sierra Wireless Airlink GX450 specifications .....	29
<b>Table 3.14:</b> Massa RoadWatch SS Specifications.....	30
<b>Table 3.15:</b> Massa M3 Level Sensor specifications.....	30
<b>Table 3.16:</b> Road surface camera. ....	32
<b>Table 3.17:</b> Tree inspection camera. ....	32
<b>Table 3.18:</b> Lab throughput test results on Zigbee radios. ....	38
<b>Table 3.19:</b> Throughput test results on Zigbee 900MHz radio. ....	39
<b>Table 3.20:</b> Road surface and tree inspection vehicle field test storage results.....	40
<b>Table 3.21:</b> Summary of simulation results for Downtown Montreal and Quartier de l’Innovation area. ....	44
<b>Table 4.1:</b> Formatting of traffic radar reports. ....	55
<b>Table 4.2:</b> Amount of data upload from different types of sensors.....	74
<b>Table 4.3:</b> Data retention prediction for all currently live sensors.....	74
<b>Table 5.1:</b> Common Security threats and countermeasures [69].....	82
<b>Table 5.2:</b> Security features that are enabled on Radio devices. ....	86
<b>Table 5.3:</b> Security features that are enabled on Cameras. ....	88
<b>Table 5.4:</b> Security features that are enabled on Adapters.....	89
<b>Table 5.5:</b> Security features that are enabled on Adapters.....	90
<b>Table 7.1:</b> Summary of the project and recommendations on what can be done next and be further studied. ....	106

# List of Figures

<b>Figure 2.1:</b> Smart Parking design and physical installation in Smart Santander project. ....	5
<b>Figure 2.2:</b> Smart waste management helps optimizing the rubbish pickup route. ....	6
<b>Figure 2.3:</b> Underground acoustic sensors on water pipeline [14]. ....	6
<b>Figure 2.4:</b> New York City 24/7 Smart Screen [16]. ....	7
<b>Figure 2.5:</b> Smart City IoT multi-layered system architecture. ....	10
<b>Figure 3.1:</b> The complete network connectivity diagram of the deployed devices. ....	19
<b>Figure 3.2:</b> Camera and wireless antenna deployed on a light pole. ....	21
<b>Figure 3.3:</b> Physical map of the deployment structure ....	21
<b>Figure 3.4:</b> NAT setup in the pilot deployment. ....	25
<b>Figure 3.5:</b> Actual NAT setups. ....	25
<b>Figure 3.6:</b> Mobile sensors connection diagram for the salt truck. ....	28
<b>Figure 3.7:</b> Installations of devices on the salt truck. ....	29
<b>Figure 3.8:</b> Block diagram of road surface and tree inspection vehicle setup. ....	31
<b>Figure 3.9:</b> Road and tree inspection vehicle camera installation setup. ....	31
<b>Figure 3.10:</b> Scripts that enables and maps network address translations. ....	33
<b>Figure 3.11:</b> Configuration page for wireless settings. ....	34
<b>Figure 3.12:</b> Data transmission setup of UHF RFID devices. ....	35
<b>Figure 3.13:</b> Data transmission setup of BLE RFID devices. ....	35
<b>Figure 3.14:</b> Throughput test setup. ....	36
<b>Figure 3.15:</b> Average and peak throughput results in Lab environment. ....	36
<b>Figure 3.16:</b> Spectrum analysis from Ubiquiti web GUI from an installed device in downtown Montreal. ....	37
<b>Figure 3.17:</b> Comparison of Encrypted vs. Non-encrypted Deployed Throughput. ....	37
<b>Figure 3.18:</b> Comparing WiFi radios in laboratory and deployed throughput with 10MHz channel width. ....	37
<b>Figure 3.19:</b> Zigbee throughput test setup. ....	38
<b>Figure 3.20:</b> Device Reliability measured by Ping. ....	39
<b>Figure 3.21:</b> Footage from street inspection camera (top) and tree inspection camera (bottom) for both left and right channels. ....	40
<b>Figure 3.22:</b> Signal strength coverage of 802.11n based wireless networks in Scenario 1. ....	41
<b>Figure 3.23:</b> Signal strength coverage of high-density 802.11ac-based wireless networks in Scenario 2. ....	42
<b>Figure 3.24:</b> Signal strength coverage of low-density 802.11ac-based wireless networks in Scenario 3. ....	42
<b>Figure 3.25:</b> Downtown and Quartier de l'Innovation area overview. ....	44
<b>Figure 4.1:</b> Basic Data Management Platform for Smart City. ....	47
<b>Figure 4.2:</b> Video data flow for On-premises setup. ....	49
<b>Figure 4.3:</b> CPU and memory consumption of the candidate software at different resolutions. ....	51
<b>Figure 4.4:</b> Network bandwidth consumption of the candidate software at different resolutions. ....	51
<b>Figure 4.5:</b> Storage size and frame rate of video archives by different solutions. ....	51
<b>Figure 4.6:</b> CPU and memory consumption of the candidate software at different number of streams. ....	52

<b>Figure 4.7:</b> Network bandwidth consumption of the candidate software at different number of streams.	52
<b>Figure 4.8:</b> On-premises data management.	54
<b>Figure 4.9:</b> Types of socket connections.	54
<b>Figure 4.10:</b> Raw data from RFID-UHF reader.	56
<b>Figure 4.11:</b> Raw data from RFID-BLE reader.	56
<b>Figure 4.12:</b> Raw data from temperature sensors.	56
<b>Figure 4.13:</b> Raw data from Ultrasonic Level Sensors.	56
<b>Figure 4.14:</b> Raw data from GPS sensors.	57
<b>Figure 4.15:</b> Block structure of documents in RFID Collection.	58
<b>Figure 4.16:</b> Visualizing salt truck data.	58
<b>Figure 4.17:</b> Speed histogram visualization from radar sensor data.	59
<b>Figure 4.18:</b> On premises device control management.	60
<b>Figure 4.19:</b> Ubiquiti AirControl2 GUI.	61
<b>Figure 4.20:</b> Axis Camera Management Client.	61
<b>Figure 4.21:</b> HikVision iVMS-4200 Client.	62
<b>Figure 4.22:</b> Panasonic ASM200 Operation Software.	62
<b>Figure 4.23:</b> Device information fields and relation implemented in the database.	63
<b>Figure 4.24:</b> Entries from csv offline log file.	64
<b>Figure 4.25:</b> ONVIF command to perform reboot.	64
<b>Figure 4.26:</b> Central Management GUI.	66
<b>Figure 4.27:</b> Overview of device connectivity and dataflow via IoT Hub.	67
<b>Figure 4.28:</b> Overview of video streaming data flow via Azure Media Services.	68
<b>Figure 4.29:</b> Video Data flow with Stratocast.	70
<b>Figure 4.30:</b> Bandwidth usage of Stratocast in comparison to on-premises solutions.	71
<b>Figure 4.31:</b> Frame rate and storage size of Stratocast in comparison ton on-premises solutions.	71
<b>Figure 4.32:</b> Video recording quality comparison of Stratocast and FFmpeg.	72
<b>Figure 4.33:</b> Cloud-based data management.	73
<b>Figure 4.34:</b> Number of approaching and receding vehicle at different time in a day.	74
<b>Figure 4.35:</b> Cloud-based device control management.	75
<b>Figure 5.1:</b> A smart device running modified firmware [58].	80
<b>Figure 5.2:</b> Hacking into a road display.	81
<b>Figure 5.3:</b> Infrastructure Dependency Matrix.	84
<b>Figure 6.1:</b> Parking place occupancy output frame.	95
<b>Figure 6.2:</b> Parking detection log file.	95
<b>Figure 6.3:</b> Parking detection image processing block diagram.	96
<b>Figure 6.4:</b> Vehicle counting application.	96
<b>Figure 6.5:</b> Vehicle counting log file.	97
<b>Figure 6.6:</b> Vehicle counting image processing block diagram.	97
<b>Figure 6.7:</b> Example of good camera setup for parking detection application.	98
<b>Figure 6.8:</b> Camera looing down a street.	99
<b>Figure 6.9:</b> Top-down view.	99
<b>Figure 6.10:</b> Isometric view.	100
<b>Figure 6.11:</b> Error detection of the two applications at different resolutions.	100

---

<b>Figure 6.12:</b> Error detection of the two applications at different frame rates. ....	101
<b>Figure 6.13:</b> Heads found by the proposed method. ....	101
<b>Figure 6.14:</b> Block diagram of the proposed method.....	102
<b>Figure 6.15:</b> Expected camera setups (left) and camera setup with too high inclination (right).....	103
<b>Figure 6.16:</b> Crowd count error percentage of the proposed algorithm. ....	103

# Chapter 1

## Executive Summary

This Technical Report<sup>1</sup> presents the technical aspects of the work done during January 6- September 30/2017 for the City of Montreal (Ville de Montréal, VdM) to study, develop, design, implement, and deploy the Montreal Smart City/Internet of Things (IoT) pilot system (MSCPS) for applications in *Intelligent Traffic (Circulation intelligente, Lot 1)*, *Urban Asset Management (Gestion des actifs urbains, Lot 3)*, and *Public Security (Sécurité publique, Lot 4)*.

For applications in *Intelligent Traffic (Circulation intelligente, Lot 1)*, and *Public Security (Sécurité publique, Lot 4)*, the deployed MSCPS includes 6 Full/Ultra HD IP cameras with 3 microphones, and 9 traffic radar sensors installed at 6 locations in the Quartier des Spectacles.

For applications in *Urban Asset Management (Gestion des actifs urbains, Lot 3)*, the deployed MSCPS includes (i) 1 GPS, 2 temperature sensors and 1 level sensor installed on the salt truck, (ii) 3 specialized high speed cameras installed on a city van for tree and road condition monitoring, (iii) both passive and active RFID systems installed in 2 locations.

The deployed MSCPS also provides designed wireless infrastructure with 11 Ubiquiti WiFi radios using WiFi 801.11n and 802.11ac technologies for wireless transport of broadband video data to optical-fiber backbone network and communication links from the mobile devices by using WiFi and LTE technologies.

Furthermore, the deployed MSCPS includes developed software functions for sensor networking, data connections to the data center of the City of Montreal through optical-fiber backbone network with support for Internet and network protocols such as IPv4, IPv6, TCP/UDP. We have deployed and tested various private and cloud applications for video management, investigated the possibility of integrating communications and storage to the cloud. To investigate the operation and performance of the deployed MSCPS in potential applications, we have also selected a number of reliable, commercial as well as free software/libraries and performed preliminary tests for computer-vision analytics applications over the deployed MSCPS, including parking space monitoring, vehicle/pedestrian counting, and crowd counting.

In addition, several comments on the draft technical report and concerns/ questions about the design and implementation of Smart City from the VdM team are compiled and point-by-point directly addressed in Appendix A with reference to the appropriate chapters/sections/appendices for details.

Based on these concerns and the obtained results on the successfully deployed MSCPS, the rest of this technical report is organized as follows.

---

<sup>1</sup> This TR has been revised in response to comments from the VdM team.

---

Chapter 2 provides an overview on the overall Smart City/IoT architectures and the issues, serving as a reference/background for the descriptions of the successfully deployed MSCPS in Chapter 3.

Chapter 3 presents the design approaches, architecture and elements of the deployed Montreal Smart City/Internet of Things (IoT) pilot system (MSCPS) with the following supporting Appendices:

- Appendix B on the *detailed implementation* of the deployed MSCPS.
- Appendix C on *datasheets* of the sensors/devices in use.
- Appendix D on detailed *device selection, deployment procedures and guidelines*.
- Appendix J on VdM Wifi Network Planning.

Storage and data flow control structures, and security are among the most important aspects of the Smart-City/IoT networks. For this, we dedicate 2 following chapters for more detailed discussions in context of the deployed MSCPS.

Chapter 4 focuses on the storage and data flow control structures of deployed MSCPS with the following Appendices:

- Appendix E illustrating the database integration in the deployed MSCPS.
- Appendix F illustrating the Monitoring and control of the IoT network.

Chapter 5 discusses the security aspects in IoT, guidelines, best practices as well as emerging security standards and open issues. Appendix G further describes the security measures that have been deployed.

Chapter 6 discusses example data analytics with detailed results and implementation presented in Appendix H for *parking place occupancy, vehicle/pedestrian counting* and Appendix I for *crowd counting*.

Finally, chapter 7 concludes the report and comments on the future work and extensions.

# Chapter 2

## Smart-City/IoT Architectures and Issues

### 2.1 Motivations and Current Trends of the Internet of Things for Smart Cities

#### 2.1.1 Motivations

The Internet of Things (IoT) is a recent communication paradigm where objects are equipped with information and communication technologies (ICT) and become integrated with the Internet [1]. As such, the concept of IoT typically consists of 2 key components: the *Things* and the *Internet*. The Things usually refers to the *smart objects* – ordinary objects that become “smart” by integrating with advanced technologies to enable identification, communication, awareness and interaction capabilities. Pioneered by the research and applications of Radio-Frequency Identification (RFID) and wireless sensor network (WSN), the concept of smart objects in IoT today extends to a wide variety of devices, including home appliances, surveillance cameras, smart phones, and monitoring sensors [2]. The second key component - the Internet, provides the means of connecting and integrating of various heterogeneous networks of smart devices to the global Internet, allowing these smart devices to be reachable from anywhere and anytime [3]. The embedding of intelligence and connectivity into physical devices facilitates the collaboration with users as well as the interactions among these smart objects. These capabilities open up a new horizon of revolutionary applications from environmental monitoring, automation, context-aware control and data analysis that promise to enhance efficiency, reduced costs and improve the quality of human life. The coverage of these applications is virtually limitless and spans across many domains, such as home automation, smart grids, traffic management, and Smart City.

Lately, the application of IoT to Smart City has particularly gathered significant interest due to the unprecedented trend of the world population moving to live in urban areas. Many city governing bodies have been promoting adoption of ICT solutions in pursuit of the Smart City concept for sustainable development. Though a formal definition of Smart City has not been available yet, from a technical perspective, the general aim of Smart City is to utilize ICT and the Internet in order to better manage public resources efficiently and more importantly, to improve the quality of life for Smart City citizens [2]. To achieve Smart City objectives, IoT can bring a number of benefits to various Smart City applications and services, such as smart grid, intelligent traffic, waste management, environment monitoring, and smart surveillance [3]. Through its extensive sensor networks, IoT deployment can supply a huge amount of data throughout the city in real-time, providing insights into city operations and policymaking process. Additionally, a unified network infrastructure of IoT can facilitate communications and interactions among multiple applications and services, allowing more effective resource sharing and information exchange [3]. It is especially beneficial when data in a specific domain can potentially support smart applications in other domains as well [4], greatly reduce the infrastructure deployment cost and functionality overlapping. For instance, video surveillance footage can be used for traffic management applications in



addition to maintaining public security. Thus, applying IoT concept and practices is extremely attractive to city administrators for their Smart City.

### 2.1.2 Current trends of the Internet of Things for Smart Cities

The applications of IoT technologies in Smart Cities are typically in 3 aspects: smart mobility, smart sustainability, and smart living.

**Smart mobility** refers to the use of IoT applications to enhance transportation and logistics operations, improve commuting efficiency, and reduce gas emissions, thus improving traveling experience for citizens throughout the city as well as reducing the pollution caused by vehicles. Examples of such applications encompass IoT-enhanced public transportation systems, smart parking monitoring and effective traffic management.

**Smart sustainability** aims to reduce environmental impact of energy consumption and pollution from city operations. The emphasis is put on monitoring, managing, and distributing resources such as waste, water, and electricity efficiently and dynamically according to demand. IoT applications such as intelligent lighting, environment monitoring, smart irrigation system, waste management, and smart grids are good examples of this category.

**Smart Living** often involves using IoT devices for enhancing quality of life, preventing and minimizing the risk and impact of adverse events such as crimes, accidents, as well as making the Smart City safer and more attractive to residents. Applications of this category include video surveillance for public security and interactive kiosks that provide users with location-based information feeds and advertisements.

Other aspects of Smart Cities, such as smart governance, smart people, and smart economy [5] also benefit from data produced by IoT applications, but are beyond the scope of this work/report.

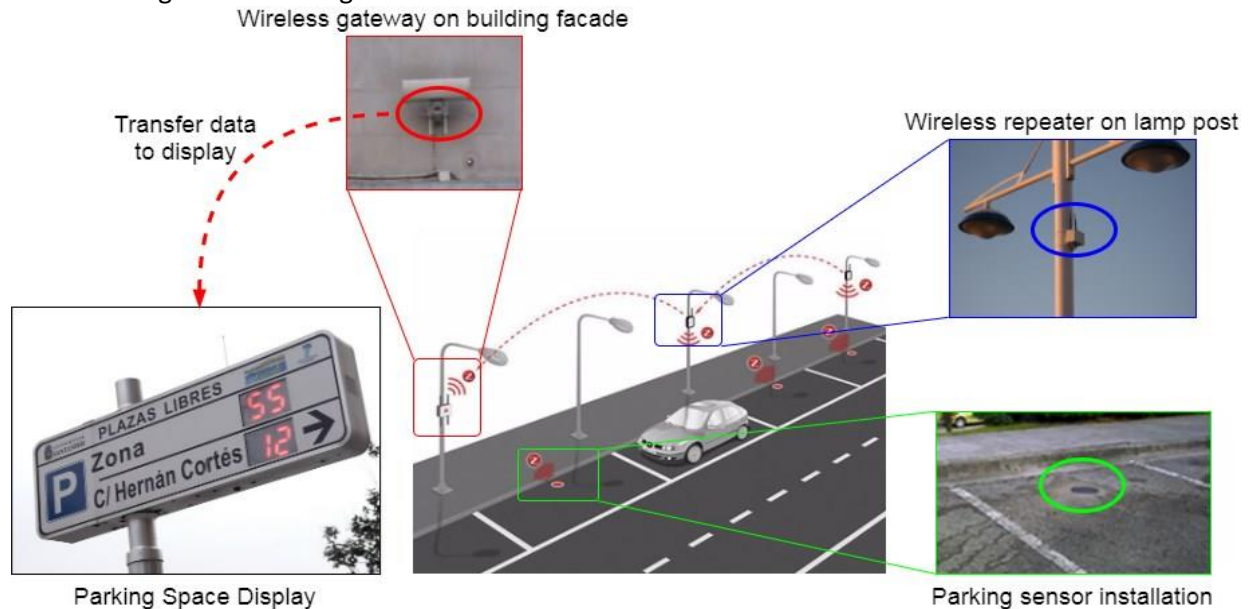
### 2.1.3 Examples of Smart City Applications around the World

Due to these many benefits, Smart City concept over an IoT platform has been adopted and implemented in many metropolitan areas around the globe. In this section, some representative examples of smart cities around the world will be discussed to highlight their smart applications utilize IoT technology to achieve the objectives of smart mobility, smart sustainability, and smart living in the Smart City paradigm.

#### 2.1.3.1 Smart Mobility Projects

- **Barcelona, Spain.** Barcelona is one of the first cities to push for Smart City model with many IoT-based initiatives, particularly the mobility projects aim to improve traveling experience for citizens through use of sensors technologies. Parking spaces are equipped with wireless sensors to notify and guide drivers to available spaces through mobile apps, reducing unnecessary commuting to find parking. In terms of public transport, city buses are equipped with wireless GPS sensors to monitor their locations, which can be accessed through interactive electronic displays mounted at smart bus stops to present information automatically to passengers about bus arrival and departure timing, thus improving customer experience while waiting for a bus [6].
- **Beijing, China.** A smart traffic management system with 157 high-definition cameras on Beijing's surrounding expressways was deployed to automatically count vehicles and provide traffic flow statistics, as well as automatically record any events such as accidents when they occur and alarm authorities as necessary [7]. At the same time, tens of thousands of traffic flow detectors are installed in expressways and near intersections to automatically collect traffic flow, speed, and density data, which is processed by a traffic control system to automatically adjust traffic signals according to the number of cars on the road. Overall, the smart traffic management system has reduced Beijing's traffic congestion by up to 60% and doubled its road capacity, reducing driving time and fuel consumption. In particular, Beijing's 5th Ring Road, 80% of intersections are regulated by this system, which has increased its road capacity by 15% [7].
- **Santander, Spain.** The Smart Santander project started in 2010, enabled installations of thousands of devices throughout the city of Santander to provide an urban city-scale real-world IoT experimentation

facility [8], [9]. One highlight application of the project is a smart outdoor parking management service that deployed over 400 ferromagnetic wireless sensors to monitor parking availability in the city center area. The sensors, equipped with 802.15.4 radio module for communication, are buried under the asphalt at each parking space, which limits the devices exclusively to battery power and wireless communication. Thus, wireless relay nodes and gateways are deployed in the area to provide connectivity to the Internet so parking occupancy data can be delivered quickly to drivers and traffic control system. Sensor data can be displayed on a smart phone app or electronic panels located at street intersections to indicate how many free parking spaces are on that street. The project offers a city-scale testbed for experimentation with IoT technologies as well as many insights in deployment and management of a large-scale IoT infrastructure.



**Figure 2.1:** Smart Parking design and physical installation in Smart Santander project.

### 2.1.3.2 Smart Sustainability Projects

- Amsterdam, Netherlands.** Since 2006, many projects have been initiated in Amsterdam, Netherlands, progressing toward Amsterdam Smart City. One of the notable and highly successful projects is a smart lighting project in collaboration with Philips, Cisco, and Aliander to install a connected lighting system and public Wi-Fi connection at Hoekenrodeplein, near Amsterdam Arena [10]. With lighting accounting for 19% of all electricity consumed, the goal is to apply adaptive lighting with smart controllers, allowing automatic adjustment based on different circumstances, such as dimming during low traffic hours, or boosting for public events, thus optimize energy usage and reduce unnecessary consumption [11]. The project is expected to generate about 130 billion euro in energy savings while providing better utility for citizens [12]. Another high impact project in Amsterdam is Climate Street initiative which aims at improving energy management and the efficiency of public services such as waste collection. The waste bin are equipped with wireless level sensors for status report so that the waste is only collected when the bins are full. The city also equipped connected electricity meters for dynamically matching of electricity demands. The project help Amsterdam to reduce the annual CO<sub>2</sub> emission of the commercial district by 62% in two years [13].

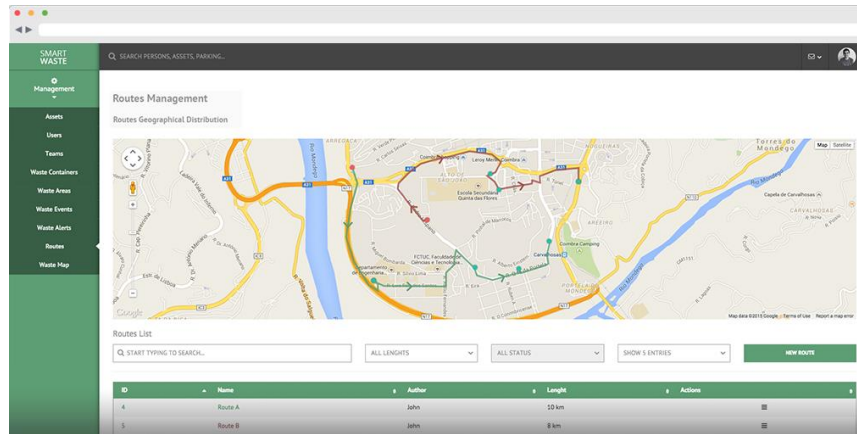


Figure 2.2: Smart waste management helps optimizing the rubbish pickup route.

- Padova, Italy.** A proof-of-concept IoT application was deployed in collaboration between the University of Padova and the city of Padova to promote early adoption of urban IoT solutions in public administration for "Padova Smart City" [2]. The target application consists of a system for collecting environmental data and monitoring public street lighting using various wireless sensors mounted on street light poles. The implementation uses Temperature, humidity, light intensity, and benzene sensors to monitor weather conditions and air quality. Data was collected over a period of 7 days which observes regular pattern of measurements corresponding to day and night cycles, which can be used to detect anomalies due to traffic congestion and weather conditions, such as rainstorms, so that system adjustments can be made automatically.

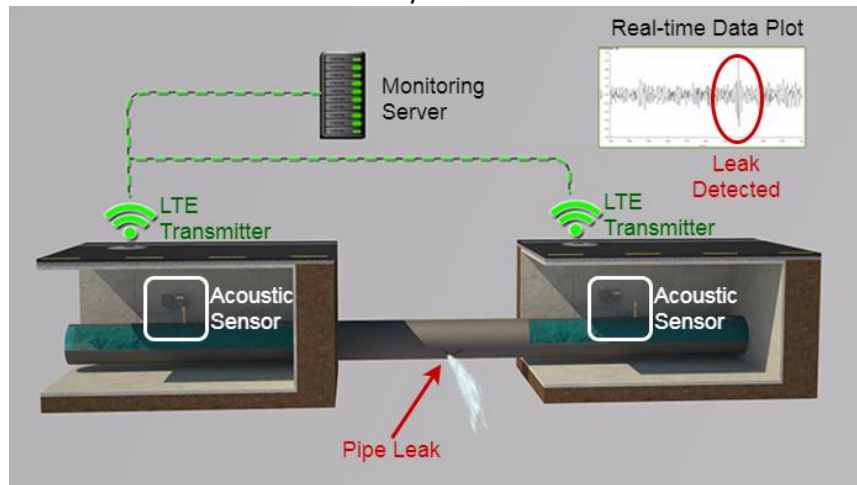


Figure 2.3: Underground acoustic sensors on water pipeline [14].

- Las Vegas, USA.** Located in the desert with limited waters supply, Las Vegas looks to optimize its water distribution and management system to be as efficient as possible. The Las Vegas Valley Water District (LVVWD) employs wireless leak detection devices that periodically listen for sounds or vibrations that may be caused by water seeping from the system. In total, 13 permanent acoustic sensors are installed monitoring 3 miles of the aging pipeline under Las Vegas Boulevard [14]. The technology enables preventive maintenance by detecting small leaks that may go unnoticed and reduce leakage loss. Though LVVWD has committed to reduce water consumption per capita to 199 liters in 2035, consumption has fallen to 205 liters per person/day at the end of 2014 thanks to its proactive leak control and maintenance programs [15].

### 2.1.3.2 Smart Living Projects

- New York City, USA.** An interactive platform called City24/7 is launched in collaboration with Cisco and the City of New York to deliver information from open government programs, local businesses, and events to citizens as requested [16]. Information is displayed on durable, easy-to-use Smart Screen kiosks that replace unused public assets such as payphone and posting boards typically located at key locations such as bus stops, train stations, and shopping malls (as seen in Figure 2.4). These kiosks incorporate touch, voice, and audio technology to deliver wide variety of local news, services, and advertisements to all types of users, including people with disabilities. The kiosks can also be equipped with sensors to support a citywide sensing network that can monitor and alert people environmental and weather conditions, as well as cameras for video surveillance applications. The initiative aims to deploy 250 Smart Screens throughout New York City with future plans to expand toward to Los Angeles, London, Boston, and many other cities in the United States and around the world [16].

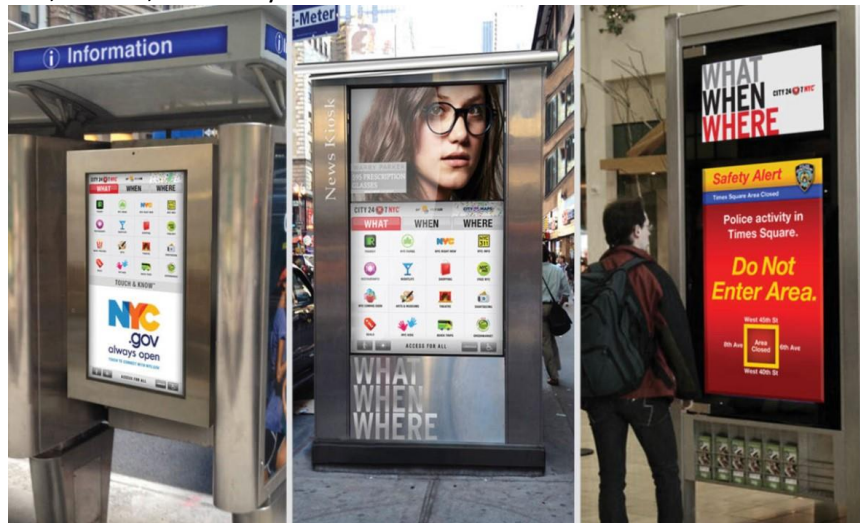


Figure 2.4: New York City 24/7 Smart Screen [16].

- Seoul, South Korea.** Seoul has operated the u-Seoul Safety Service since 2008 using advanced location-based services and CCTV technologies to notify authorities and family members of emergencies involving children, the disabled, or the elderly [17]. A dedicated smart device was developed to notify its holder when they leave a designated safe zone and provide an emergency call button that can notify guardians and emergency respondents when pressed. The device also includes a GPS tag to monitor its holder's location, whose data can be complemented by real-time CCTV networks to quickly locate missing children. Seoul is expected to have 50,000 users registered to the service with multiple efforts to provide necessary devices to low-income and vulnerable groups.
- Medellin, Colombia.** Medellin created the Integrated Emergency and Security System (SIES-M) in 2013 to bring together more than 10 different government agencies responsible for responding to emergencies [18]. The system enables different services such as police, medical, and fire departments to mount a coordinate response to citizen reports through a single security and emergency number, 123. Information from citizen reports is cross referenced with data from 823 video surveillance cameras distributed throughout the city. Each camera covers a 120-meter radius area, and is linked to fiber optics and wireless network to transmit high-definition video to integrated control center. The SIES-M has enabled optimization of resources for city security organizations and increased timeliness of services to citizens in emergency and security situations.

As summarized in Table 2.1, Smart City initiatives are gaining attention all over the world, with many cities already develop fully operational smart services while others begin pilot projects. In these initiatives, it is clear that IoT technology plays an integral role in many Smart City applications to collect comprehensive

data about the city and their residents. While most designs utilize dedicated sensors to collect specific data for their smart services, devices producing multi-purpose data like *cameras are becoming more common* and often seen in many applications across multiple domains, such as high-definition cameras being used in both smart traffic management and public safety applications. As such, it is *more efficient in terms of both deployment and maintenance* cost to reuse generic data from these devices for various Smart City applications. Moreover, it is also observed that *wireless communication* also gains its popularity as the glue, connecting the low-power, low-capacity IoT devices to the core network.

Motivated by the above observations and guided by the structural initiatives of Montréal City to become a world renowned leader among smart and digital cities<sup>2</sup>, this small-scale Smart City pilot deployment and demonstration project is conducted. The project consists of three interconnected sub-projects: Intelligent Traffic (lot 1), Urban Asset Management (lot 3) and Public Security (lot 4) are implemented in various locations of the Quartier de Spectacles in the City of Montréal. This report will provide a summary of all of the initial proposed technical contents as well as the detailed implementations and designs.

**Table 2.1:** IoT Applications in Smart Cities.

Category	City	Application	Instruments	Outcome/Impact
Smart Mobility	Barcelona [6]	Parking notification, public transport monitoring	wireless parking sensors, GPS sensors	encourages public transportation and improves customer experience
	Beijing [7]	Smart traffic management	HD cameras, traffic flow detectors, traffic signals	reduces congestion by 60% and doubled road capacity
	Santander [8], [9]	Smart parking management	parking sensor, wireless communications	experimental application to notify drivers of available parking to reduce commuting
Smart Sustainability	Amsterdam [10]-[13]	Smart lighting, smart public service	light sensors, wireless electricity meters, wireless smart bins	reduces energy consumption cost by 130 billion euro reduces the CO2 emission by 62% in 2 years
	Padova [2]	Smart lighting, environmental monitoring	wireless temperature, humidity, light, benzene sensors	proof-of-concept implementation to observe light intensity and weather conditions

<sup>2</sup> Montréal Smart City Strategies - <http://villeintelligente.montreal.ca/en/strategy>



	Las Vegas [15]	smart water leak detection	wireless acoustic sensors	enables preventive maintenance and reduces water loss to leakage
<b>Smart Living</b>	New York [16]	Smart information kiosks	Smart Screen kiosks with video, video capabilities and sensors	delivers information and advertisements on-demand to citizens environment monitoring and surveillance
	Seoul [17]	Smart safety	GPS monitor, CCTV cameras	enhances security for children and vulnerable citizens
	Medellin [18]	Smart public safety	HD cameras	enables coordinated and enhanced emergency response

## 2.2 IoT System Architecture and Enabling Technologies for Smart City

The IoT system requirements and challenges have driven a new architecture discipline. Typically, IoT follows service-oriented architecture (SoA) paradigm, which separates functionalities into generic, well-defined layers that interact with each other via interfaces and protocols [1], [3], [24]. Various architectural standards and frameworks have been proposed to address the challenge of designing massive-scale IoT networks in the past several years, e.g., *oneM2M* ([www.onem2m.org](http://www.onem2m.org)), *IoTWF* (IoT World Forum), Purdue Model, *IIoT* (by Industrial Internet Consortium), *IoT-A*, *ITU-T IoT* reference model. The foundational concept in all these architectures is supporting data, process, and the functions that endpoint devices perform. The abstraction of different technologies into common sets of functions and services simplifies the development and integration of new components and services to IoT. This allows various IoT applications to be supported by a common infrastructure.

Figure 2.5 illustrates the proposed Smart City IoT 4-layer system architecture, which is quite closed to the well-known IoTWF and ITU-T IoT reference models<sup>3</sup>. We believe that this architecture provides a good abstraction for understanding the Smart City system as a whole as well as enough details for implementation.

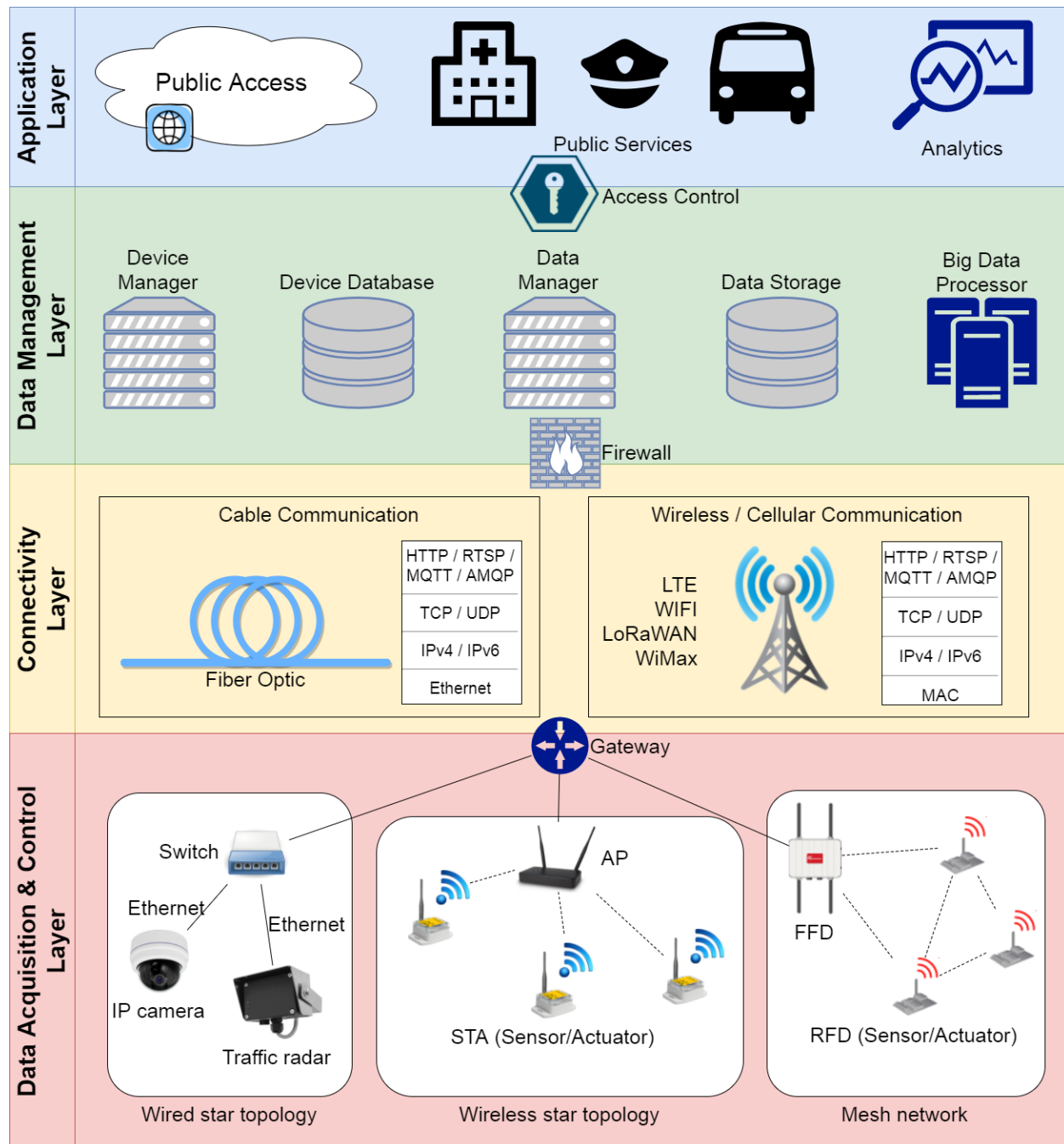
The *data acquisition & control* layer is responsible for gathering/sending data from/to a variety of sources throughout Smart City with its massive deployment of IoT devices.

The *connectivity* layer includes the backbone network that carries data and device commands between the *data management* layer and the *data acquisition & control* layer.

The *data management* layer provides services and resources to manage and manipulate IoT devices and aggregate data from them.

The *application* layer provides an interface for users to request data or services from IoT for applications such as data analytics, traffic management, smart transportation, smart grids, etc. As the application layer implementation is unique for every context, it is too extensive to cover the details of this layer in the common infrastructure. The rest of the section will focus on the details and enabling technologies of the *data acquisition* layer, the *connectivity* layer, and the *data management* layer.

<sup>3</sup> See Appendix A for more detailed discussions.



**Figure 2.5:** Smart City IoT multi-layered system architecture.

### 2.2.1 Data Acquisition & Control Layer

The data acquisition & control layer is where all data gathering in Smart City takes place. Enormous amount of assorted information is collected by various types of IoT sensors. This layer also composes of actuators such as traffic lights, street panels to react according to some preconfigured configurations or explicit directions from the human in charge. In fact, the components of this layer vary the most in type, capability, and quantity according to the requirements of Smart City applications. RFID and sensor networks are considered the basic building blocks of this layer [26], mainly due to the low cost hardware.

Thus most implementations employ these technologies to provide identification and sensing capabilities for IoT devices.

RFID is a non-contact communication technology used to identify and track objects over a short distance [27]. It is typically composed of RFID readers and RFID tags. Each tag is attached to an object and has a unique identification number, which can be queried by a reader that generates an appropriate query signal. Usually, RFID tags are passive, i.e., they consist of a microchip attached to an antenna with no power supplies and harvest energy required for transmission from the query signal. As a result, they can only be read accurately within a few meters [1]. However, there are also semi-passive and active RFID tags equipped with batteries to power the microchip and the transmission signal to achieve longer range at a higher production cost [1].

Sensor network also plays an important role in IoT data acquisition by monitoring and tracking status of objects and environments. Therefore, it can enhance awareness of the surroundings, acting as a bridge between physical and digital world [3], [28]. Sensor network usually consists of many small-sized sensing nodes that communicate with each other and transfer data via wireless communications, allowing for versatile deployment in various areas. Most commercial sensor networks are multi-hop WSNs based on IEEE 802.15.4 standard [1], which defines the physical and MAC layers for low-power, low bit rate communication in wireless personal area network (WPAN). IEEE 802.15.4 can support bands of 868/915 MHz and 2.4 GHz, with data rate up to 20 kbps, 40 kbps, and 250 kbps, respectively [29]. The most popular example is Zigbee, a wireless network protocol based on IEEE 802.15.4 that offers low energy consumption, low cost, and low data rate [30], [31].

Recently, new applications like smart video surveillance and traffic control demand sensing capability and data rate much higher than supported by RFID and 802.15.4-based WSNs. These applications require real-time high-definition (HD) images from Internet protocol (IP) cameras, which requires very high-speed data transmission. To accommodate the new requirements, IEEE 802.11 (Wi-Fi) has been incorporated into WSN to provide the necessary data rate [32]. Wi-Fi offers higher throughput (up to 1300 Mbps in IEEE 802.11ac) and longer coverage range than 802.15.4-based WPANs, at the cost of scalability due to the limited number of simultaneous devices that can be supported [31].

### **2.2.2 Connectivity Layer**

The connectivity layer transports data collected from IoT devices to data center and delivers commands from remote control operator to the IoT devices. It is the backbone network infrastructure that bridges communication between the management layer and the data acquisition layer. The communication is carried out by either wired network (e.g., fiber optic) or wireless/cellular network (e.g., LTE, LoRaWAN), depending on the availability of such technologies and requirements of Smart City applications.

Wired communications generally serves as the backbone infrastructure that connect stationary wired IoT devices (e.g., IP cameras) and short to medium-ranged WSNs (e.g., Wi-Fi and Zigbee sensor networks) to the Internet. Fiber-optic network is expected to be the foundation of Smart City wired infrastructure as it offers high bit-rate capacity in comparison to traditional copper cable technology [33]. Its capability allows fiber-optic network to support many high-speed multimedia Smart City applications, such as real-time video surveillance. Communication is typically aggregated at specific fiber drop locations, where a gateway transfers data to fiber-optic network. These locations are determined by the service provider and are often fixed, limiting the coverage of fiber-optic network.

On the other hand, various wireless and cellular technologies offers wide coverage area that can support long range communication between devices. LTE has been making progress to support low-powered M2M communications, introducing various features such as power saving mode and reduced device capability to extend battery life [34]. LoRaWAN is also developed as a stand-alone network dedicated for device communication, aiming to standardize low-power, wide area communications for IoT [34]. However, these wireless technologies does not support the data rate and capacity required for a large number of high throughput devices such as IP cameras in Smart City deployment.



### 2.2.3 Data Management Layer

The Data Management Layer oversees the operation of Smart-City devices and data. This is where the collected data from the entire Smart-City is stored and processed. It also provides interface for remotely monitor, configure, and control IoT devices in the data acquisition & control layer. Usually, the logical components of the management layer are separated into two subsystems: device management and data management. Device management system includes a device manager and a device database for identification, tracking, and configuration of devices. Data management group includes data manager and data storage for data aggregation, vitalization, and archive. Each system may contain one or several middleware subcomponents for interoperability due to the heterogeneous nature of IoT devices and data. Additionally, the data management layer maintains access control over devices and data to prevent misuse of information and services provided by Smart City IoT infrastructure.

Generally, these management components are hosted on a private platform consisting of back-end servers located in a control center. In this approach, back-end servers provide the processing resources and storage capacity to run various database management systems, storing the huge amount of data and device information produced by IoT sensing devices [2]. The load on these systems can become incredibly huge so many back-end servers may have to be setup in parallel and properly managed, thus careful planning of resource provision and maintenance is required to ensure that the management system can support all operations by IoT devices and users. The cost of deploying, operating, and maintaining such a control center can become very expensive, which significantly hinders the adoption of IoT paradigm.

Meanwhile, Cloud Computing provides a convenient, on-demand and scalable management platform with access to a pool of virtual computing resources [35]. Cloud Computing and IoT integration, also referred to as Cloud of Things [36] or CloudIoT [37], and abstracts away the hardware back-end server details that would have to be considered in the private control center approach. Instead, cloud resources are offered in various forms as a service: IaaS (Infrastructure as a Service) offers a pool virtual machines (VM) for computing and storage, PaaS (Platform as a service) allows an abstract interface to deploy IoT applications, and SaaS (Software as a Service) presents a collection of predefined applications to users [35]. In recent years, multiple cloud platforms, such as Amazon Web Services (AWS) [38], IBM Bluemix [39], and Microsoft Azure [40], have emerged to tackle the challenges of Cloud Computing and IoT integration.

In addition, the data management layer has to interface with IoT devices and process data generated by those devices. For a vertical IoT application, i.e., all data acquisition devices, communication networks, and management services are dedicated to a single application, interface and management system of devices and data are fairly straightforward. However, the integration of multiple technologies and applications across many domains into a common infrastructure is difficult due to the heterogeneity of IoT devices, technologies, and services. A middleware layer can act as a bond joining heterogeneous domains of IoT technologies and managing applications [41]. Middleware is a software component that provides an abstraction of different IoT technologies, allowing management services to control and configure devices through standard interfaces. It offers the benefits of supporting heterogeneous networks, devices, and applications, enabling interoperability between IoT components [3], [42].

Many implementations of middleware have been developed for IoT scenarios, divided into 3 categories: service-based middleware, cloud-based middleware, and actor-based middleware [43]. Service-based middleware, such as Hydra [44] and Global Sensor Network (GSN) [45], adopts SoA architecture and allows developers to add or deploy IoT device functions as services. Cloud-based middleware solutions, like Google Fit [46] and Xively [47], are deployed on cloud computing platforms to enable users to connect to IoT devices and collect data from any location. Actor-based middleware emphasizes on open, plug-and-play IoT architecture with reusable components such as gateways to ease integration of devices to IoT infrastructure. Calvin [48] and Node-RED [49] are examples of this IoT middleware platform category.

### 2.2.4 Application Layer

Application Layer is where the data is processed and analysed in order to extract useful information about the vast amount of collected data. The applications that can be provided from this layer are what make up the attractiveness of IoT and Smart City, ranging from public services such as arrival time of the next bus to a medical report of an individual. Depending on the analytic functions applied on the data, the value added information could be as raw as counting the number of vehicles passing an intersection or as complicated as the recommendations for planning the city infrastructure for the next 10 years. Moreover, the readings of the collected data could raise an alarm or warning on some critical events and depending on the algorithms they could trigger an actuator to react quickly to the situation. Furthermore, each iteration of processing produces more valuable information that may be applicable to other application domains. The possibilities for IoT applications in Smart City, are virtually endless, though they generally aim to achieve one of the following 3 purposes: presentation, optimization, and prediction.

- **Presentation** enables human operators to observe and comprehend a situation through data collected from IoT devices. These applications generally compile a huge amount of data and display it in a coherent, meaningful manner, such as graphs and diagrams. They can visualize incoming data in real-time or combine it with historical data to identify trending patterns that may indicate anomalies in the system. For example, a video surveillance application identifies suspicious objects in the video footage from a remote IP camera, then displays the processed video with highlighted frames and raises an alarm to notify operators if necessary.
- **Optimization** uses advanced analytics to automatically optimize a complex system. In addition to monitoring incoming data from IoT devices, these applications also have active control over IoT components, such as actuators, to adjust the system dynamically in response to operating conditions observed in the data. An example of this type of application is a smart lighting application that adaptively regulates the amount of power to public street lights according to nearby activities and weather conditions.
- **Prediction** applies statistical analysis to predict the likelihood of events in the future. These applications incorporate historical data from many sources to determine the prospect of events such as traffic jams, floods, and power outages, based on relevant traffic or weather conditions. The prediction results allow the authorities to quickly mobilize resources and develop actionable plans in response to the event, thus minimizing its impact to the smart city and its citizens.

## 2.3 Open issues of Smart City based on IoT

Although IoT and Smart City are gaining popularity in both research community and city administration, there are still questions that need to be addressed to fully realize their potentials. These issues include resource provisioning, scalability, virtualization, security and privacy and are elaborated further in the following.

### 2.3.1 Scalability

Scalability refers to the possibility of expanding the Smart City infrastructure to accommodate more services as well as new type of hardware in the future. In other words, what are the device and data requirements that a full-scale Smart City IoT system should expect to support? The scale of IoT system is critical to the design and planning of Smart City infrastructure. For example, the large number of IoT devices and their massive data throughput requirement can have significant influence on the planning and installation of Smart City network infrastructure. While most recent literature mentioned the scalability of IoT and Smart City as an important issue to be addressed, few have provided an estimation of the quantity of devices and data to be supported in a Smart City IoT scenario.

### 2.3.2 Virtualization

In opposed to the Wireless Sensor Network world where the deployed network has to be cost effective and sensors are tailored to fit specific tasks for a particular application, the smart infrastructure of Smart City has to be designed to be used as a public platform or to be shared among various applications from multiple parties such as police, fire fighter departments, hospital, governments as well as for private companies. This design is not only to maximize the benefits of this infrastructure; deploying a separate infrastructure for additional applications is cost ineffective, redundant and to some extents would limit the operations of the existing network due to interference and bandwidth contention. At the same time, it is also crucial that resource allocation to different applications are isolated so that one application is not taking resources from another application. As such, virtualization becomes an important issue in the context of a Smart City, for example, how can different users/applications control the same device at the same time? Thus, designing an effective virtualization scheme for IoT and Smart City to provide abstraction of the physical resources so that different applications can utilize at the same time is a significant challenge to overcome

### 2.3.3 Security and Privacy

Security is among the top priority issues to be addressed when implementing IoT system at a large scale like Smart City. IoT system is subjected to many security vulnerabilities such as vandalism, denial-of-service (DoS), and data leakage and tampering [50], [51]. As IoT devices are often unattended, they are vulnerable to physical attacks and vandalism [1]. Attackers can also use DoS attacks to disrupt communication channels or devices [37], jeopardizing the operations of Smart City applications. Components and services that require real-time data are particularly vulnerable to such attacks. Data from sensors may be eavesdropped or tampered with, potentially revealing confidential information to malicious hackers or exposing the system to erroneous data. The effects of these attacks can be very devastating from breaking into a house, altering medical records to endangering public safety of a city by manipulating the actuators in Smart City. These dangers make security protection especially critical when Smart City collects personal data on its citizens. A more extensive discussion of Security will be provided in Chapter 5.

## 2.4 Conclusion

In this chapter, the current trends in IoT technologies towards Smart City is described, followed by a brief review of sample Smart City applications deployed around the world. Then, the overall Smart City IoT architecture is described through a four-layer approach. In each layer, the functionalities, components, enabling technologies were presented and discussed. Finally, the key issues that are critical to the deployment and development of a Smart City were highlighted.

## Chapter 3

# Montréal Smart City Pilot System (MSCPS)

### 3.1 MSCPS deployment summary (Lot 1, 3, 4)

Following Montreal's Strategic Plan for the Smart and Digital City, in this pilot project, various types of sensors and commercial or free software are tested to determine the best standards to be followed, to determine the best integration architecture and sensor management in terms of transport and data processing, and to validate the business gains of the smart transportation concept. In particular, the Montreal Smart City Pilot System (MSCPS) follows a similar direction as the other Smart Cities mentioned in Chapter 2 with an emphasis on Intelligent Traffic (Lot 1), Urban Asset Management (Lot 3) and Public Security (Lot 4). The pilot deployment system in this project spans across all layers of the general architecture shown in Figure 2.5; however, due to the time limitation, most of the efforts focused on the Data Acquisition Layer and Management Layer, investigations were also done on the Communication Layer and Application Layer to illustrate the feasibility of different approaches. In this section, the applications and tasks in each lot are listed along with the actual achievements (in the deployments), and corresponding comments/notes. They will be organized based on each layer as shown in Figure 2.5 for the ease of understanding.

#### 3.1.1 Lot 1 – Intelligent traffic

For Lot 1, the focus is on Intelligent Traffic applications; the proposed tasks and actual achievements are summarized in Table 3.1. In order to realize this application, a wireless infrastructure was built first to provide the necessary coverage to installation locations that are not at the fiber drop. For efficient performance, quasi Time Division Multiple Access (TDMA) was used over the existing WiFi 802.11n and 802.11ac technologies. Then, a system of 6 Full/UltraHD cameras with microphones and 9 traffic radar sensors (+1 from the initial proposal) was deployed. For data communications, the data from different sensors are routed to a nearest fiber drop at QdS site and then to VdM and BCRL McGill University operational sites. Initially, to secure the data, Virtual Private Network (VPN) was used between QdS, VdM and McGill sites. In Aug 2017, QdS and VdM VPN link was replaced by private fiber connection, the remaining VPN link between McGill and QdS site is an example of deployment solutions where private fiber/wired network does not reach. For future planning, a planning estimation for the Montreal area was also studied.

Regarding the application, a scaled-down data center in BCRL at McGill University was deployed and operates from Feb, 2017 until Sep. 2017, and then it was replicated and operated at VdM site. This facilities enable data storage for testing of various applications which cannot be done without it. Various private and cloud applications were deployed and tested for video management and storage. Several commercial computer vision applications as well as in-house build applications based on free/open source library were also tested for video analytic purpose.

**Table 3.1:** Summary of proposed tasks and achievements in Lot 1- Intelligent traffic.

Task:	Achievement & Note:
<b>Data Acquisition (DA) Layer:</b>	
Topology selection	Point-to-point and point-to-multipoint/Star topologies were selected and implemented with cameras and traffic surveillance radar sensors.
Test of PHY/MAC protocols	Pilot wireless infrastructure was deployed with TDMA media access protocol to avoid collisions and increase airtime efficiency.
Wireless infrastructure deployment using WiFi (802.11b/a/g/n/ac)	Pilot WiFi infrastructure was deployed with 802.11n and 802.11ac technologies.
Deploy 6 Full/UltraHD IP cameras	6 Full/UltraHD IP cameras and 3 microphones were deployed.
Deploy 8 traffic radar sensors	9 traffic radar sensors were deployed <u>Note:</u> +1 radar sensor
Data connection to the big data center of the City of Montreal through the backbone fiber network.	Fully deployed
<b>Connectivity/ Transport (CT) Layer:</b>	
Communications between DA & CT Layers with support for Internet and network protocols such as IPv4, IPv6, TCP/UDP.	Fully deployed.
Investigate the current and emerging standards and configurations and testing the best security service required for transport layer.	A survey of current security features and configurations on the deployed devices is provided. <u>Note:</u> Testing of security services will be done in the next steps
Security of data transport.	Secure data transport is realized through the use of Private network and VPN link over public network. Studied planning estimation for Montreal downtown and the whole Montreal area. <u>NOTE:</u> to further improve security of data transport, only particular protocols, and ports which are needed, should be enabled on the firewalls of the private network.
<b>Data Management Layer:</b>	Implement a prototype platform for data management and device control
<b>Application Layer:</b>	
Implement a scaled down data center in BCRL at McGill. Final main Application Layer will be setup and managed by the City of Montreal	The scaled down data center in BCRL at McGill was setup and run and run from Feb. to Sep. 2017. <u>Note:</u> The final main Application Layer at the City of Montreal was replicated in Sep. 2017.
Select, procure and test a number of reliable, commercial computer vision analytics applications.	Possible selected applications for testing include “Video Intelligence” software from AdMobilize <sup>4</sup> using people metrics and crowd metrics (people counting, pedestrian Traffic Direction).

<sup>4</sup> AdMobilize - <https://www.admobilize.com/>

	<b>NOTE:</b> Used free AdMobilze examples, instead of our own video clips, as the price seems excessive (\$5000 USD for 5 short video clips analysis).
Test candidate software packages for computer vision applications.	<ul style="list-style-type: none"> <li>Developed a test application for street parking space monitoring and vehicle/pedestrian counting based on OpenCV open library</li> <li>Deployed and test various private and cloud applications for video management, and storage.</li> </ul>

### 3.1.2 Lot 3 – Urban Asset Management

For Lot 3, the focus is on Urban Asset Management applications; the proposed tasks and the actual achievements are illustrated in Table 3.2. In this Lot, Zigbee and Bluetooth Low Energy sensors were integrated to a VdM salt-and-abrasive spreading vehicle to monitor the trajectory, temperature and the remaining salt/abrasive level in the container on the vehicle. In addition, one smart inspection vehicle was also deployed for monitoring city trees and road conditions. Data communication channels for these vehicles are realized through LTE and Wifi technologies. Regarding the security, a survey of current and emerging standards and configurations is provided.

Regarding the applications, the scaled-down data center at BCRL, McGill University was used for data storage. In addition, the possibility of communication to the cloud as well as storage offload to the cloud was also investigated.

**Table 3.2:** Summary of proposed tasks and achievements in Lot 3- Urban Asset Management.

Task:	Achievement & Note:
<b>Data Acquisition (DA) Layer:</b>	
Topology selection	Point-to-point and point-to-multipoint topology were selected for the integration of Zigbee and Bluetooth Low Energy sensors.
Deploy fire-hydrant, garbage-container-fill-level and mobile-salt/gravel-container-fill-level sensors	<p>One salt truck with GPS, temperature and salt level sensors were deployed.</p> <p>In addition, one smart inspection vehicle for tree and road condition monitoring applications was deployed</p> <p><b>NOTE:</b> The applications for fire-hydrant and garbage-container-fill-level were removed as suggested by VdM</p>
Deploy urban assets active sensors using IEEE 802.15.4/Zigbee standards, Bluetooth and 6LoWPAN	<p>IEEE 802.15.4/Zigbee and Bluetooth enabled sensors were deployed.</p> <p><b>NOTE:</b> 6LoWPAN-compatible and deployable devices were not commercial available at the time of procurement.</p>
Implement an easy to use but secure mobile phone/device application for updating status of urban assets.	<p>Not deployed</p> <p><b>NOTE:</b> To be decided for deployment if time permits. Possibly for next steps deployment.</p>
<b>Connectivity / Transport (CT) Layer:</b>	
Communications from Data Acquisition layer to Communication/Transport Layer with support for Internet and network protocols such as IPv4, IPv6, TCP/UDP.	Fully deployed through Wifi and LTE technologies.
Investigate the current and emerging standards and configurations and	A survey of current and emerging standards and configurations is provided.

testing the best security service required for transport layer.	<u>NOTE:</u> More extensive testing of security services will be done in the next steps
<b>Data Management Layer:</b>	Implement a prototype platform for data management and device control
<b>Application Layer:</b>	
Implement a scaled down data center in BCRL at McGill. Final main Application Layer will be setup and managed by the City of Montreal	The scaled down data center in BCRL at McGill was setup and run from Feb. to Sep. 2017. Investigate the possibility of integrating communications and storage to the cloud. <u>NOTE:</u> The final main Application Layer at the City of Montreal replicated in Sep. 2017.

### 3.1.3 Lot 4 – Public Security

For Lot 4, the focus on Public Security applications; the tasks which were proposed and the actual achievements are illustrated in Table 3.3. In this Lot, most of the infrastructure for sensing and data collection was used in parallel with the applications in Lot 1. For the testing of applications, beside the commercial based applications, a test application for crowd counting was developed at BCRL, McGill, based on supported Matlab functions.

**Table 3.3:** Summary of proposed tasks and achievements in Lot 4 – Public Security.

Task:	Achievement & Note:
<b>Data Acquisition (DA) Layer:</b>	
Deploy 6 Full/UltraHD IP cameras	6 Full/UltraHD IP cameras and 3 microphones were deployed. <u>NOTE:</u> Utilize the same cameras as in Lot 1.
Wireless infrastructure deployment using WiFi (802.11b/a/g/n/ac)	Pilot WiFi infrastructure was deployed with 802.11n and 802.11ac technologies. <u>NOTE:</u> Utilize the same infrastructure as in Lot 1.
Deploy 2 pedestrian InfraRed sensors	Not deployed <u>NOTE:</u> Hardware cost to benefit ration not attractive as agreed with VdM
<b>Connectivity / Transport (CT) Layer:</b>	
Data connection to the big data center of the City of Montreal through the backbone fiber network.	Fully deployed
Security of data transport.	Secure data transport is realized through the use of Private network and VPN link over public network.
<b>Data Management Layer:</b>	Implement a prototype platform for data management and device control
<b>Application Layer:</b>	
Implement a scaled down data center in BCRL at McGill. Final main Application Layer will be setup and managed by the City of Montreal	The scaled down data center in BCRL at McGill was setup and run from Feb. to Sep. 2017. <u>NOTE:</u> The final main Application Layer at the City of Montreal is replicated in Sep. 2017.
Select, procure and test a number of reliable, commercial computer vision analytics applications.	Possible selected applications for testing include iQ-140 (detecting non-motion objects) and iQ-Smoke (detecting fire and smoke) from iOmniscient <sup>5</sup> .

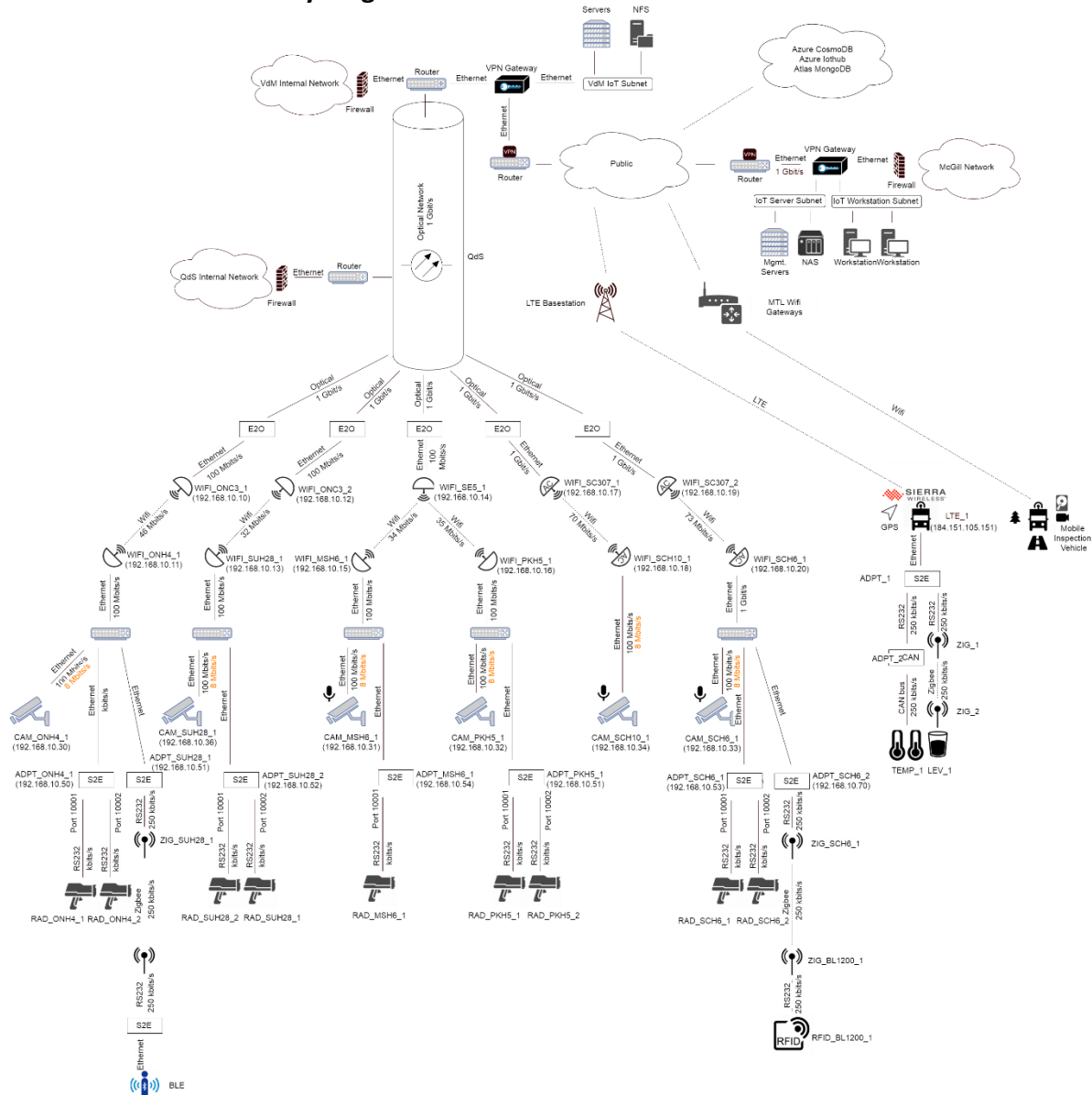
<sup>5</sup> iOmniscient - <http://iomniscient.com/>



	<b>NOTE:</b> Sample video clips for QdS deployed cameras analyzed by iOmniscient (courtesy service).
Test candidate free software for computer vision applications.	Developed a test application for crowd counting based on Matlab functions.

## 3.2 VdM pilot deployment network structure

### 3.2.1 Network connectivity diagram



**Figure 3.1:** The complete network connectivity diagram of the deployed devices.

The main IoT network deployment is concentrated in Quartier des Spectacles in Montreal. In the current setup, there are more than 50 devices deployed and interconnected using various wired and wireless communications. These devices include WiFi and Zigbee radios, switches, cameras, RFID readers, and traffic radars. Figure 3.1 details the complete network connectivity diagram of all the devices and their communications methods. Due to the fixed locations of the fiber drops, several wireless communication technologies were used depending on the use cases. WiFi was used to extend the network coverage and



connect to high volume traffic such as cameras. For those devices which does not have IP connection (radars, RFID), adapters are used to bridge the communication mediums such as RS232 to Ethernet bridges, and Zigbee adapters, then to the WiFi network. In addition, data from sensors which are attached to mobile vehicles are collected in real-time via LTE networks (if data volume is low) or stored locally in an onboard storage (if data volume is high).

To enable data collection and device control to the IoT network, a NAT gateway is used. The public IP representation of the whole IoT network is **132.206.68.25**. A management server is in charge of this task, translating the destination addresses of incoming messages to the appropriate device's address and forward the packets to the node. This server is also the source of NPT time synchronization across all of the devices. In order to secure the connections to and from the IoT network as well as the confidentiality of the transmitted data, a private VdM fiber network is used between QdS and VdM sites, for the connection to BCRL McGill site over the public network, Virtual Private Network (VPN) is used. For data that will be pushed to the Azure cloud for testing, they will be relayed and translated by servers at BCRL. The network diagram in Figure 3.1 follows closely the architecture in Chapter 2 with the different kinds of sensors and WiFi connection at the Data Acquisition & Control Layer, the fiber, VPN and LTE acts as the Connectivity Layer, the upper two layers are realized in software in servers so they are not physically separated.

In Figure 3.1, data speeds annotated in black represent the maximum connection speed between devices and those in orange shows the average transmitted data rate in current deployment. Naming convention follows the format of a shorthand device-type naming followed by the location code and an index number when multiple devices were installed at the same location. Device codes in current deployment are shown in Table 3.4. Location code mappings to real deployed location are also shown in Table 3.5. Devices without static locations, such as sensors installed on mobile vehicles will have their location labels omitted.

**Table 3.4:** Device naming convention codes

Name	Device
ADPT	Adapter
CAM	Camera
LEV	Level Sensor
LTE	LTE Gateway
RAD	Traffic radar
RFID	RFID reader (includes BLE reader)
TEMP	Temperature sensor
WIFI	Wi-Fi radio
ZIG	ZigBee radio

**Table 3.5:** Location code and real deployed location

Name	Location
BL1200	1200 Rue de Bleury
MSH6	Intersection of Rue Jeanne-Mance and Boulevard de Maisonneuve O
ONC3	32 Rue Ontario O
ONC4	Intersection of Rue Clark and Rue Ontario O
PKH5	Intersection of Rue Jeanne-Mance and Avenue du President-Kennedy
SC307	307 Rue Sainte-Catherine O
SCH10	Intersection of Rue Jeanne-Mance and Rue Sainte-Catherine O
SCH6	Intersection of Rue de Bleury and Rue Sainte-Catherine O
SE5	On Avenue du President-Kennedy
SUH28	Intersection Saint-Urbain and Boulevard de Maisonneuve O

### 3.2.1 Physical deployment network structure

#### Installation locations

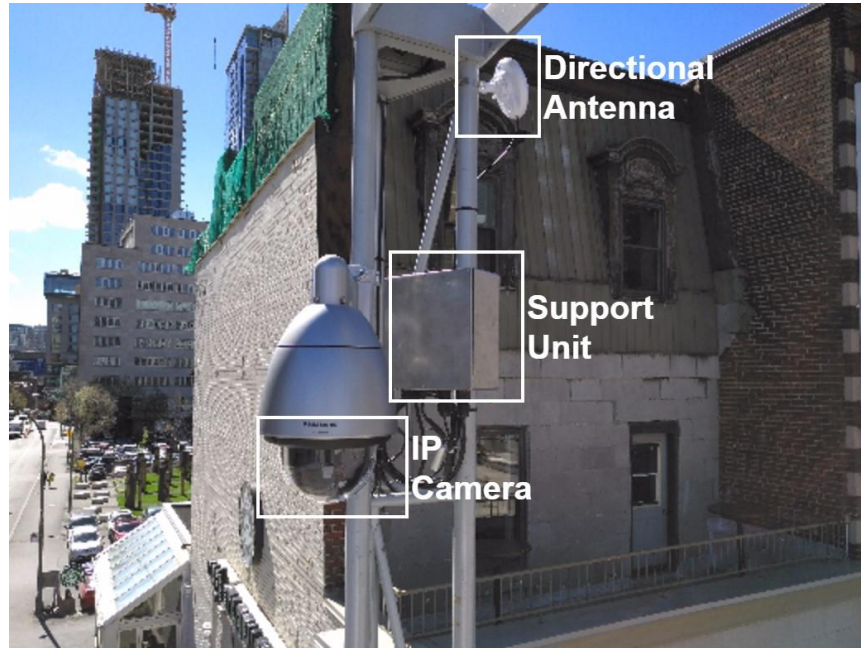


Figure 3.2: Camera and wireless antenna deployed on a light pole.

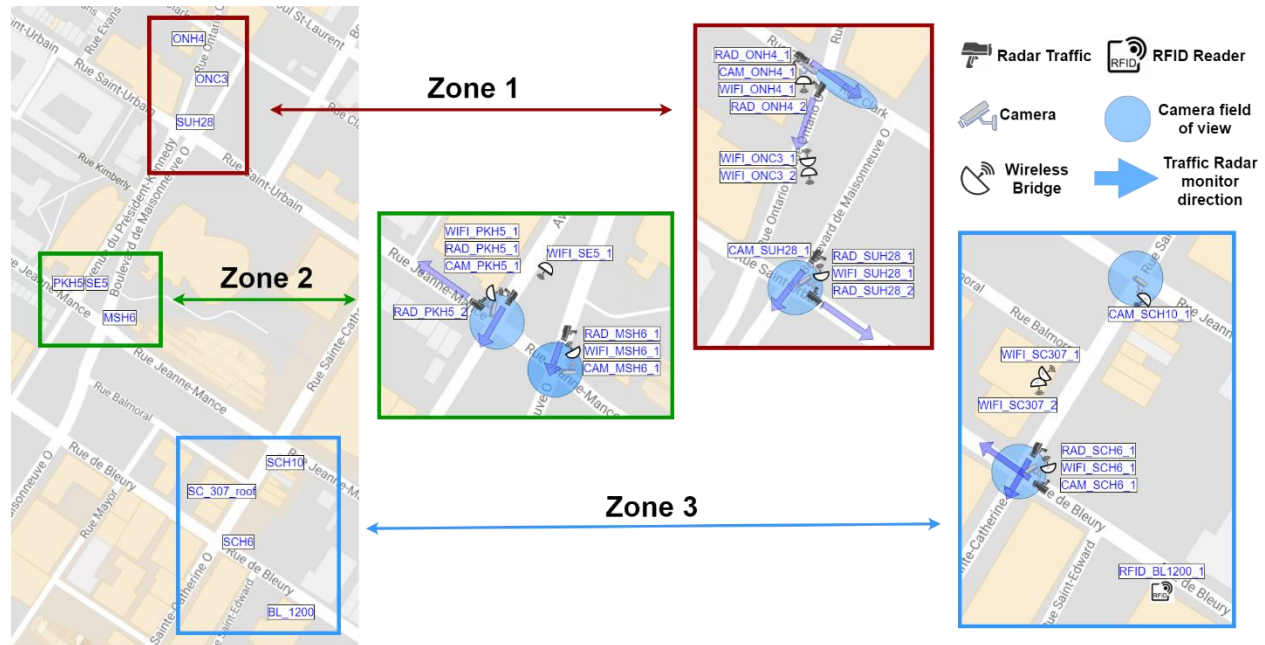


Figure 3.3: Physical map of the deployment structure

Installation location of IoT devices can address many issues of the data acquisition layer such as routing to gateway and device physical security. Target surveillance locations for sensing and surveillance are selected based on two main factors: *distance to the closest fiber drop*, and *daily traffic density*. The *distance to the closest fiber drop* from the installation location is an important factor that influences the connection between the IoT sensors and gateway. However, these fiber drop locations do not always offer an ideal, unobstructed view to monitor nearby traffic. A more practical approach is to establish a wireless

bridge between the sensors and the fiber drop locations to extend coverage farther to locations where a more desirable view can be obtained, offering more flexibilities in location selection so that more interesting data can be gathered. It is remarked that the distance to the fiber drop should be minimized to reduce signal degradation and interference with public wireless network and to maintain a stable, high data rate to the fiber drop. *Daily traffic density* of a target area is crucial to ensure that the data collected is sufficient in volume and diversity for meaningful analytics. It is especially beneficial to install these devices in locations where they can be used for multiple applications such as near busy intersections and public spaces.

Once a target location is selected, where to install the surveillance camera for that area needs to be considered. The position of camera installation must provide a good view of the target area, prevent vandalism, and reduce obstructions to the wireless bridge connection. The camera view will depend on the application and will be discussed in Chapter 6. In addition, the camera and the directional antenna require a constant power supply for their continuous operation. As a result, a light pole (with existing 110V AC source) provides an excellent spot to deploy the sensors and cameras. Figure 3.2 illustrates the setup of a camera, an antenna, and their support unit on a light pole. At the top of a light pole, the camera can obtain good overview of the surrounding area and avoid obstacles to the wireless connection. The power source of the light pole can also be used to supply camera and antenna operation. Furthermore, the light pole height also holds the devices high above the ground, which makes it difficult to vandalize the devices.

The physical device deployment locations are concentrated in the area of Cartier des Spectacles in three main zones. To save installation, maintenance costs, and make best use of the wireless bridge connections, in each zone, IoT sensors are normally installed together in groups of many sensors. Figure 3.3 highlights the sensing devices and radios locations. Please note that the various location codes seen in Figure 3.3 match location codes underneath devices in Figure 3.1. Each zone contains several cameras and traffic radars, Zone 3 also contains an RFID reader. In Figure 3.3 camera viewing fields are shown in blue and traffic radar setup directions are illustrated by purple arrows. Not pictured, but also present in these deployment zones are Ethernet switches, and Serial-to-Ethernet adapters to attach cameras and radars to the radios.

### 3.3 Device capabilities

This section summarizes the device models and capabilities used within the MSCPS. More detail about the implementation, device selection and deployment procedures and technical specifications of each of the devices can be found in *Appendix B, C and D*, attached with this report. In the project, whenever the budget allow, devices with highest set of features and state-of-the-art commercial available technologies are purchased. The devices are divided into three categories: *communication devices* such as Wi-Fi radios, *sensing devices* which include cameras and traffic radars and mobile sensors which are installed on trucks. Communication deployment consists of 11 Ubiquiti Wi-Fi radios, 6 Serial-to-Ethernet adapters from three different vendors, and 6 Digi ZigBee radios. Sensors include 6 cameras from three different vendors, 9 Geolux radars, an FEIG RFID reader, and a BLE beacon reader on the stationary network. Mobile sensors includes several sensors and communication devices which will be described in the next section. All devices were designed for outdoor applications with some devices were IP66, IP67 certified.

#### 3.3.2 Communication devices




Communication devices within the network are divided into two main categories: wireless and wired. Also discussed in this section are bridging methods such as virtual private network tunneling and network address translation.

##### 3.3.2.1 Wireless communication devices

Wireless technology is used to deploy devices in areas where it is hard to route Ethernet or Fiber Optic cables. Two types of wireless devices are used within the deployment: WiFi and Zibbee.

### 3.3.2.1.1 WiFi radios

**Table 3.6:** Ubiquiti radios used in the deployment project.

			
	NanoBeam NBE-M5-16	NanoBeam NBE-5AC-16	Rocket M5
Frequency	5 GHz, 802.11n	5 GHz, 802.11ac	5 GHz, 802.11n
Throughput	150+ Mbits/s	450+ Mbits/s	150+ Mbits/s
Range	10+ km	10+ km	
LAN speed	100 Mbits/s	1 Gbit/s	100 Mbits/s
Deployed Locations	ONC3, ONH4, SUH28, MSH6, PKH5	SC307, SCH10, SCH6	SE5

Within the deployment project, two types of WiFi radios were used to determine the best solution for connectivity in terms of performance and cost: IEEE 802.11ac and IEEE 802.11n. To create a communication Wi-Fi bridge, two radios are required. One acts as an access point (AP) and connects to the main wired network and the other behaves as a station (ST) and is connected to the appropriate sensor devices. The devices are thereby connected to the main network via the wireless communication channel. The Wi-Fi radio deployment in testing is composed of three models of Ubiquiti radios: NanoBeam NBE-5AC-16, NanoBeam NBE-M5-16 and Rocket M5 with the summary details as shown in Table 3.1. These devices operate at a 5 GHz frequency over the IEEE 802.11ac network protocol. The LAN connections of the 801.11n radios is limited at 100Mbps while that of the 802.11ac is 1Gbps. Configuration of the WiFi radios can be accomplished through SSH, via web interface or using Ubiquiti AirControl central management server.

### 3.3.2.1.2 Zigbee radios

In deployment, ZigBee is used to maintain a serial connection over wireless. The product used in the test-system is the Digi Xbee 232 Adapter S1 Pro and the Digi Xbee-Pro 900HP. These devices have a DB-9M connector to attach to serial devices and contains the Digi XBee-Pro radio module which has an RF data rate of 250 kbits/s. The device can be configured using the frbee Digi XCTU software. The summary details of the Zigbee radio is shown in Table 3.7.

**Table 3.7:** Zigbee radio used in the deployment project.


		
	Digi Xbee 232 Adapter S1 Pro	Digi Xbee-Pro 900HP RF Modem
RF Data Rate	250 kbits/s	250 kbits/s
Indoor/Urban Range	300 feet	1000 feet

Outdoor/RF Line-of-Sight Range	Up to 1 mile	Up to 28 mile
Deployed Locations	SUH28	SUH28

### 3.3.2.2 Wired communication devices

#### Serial

**Table 3.8:** Serial adapters used in the deployment project

			
	USR IOT USR N-520	Lantronix ED2100002-01	Perle IOLAN SDS2 W
Serial Capabilities	RS232/RS485/RS422	RS232/RS485/RS422	RS232/RS485/RS422
Protocols	HTTP, UDP broadcast	HTTP, HTTPS, FTP, Telnet, SNMP, SSLv2, SSHv2, SNMPv2	HTTP, HTTPS, Telnet, SSLv3, SSHv2, SNMPv3
Deployment Locations	PKH5, SUH28, SCH6,	MSH6	ONH4


Some of the deployed devices require a serial communication line. RS232 was used as a connection medium for ZigBee radios, traffic radars, and the RFID reader. To connect serial devices to the rest of IP designated network, Serial-to-Ethernet adapters were used to encapsulate the serial data to an IP packet. The list of the 3 deployed adapters and their capabilities is shown in Table 3.8.

### 3.3.2.3 Bridging Networks/Mediums

#### 3.3.2.2.1 VPN Tunnel

In the pilot setup, in order to protect the transported data between the deployment area and McGill BRCL and initially as well between the deployment area and VdM computing facilities, Virtual Private Network tunnels are used. A virtual private network tunnel is designed to securely bridge two private networks across the public network. The transported data through this VPN tunnel are protected against confidentiality, and integrity violations through cryptography mechanisms. In the pilot deployment, Cisco Meraki MX84 (as shown in Table 3.9) is used to create these tunnels. The current bandwidth of this tunnel is 320Mbps. (Note, initially used Meraki VPN device had a 60Mbps limit)

**Table 3.9:** Cisco Meraki MX84 device.

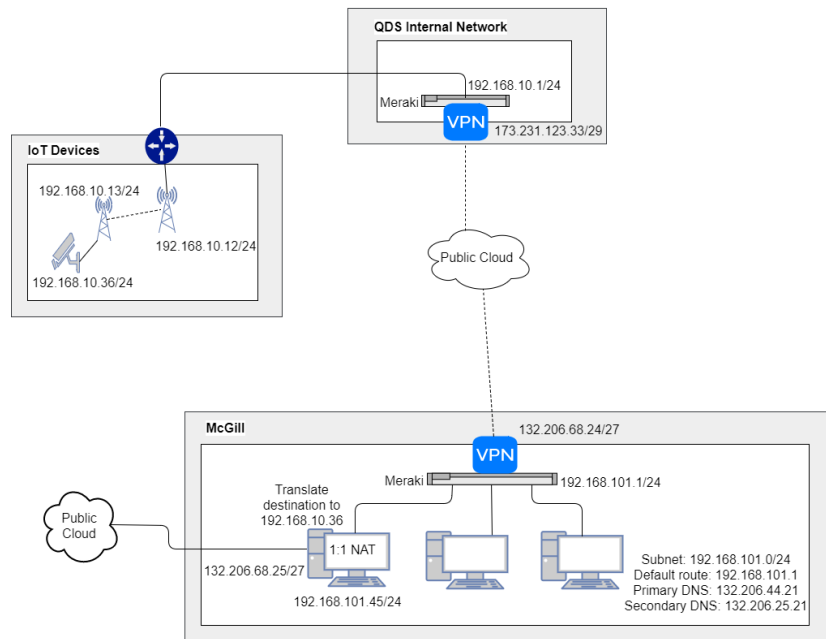
	
	Cisco Meraki MX84
Stateful Firewall Throughput	500 Mbits/s
Advanced Security Throughput	320 Mbits/s
Recommended Maximum Users	200
Maximum concurrent VPN tunnels	100

#### 3.3.2.2.2 Network Address Translation configuration

In the scenario of the test-deployment, network address translation is set up to enable a communication pathway from the public network to several specific devices. An example is illustrated in Figure 3.4, where



port 80 on the public IP address maps to a camera on the IoT Network. The actual setup of NAT for the above camera access and other services, running on 192.168.101.45 server in BCR lab is shown in Figure 3.5.



**Figure 3.4:** NAT setup in the pilot deployment.

```
[root@IoT-Device2 ~]# iptables -t nat -L -n --line-numbers
Chain PREROUTING (policy ACCEPT)
num  target      prot opt source                destination
1    DNAT         tcp  --  0.0.0.0/0             132.206.68.25          tcp dpt:80 to:192.168.10.36
2    DNAT         tcp  --  0.0.0.0/0             132.206.68.25          tcp dpt:8080 to:192.168.101.101:80
3    DNAT         udp  --  0.0.0.0/0             132.206.68.25          udp dpt:8080 to:192.168.101.101:80
4    DNAT         tcp  --  0.0.0.0/0             132.206.68.25          tcp dpt:8081 to:192.168.101.101:9191
5    DNAT         udp  --  0.0.0.0/0             132.206.68.25          udp dpt:8081 to:192.168.101.101:9191
6    DNAT         tcp  --  0.0.0.0/0             132.206.68.25          tcp dpt:443 to:192.168.101.101:9494
7    DNAT         udp  --  0.0.0.0/0             132.206.68.25          udp dpt:443 to:192.168.101.101:9494
8    DNAT         tcp  --  0.0.0.0/0             132.206.68.25          tcp dpt:21001 to:192.168.101.101:10001
9    DNAT         udp  --  0.0.0.0/0             132.206.68.25          udp dpt:21001 to:192.168.101.101:10001
10   DNAT         tcp  --  0.0.0.0/0             132.206.68.25          tcp dpt:21050 to:192.168.101.101:10002
11   DNAT         udp  --  0.0.0.0/0             132.206.68.25          udp dpt:21050 to:192.168.101.101:10002
12   DNAT         tcp  --  0.0.0.0/0             132.206.68.25          tcp dpt:554 to:192.168.101.101:22335
13   DNAT         udp  --  0.0.0.0/0             132.206.68.25          udp dpt:554 to:192.168.101.101:22335
14   DNAT         tcp  --  0.0.0.0/0             132.206.68.25          tcp dpt:1880 to:192.168.101.101:1880
15   DNAT         udp  --  0.0.0.0/0             132.206.68.25          udp dpt:1880 to:192.168.101.101:1880

Chain INPUT (policy ACCEPT)
num  target      prot opt source                destination

Chain OUTPUT (policy ACCEPT)
num  target      prot opt source                destination

Chain POSTROUTING (policy ACCEPT)
num  target      prot opt source                destination
1    RETURN     all  --  192.168.122.0/24      224.0.0.0/24
2    RETURN     all  --  192.168.122.0/24      255.255.255.255
3    MASQUERADE tcp  --  192.168.122.0/24      !192.168.122.0/24      masq ports: 1024-65535
4    MASQUERADE udp  --  192.168.122.0/24      !192.168.122.0/24      masq ports: 1024-65535
5    MASQUERADE all  --  192.168.122.0/24      !192.168.122.0/24
6    MASQUERADE all  --  0.0.0.0/0             0.0.0.0/0
```

**Figure 3.5:** Actual NAT setups.

### 3.3.3 Sensing devices

#### 3.3.3.1 Cameras

**Table 3.10:** Camera functional specifications.

				
	Axis Q6128-E	HikVision DS-2DF6236-AEL	HikVision DS-2CD4585F-IZH	Panasonic WV-SW598A
Max Resolution, FPS	3840x2160, 30	1920x1080, 30	4096x2160, 24	1920x1080, 30
PTZ	Yes	Yes	No zoom	Yes
Bit rate	Up to 50Mbps	Up to 16Mbps	Up to 16Mbps	Up to 14Mbps
API	ONVIF S/G, CGI	ONVIF, CGI	ONVIF, CGI	ONVIF S/G, CGI
Other		36x optical zoom, optical defog	People counting, IR	30x optical zoom, waterproof, super dynamic
Deployed Locations	PKH5, SUH28	MSH6, SCH6	ONH4	SCH10

To produce data with sufficient resolution and quality for video analytics, high-definition (HD) and ultra-high-definition (UHD) IP cameras are utilized for the pilot deployment. All of our deployed cameras can provide at least 1920×1080 resolution and 24 frames per second (FPS), compressed using H.264 and MJPEG encoding format. Pan-tilt-zoom (PTZ) functionality is also included in several selected models, allowing directional and zooming control from a remote location. These PTZ cameras can be adjusted in real-time to capture views around their install locations for applications such as object and person tracking. Overall, these features allow our cameras to be versatile and offer a variety of multi-purpose video data to Smart City applications and services.

4 camera models from 3 different vendors were selected: Axis Q6128-E (x2), Hikvision DS-2CD4585F-IZH (x1), Hikvision DS-2DF6236-AEL (x2), and Panasonic WV-SW598A (x1). The heterogeneity in camera selections is intended to allow investigation on the interoperability between different manufacturers and to reproduce the diversity of devices expected in a Smart City scenario. Table 3.10 summarizes the functional specifications of each camera model.

All cameras offer user authentication, role differentiation, and IP address filtering as security measures. User access to certain camera functions such as PTZ control, recording configurations, and live view quality is restricted based on role assignments. For instance, a client user is only allowed to view real-time footage of the camera while the operator role can change resolution, frame rate, and bit rate settings. This allows administrators to define access control for each camera, limiting users to only a finite set of configurations required for their operations.

For the ease of integration, the cameras should support ONVIF standards. However, during the operation, it is shown that ONVIF compatibility is not enough as some features of ONVIF camera may not be recognized by the VMS software (PTZ feature of Panasonic camera was not recognized). We suggest that


for a larger scale installation, sample test must be done for compatibility between the cameras and the VMS software before a mass purchase/deployment. Customer support may also varies from vendors to vendors, in our experience, Panasonic support was not as good as the other two brands. Regarding the pricing aspect, there could be a huge difference between brands with the same feature set, within the duration of the project so far, we don't see any advantage of Axis consider its high price tag.

### 3.3.3.2 Radars

Radars are used to perform traffic monitoring and statistics measurements, they can be used in locations that prohibit the use of cameras due to privacy issues or limited access to high bandwidth data transport network. The radars detect moving objects and their speeds by transmitting a radio signal and measuring the amount of time before the signal returns after hitting an object.



The traffic radar deployed for this Montreal IoT smart city project was the Geolux RSS-2-300 T Speed Sensor. It can communicate with the network over the RS-232 interface and can detect vehicles up to 400m away. The radar's capabilities are summarized in Table 3.11 and more details can be found in Appendix C.

**Table 3.11:** Traffic radar specifications

	 <p>GeoluxRSS-2-300 T</p>
Protocols	RS-232, RS-485, CAN, Alarm open-drain outputs
Max Detection Range	400m
Measurement Precision	+/- 1 km/h
Measurement Range	5 km/h to 336 km/h
Deployed Locations	ONH4, SUH28, MSH6, PKH5, SCH6

### 3.3.3.3 RFID

**Table 3.12:** RFID reader specifications

	 <p>FEIG ISC.LRU1002 (Pre-2017 model)</p>	 <p>BW-BLEG-WME</p>
Protocols	TCP/IP, RS232, USB	TCP/IP, Ethernet
Tag Support	Class1 Gen2 V1, Class1 Gen2 V2	Read/write Beacon, iBeacon Tag
Max Number of Antennas	4	1 (integrated ceramic antenna)
Max Antenna Output	2 W	2 mW
Deployed Locations	BL1200	BL1200

RFID technology has many potential applications in a smart city for identification purpose. The RFID reader deployed for field testing is a Pre-2017 model FEIG ISC.LRU1002. This device is capable of reading and communicating with EPC Class1 Gen2 tags as well as EPC Class1 Gen2 V2 which features secure encrypted

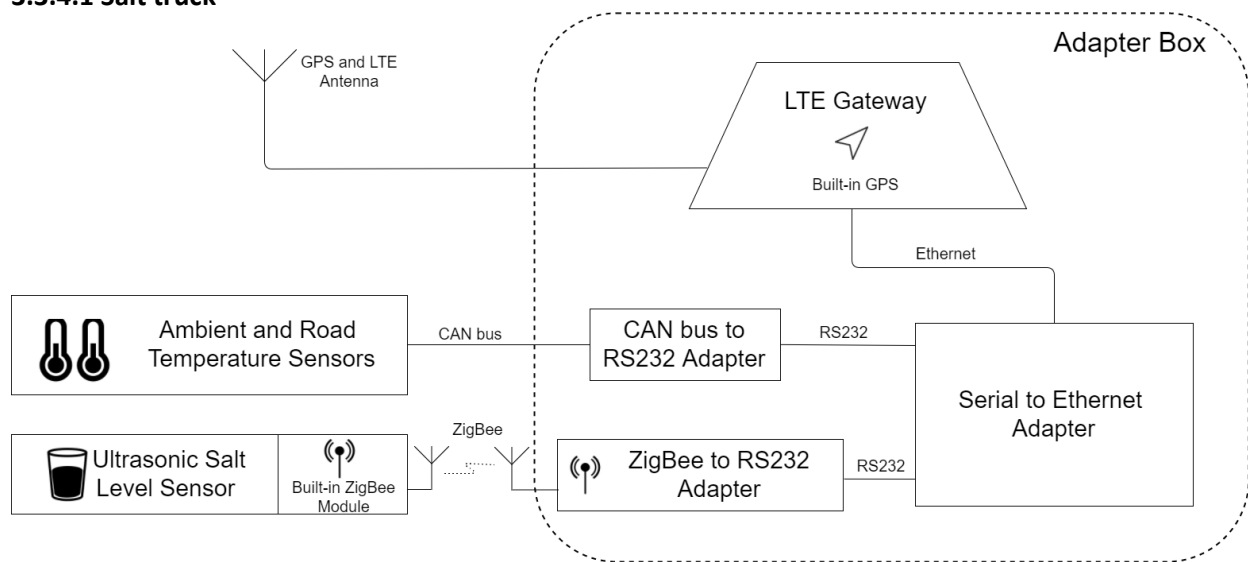


communication. The device can supply an output power of 2 W to each of four possible antennas to energize the passive RFID tags for communications. In addition a Bluetooth Low Energy (BLE) RFID technology is also investigated. The BLE reader that is used is BW-BLEG-WME which works with BLE active RFID tags. Active tags have integrated batteries so it can achieve a higher communication range. A summary of the readers' capabilities can be seen in Table 3.12 and the more detailed datasheet can be found in Appendix C.

### 3.3.4 Mobile sensors

In the pilot deployment mobile sensors are service vehicle equipped with several types of sensors such as temperature sensors, level sensors, GPS, cameras and communication devices such as adapters and cellular gateways to provide readings and logging of environmental factors when the vehicle is in service. This type of sensors is very important as it can cover a wider sensing area, even cover those locations that are not feasible for installing stationary sensors.

#### 3.3.4.1 Salt truck



**Figure 3.6:** Mobile sensors connection diagram for the salt truck.


The first vehicle used for mobile sensors is a salt truck which is equipped with a temperature sensor for measuring both the ambient and the road surface temperatures, an ultrasonic level sensor for measuring the salt level in the container, a GPS sensor for logging the location of the vehicle, 3 adapters and a LTE gateway for communication purpose. The connection diagram is presented in Figure 3.6 with all the sensor modules, communication mediums and adapters. In the deployment, the LTE gateway and all the adapters are enclosed together in side one box. The physical installation locations of the devices are illustrated in Figure 3.7. The adapter box is installed inside the cabin behind the seat for protection from the elements, and the salt level sensor is placed on a grate overlooking the salt container. Mounted to the top of the truck roof is the GPS and LTE antennas to collect and send data.



Figure 3.7: Installations of devices on the salt truck.

3.3.4.1.1 LTE Gateway

Table 3.13: Sierra Wireless Airlink GX450 specifications


	 <p>Sierra Wireless Airlink GX450</p>
Protocols	Ethernet, RS-232, digital I/O, USB, Cellular, Wi-Fi, SNMPv1, SMS commands, Telnet, SSH
Security	Up to 5 VPN tunnels, WEP, WPA-PSK, WPA2-PSK
Operating Temperature	-30C to 70C
Storage Temperature	-40C to 85C
Ruggedness	Military Spec MIL-STD-810G conformance to shock, vibration, thermal shock, and humidity

An LTE gateway is mounted on the salt truck seen in Figure 3.7 to allow mobile data transfer from the sensors to the database. The device deployed as the gateway is a Sierra Wireless Airlink GX450. It is capable of connectivity through both 802.11b/g/n Wi-Fi and LTE but only LTE connection is used in this scenario. It features RS-232 serial, USB, Ethernet, and GPS antenna connectors. Table 3.13 highlights device information and further specifications can be found in Appendix C.

#### 3.3.4.1.2 Temperature Sensor


Temperature sensors were attached on the salt truck to measure ambient and road temperature. The temperature sensor installed is the CVG RoadWatch SS along with the CVG RoadWatch SS RS-232 adapter. It is capable of accuracy within 2°F to 6°F. It features a 15-degree field of view and 1/10 second response time. Table 3.15 summarizes the device information and more information can be found in Appendix C.

**Table 3.14:** Massa RoadWatch SS Specifications.

	 <p>CVG Roadwatch SS</p>
Type	Passive Infrared
Weight	11 oz
Temperature Range	Road surface: -40F to 150F Road accuracy: Within 2F (23F to 41F ambient) Within 6F (-40F to 23F, 41F to 150F) Air: -40F to 131F Air accuracy: Within 2F (-40F to 131 F)
Response Time	1/10 seconds
Field of View	15 degrees

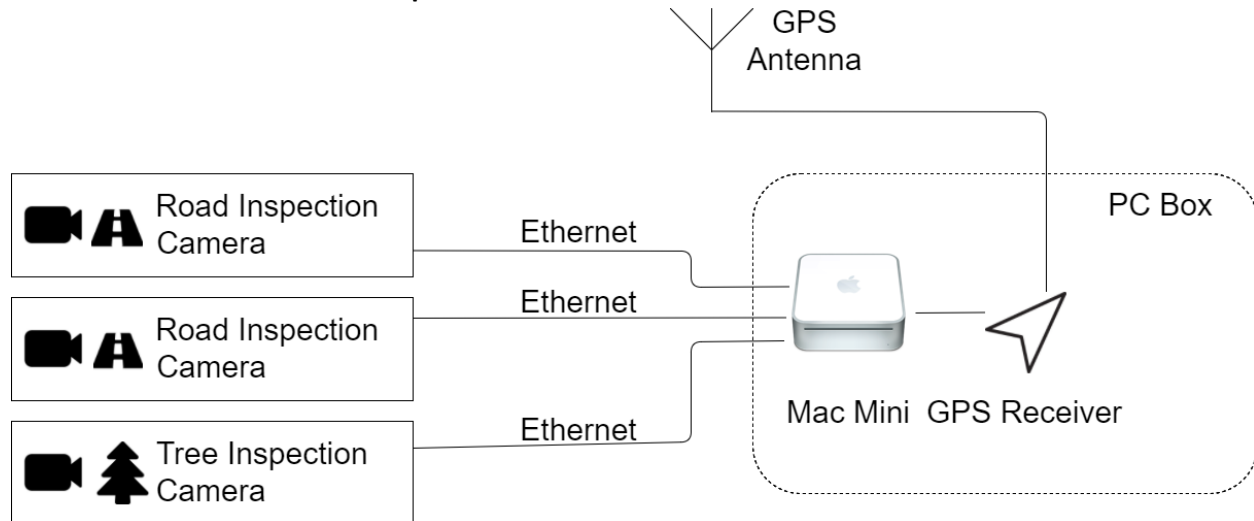
#### 3.3.4.1.3 Level Sensor

**Table 3.15:** Massa M3 Level Sensor specifications.

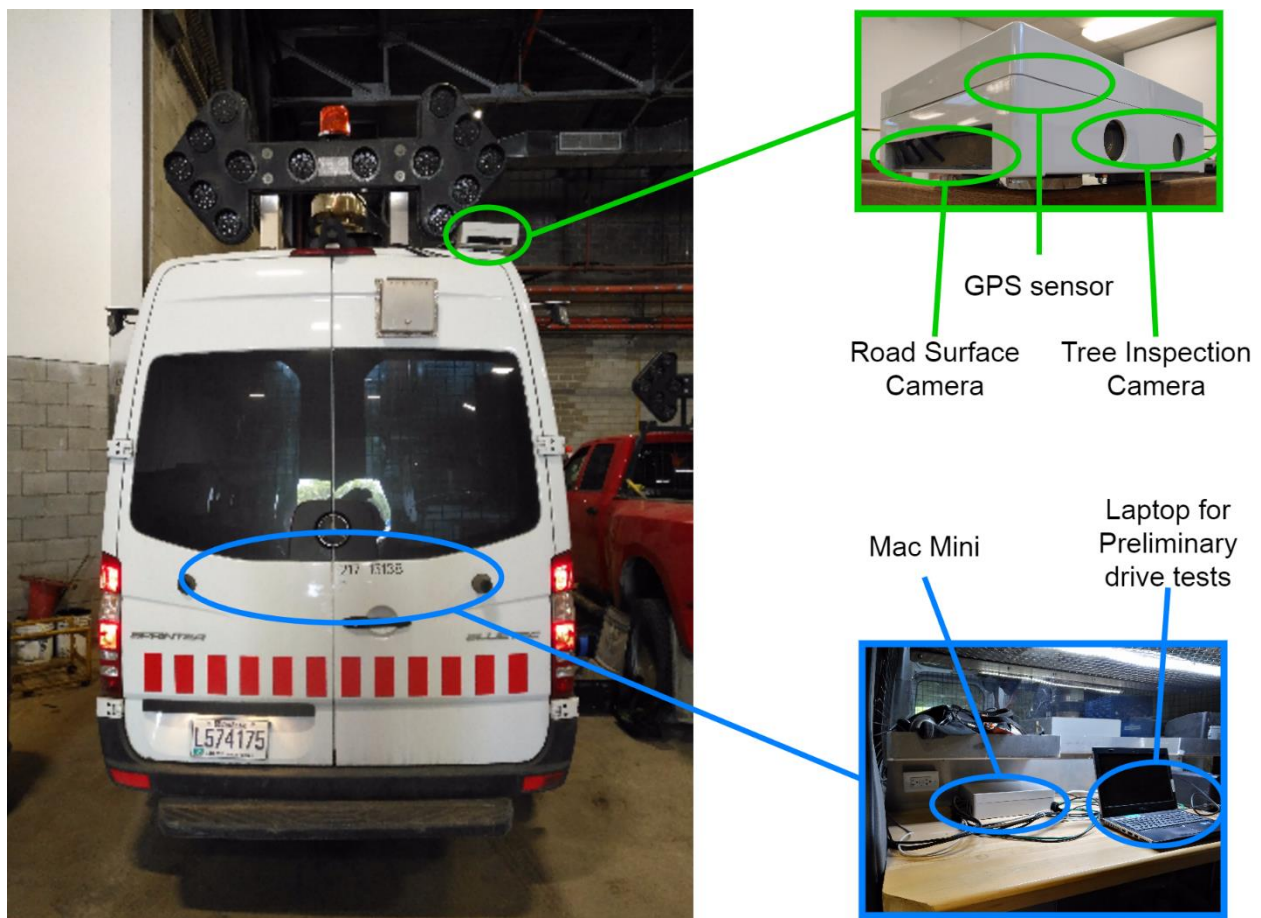
	 <p>Massa M3 Wireless Tank Level Sensor</p>
Operating temperature	-30C to 65C
Ultrasonic range response time	150ms to 500ms
Data acquisition interval	Programmable 10s to 194 days
Battery life	3 years
Power	3 Lithium Energizer AA batteries

To monitor the salt level in the truck container, the Massa Model M3 wireless tank level sensor and digi gateway were installed. The device is an ultrasonic level sensor and has a built in ZigBee module to send information wirelessly. It is reprogrammable and has an ultrasonic range response time of 150ms to 500ms. Table 3.15 presents basic specifications and more can be found in Appendix C.

### 3.3.4.2 Road Surface and tree inspection Vehicle



**Figure 3.8:** Block diagram of road surface and tree inspection vehicle setup.



**Figure 3.9:** Road and tree inspection vehicle camera installation setup.


The road surface and tree inspection vehicle will drive around Montreal collecting video of roads and trees along the road to enable the city of detection of pothole locations and help identify tree sickness or maintenance requirements. The actual detection/identification algorithms (TBD in the next steps) would be implemented in the data analysis center based on the recorded video streams. The van is fitted with two 3D video recording devices with video footages stored onto a hard drive. The contents of the hard drive could be uploaded to a data processing center by embedded 802.11n WiFi interface (when in a Montreal Wi-Fi area) or could be offloaded through much faster USB 3.0 or 1Gbps Ethernet wired interface. The interconnectivity of devices can be seen in the block diagram in Figure 3.8.

Physical installation setup can be seen in Figure 3.9. The road surface and the tree inspection cameras are installed into one housing and mounted at the back of the vehicle for a clear view. The road surface camera is looking downwards to the road at the back of the vehicle and the tree inspection camera is facing sideways to the right side of the vehicle for tree inspection. These devices connects to a Mac Mini inside the truck which will stored videos from the cameras and also store the GPS logs from the GPS sensor. The videos are timestamped at the time of recording so that latter processing can match the location of sick trees or pot holes with the location information from the GPS logs.

#### 3.3.4.2.1 Road Surface Cameras


Two FLIR Grasshopper3 cameras were mounted on the vehicle to record road surfaces. It features a 2.3 MP camera with 1920x1200 resolution at a frame rate of 48 frames per second, operates between 0C and 50C, and uses the Sony IMX174 sensor (as shown in Table 3.16). This camera is capable of capturing high definition, high speed videos for pothole detection.

**Table 3.16:** Road surface camera.

	 FLIR GrassHopper3 GS3-PGE-23S6C-C
Resolution	1920x1200
Frame Rate	48 FPS
Exposure Range	0.017ms to 32s
Operating Temperature Range	0C to 50C
Dimensions	44mm x 29mm x 58mm

#### 3.3.4.2.2 Tree Inspection Cameras

**Table 3.17:** Tree inspection camera.

	 FLIR BumbleBee2 BB2-08S2C-38
Resolution	1032x776
Frame Rate	20 FPS
Aperture	f/2.0
Exposure Range	0.03ms to 66.63ms
Operating Temperature Range	0C to 45C
Operating Humidity	20%-80% (No condensation)
Dimensions	157mm x 36mm x 47.4mm

For tree inspection purpose, a 3D vision stereoscopic camera is installed. The device used is the FLIR BumbleBee2 BB2-08S2C-38. It is 0.8 MP, supports a resolution of 1032x776 at 20 frames per second, and

uses the Sony ICX204 sensor. Further information regarding this camera is summarized in Table 3.17 and its datasheet in Appendix C.

## 3.4 Configuration summary

### 3.4.1 Time synchronization

In a full-scale IoT smart city deployment, it can be expected that there will be tens of thousands of devices sending their data to processing centers. Unfortunately, communication latencies are highly variable and there is no accurate way of predicting how long it will take for a data packet to reach its target destination. As a result, synchronized time stamps are important for correlating data and maintaining records, and may be critical in time sensitive implementations.

In the deployment system, NTP time synchronization is currently enabled on all IP devices and is synchronized to the 192.168.101.45 management server. This CentOS server was set to run `ntpd`, a network time protocol daemon. The server synchronizes itself with 4 public time servers to determine its own relevant time.

### 3.4.2 IP addressing and NAT

<code>iptables --flush</code> <code>iptables -t mangle --flush</code> <code>iptables -t nat --flush</code>	Clears all iptables rules
<code>route add -net 192.168.10.0 gw 192.168.101.1 netmask 255.255.255.0 ens224</code>	Adds the IoT subnet route to the route table
<code>sysctl -w net.ipv4.ip_forward=1</code> <code>iptables -t nat -A PREROUTING -p tcp -d 132.206.68.25 --dport 80 -j DNAT --to-destination 192.168.10.36:80</code> <code>iptables -t nat -A PREROUTING -p tcp -d 132.206.68.25 --dport 8080 -j DNAT --to-destination 192.168.101.101:80</code> <code>iptables -t nat -A POSTROUTING -j MASQUERADE</code>	Enables and configures the NAT address mappings

**Figure 3.10:** Scripts that enables and maps network address translations.

In the connected world of the Internet, IP addresses are the unified identifications. As a result, it is expected that all networking-enabled devices including IoT devices in the future will communicate using IP addresses. There are two generations of IP addressing: IPv4 and IPv6. IPv4 is the most widely used addressing today, however, with the growth in number of IoT devices, the offered identification volume of IPv4 (32 bits) is no longer sufficient and IPv6 is developed as a replacement for IPv4 with a lot of rooms for future growth in volume (128 bits) and features.

Currently, all the devices in the test deployment system are addressed using IPv4. Regarding the IPv6 testing, partial connection between a PC and some deployed devices (cameras, wifi radios) over IPv6 were successfully tested. However, a full-scale testing and conversion to IPv6 was not yet done due to the time limitation. It is noted that most of the devices in the project can support both IPv4 and IPv6 running in parallel.

To connect some of the IoT devices to public network access (for example for testing QdS deployed camera streaming to Cloud based VMS), or to connect some of the mobile IoT devices from a public network to our centralized, private network based, IoT devices control and data center, a virtual machine CentOS server is set up as a routing device. The virtual machine has access to several network interface cards with one card connected to the public network with IP address 132.206.68.25 and another referenced on the IoT server subnet. To allow beyond a 1:1 mapping for the public IP address, NAT using communication ports are used to differentiate different devices in the deployment network.

Setting up network address translation was accomplished by use of *iptables* and *route*. A service was created to run the script as shown in Figure 3.10 at startup. The script clears all iptable rules, adds the route rule for the 192.168.10.x subnet and initializes all the network address translation settings.



### 3.4.3 WiFi radios

As the Wi-Fi radios were deployed in a busy section of downtown Montreal, they were susceptible to a large amount of interference and noise. To provide the maximum available throughput, the radio channel frequency was set to the one featuring the lowest interference level using the spectrum analyzer (build-in tool in the Wi-Fi radios). In an effort to maintain throughput stability, a minimum channel width of 10 MHz was selected to avoid interference from other devices. Output power is also set to a minimum available level of -4 dBm to reduce interference to other Wi-Fi radios. Because the deployed WiFi links have relatively short distance, this minimum power level and small BW are still sufficient for reliable link connection.

The Ubiquiti Wi-Fi radios are enabled in such a manner as to maintain high levels of security, but to allow easy experimentation with settings and tools. Each ST was locked to its accompanying AP to allow it to maintain a connection to the proper source. The wireless communication channel between the two devices was also encrypted via WPA2-PSK. Each of these configuration setting can be seen below in Figure 3.11 which shows a screenshot of the webserver wireless settings.

**Figure 3.11:** Configuration page for wireless settings.

The enabled Ping Watchdog, will force auto reboot of the radio device in case the connection to the server is lost. When an AP device loses connection to the main system gateway or if a ST loses connection to the connected device for a successive period of 6 minutes, it is set to restart under Ping Watchdog. This feature helps with auto-recovery/self-healing of the system in case of random issues. In the QdS deployed case there were/are some LAN port stability issues of some of the Wi-Fi devices, which occasionally will disconnect the cameras from the radios. The enabled watchdog help to recover from these situations, and keeps the system running until the faulty radios are replaced. All IoT devices should have some function like Watchdog to help with self-recovery from potential issues, otherwise a physical visit to a given IoT Device might be needed and costly.

SNMP is enabled to allow for SNMP management protocol testing. **NOTE:** only SNMP v1 was supported, which is not as secure as the latest SNMP v3 version.

The Web Server provided an intuitive interface for changing configurations and running tests in an experimental setting. **NOTE:** this could be normally disabled to increase security.

The SSH Server was enabled to provide SSH access to the central management platform. **NOTE:** normally SSH or SNMP would be used to control the IoT device. SSH could provide more diagnostic options, for example we could run ipref tools from SSH shell to test throughput between different network nodes.

The NTP client was needed to keep time synchronized effectively across all devices.



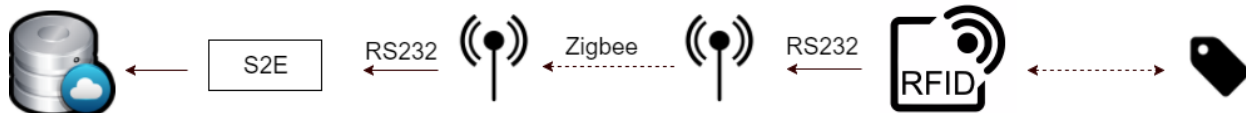
### 3.4.4 Cameras

To connect to the cameras from the radios, Ethernet cable is used. Each camera is enabled to communicate via http and each has its own unique address. (IPv4 address were manually preplanned for this QdS deployment, however, other standard automatic address assignment techniques, such as DHCP, could be used). Video resolution was set initially set to 1280x70 to provide clear video, yet limit the amount of data transmission throughput, for the initial testing. Depending on the need and test purpose, the cameras could be reconfigured to maximum supported resolution. To verify overall network support, tests have been successfully performed.

ONVIF is also enabled on each of the cameras to provide a standard control method.

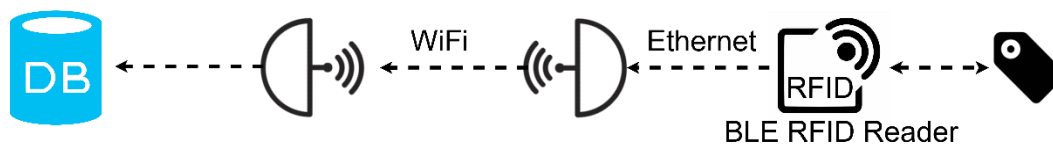
### 3.4.5 RFID

In this project, the objective of the RFID reader is to help maintain inventory records of whether certain items are in storage or in-use.



**Figure 3.12:** Data transmission setup of UHF RFID devices.

For UHF FRID setup, inventory items are marked with RFID transponder tags and carried in and out through a choke reader point where the RFID reader is installed to monitor the logistics. The reader connects to the network via its RS232 port. The RS232 port communicates over a Zigbee radio channel to an RS232 to Ethernet adapter as shown in Figure 3.12. The RFID reader is configured in scan mode, which outputs programmed transponder data through the RS232 port. The port on the Ethernet adapter is monitored to collect data and update the database when a transponder signal is received.



**Figure 3.13:** Data transmission setup of BLE RFID devices.

Similarly setup is used for BLE RFID as shown in Figure 3.13. However, since BLE RFID reader is equipped with an Ethernet port, it can connect directly to the existing WiFi infrastructure.

## 3.5 Throughput and device reliability test results

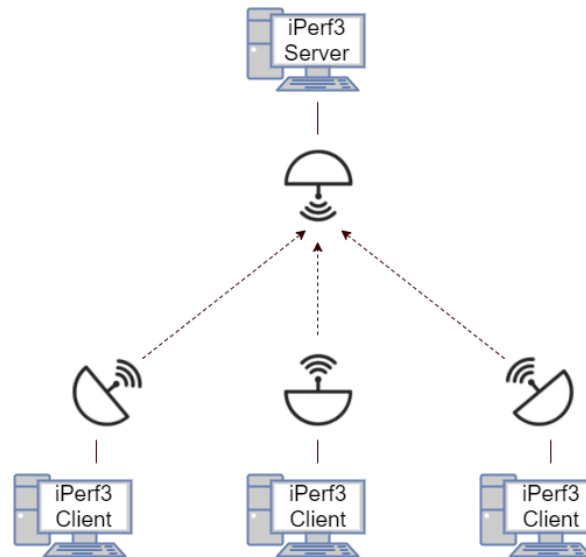
This section will summarize the test results related to the network structure including the device throughput and reliability.

### 3.5.1 WiFi Radios throughput tests

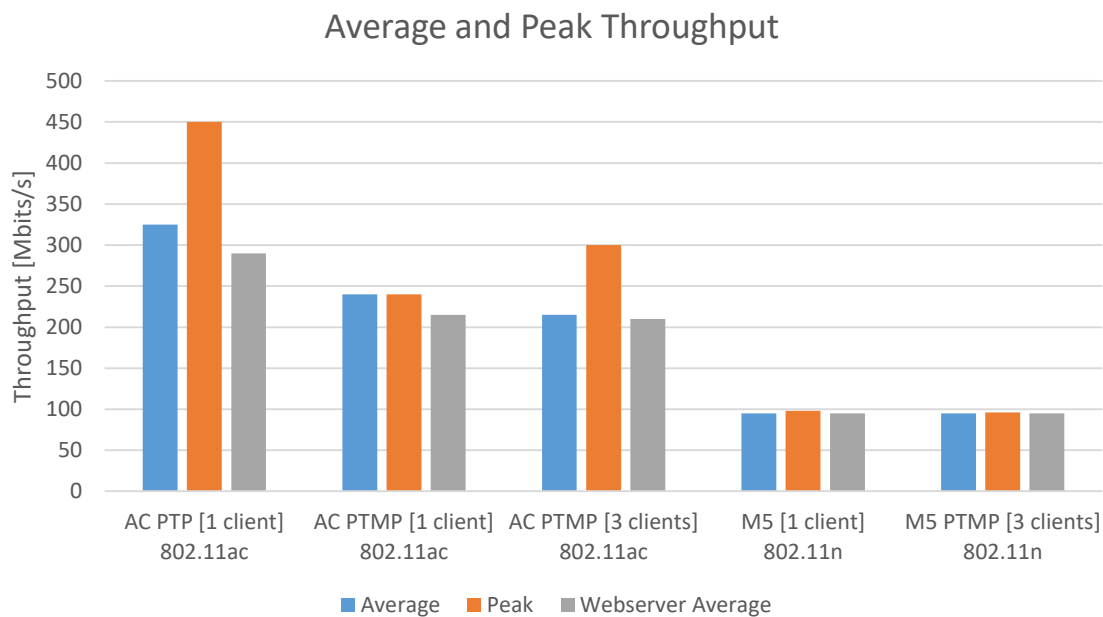
#### 3.5.1.1 Lab throughput test

Experiments were performed in ideal condition in McGill BCRL to verify the maximum throughput advertised by Ubiquiti and to have a reference point for comparison once devices were deployed and subject to external interference. Two models of Ubiquiti radios were used in these tests: NanoBeam NBE-M5-16 (802.11n) and NanoBeam NBE-5AC-16 (802.11ac). Each radio was set to a 40MHz channel width. Each radio type was configured and tested in a PTP and PTMP [3 nodes] setting to compare maximum throughputs with different setups. The ideal maximum throughput is obtained using the iperf3 network performance management tool. This tool allows one device, the client, to generate IP traffic and direct it to another listening device, the server. To generate maximum traffic, the UDP protocol was used. To

simulate an idealized deployment scenario, one radio was initialized as an AP and was connected to the server computer, while the rest designated as ST radios and connected to client computers as shown in Figure 3.14.



**Figure 3.14:** Throughput test setup.



**Figure 3.15:** Average and peak throughput results in Lab environment.

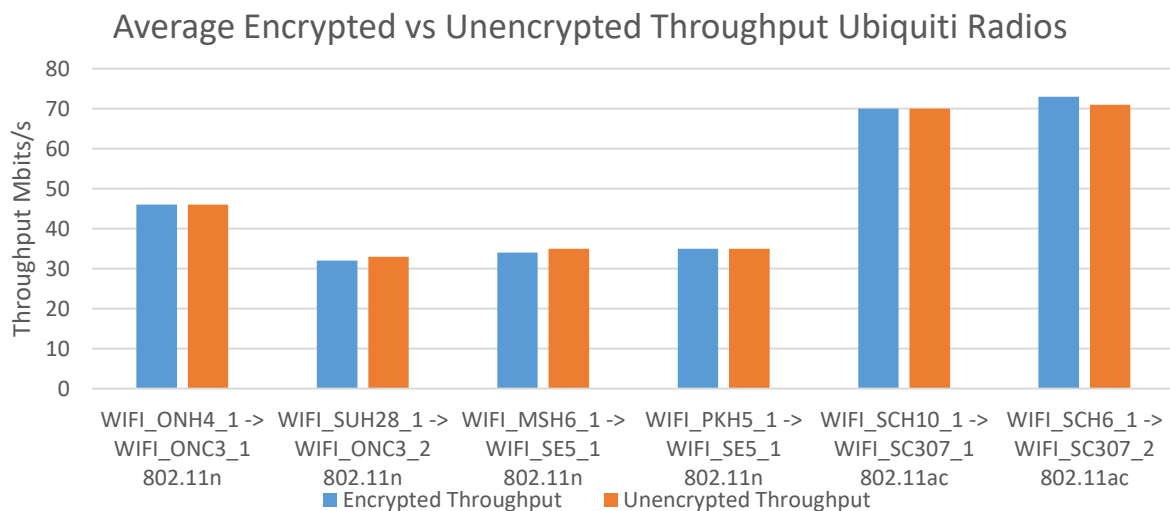
As shown in Figure 3.15, throughput tests verified and put some manufacturer advertised data rates into perspective. Ubiquiti advertises 450+ Mbits/s throughput speeds for the NanoBeam NBE-AC-16 devices, and while a peak of 450 Mbit/s was noted in PTP mode, it was not consistent. The average hovered at 325 Mbit/s. The data rate suffered a large performance hit when the AC radio was switched to the PTMP configuration mode setting, even when only 1 client is connected. With the configuration change the throughput dropped to an average of 240 Mbits/s. While adding an additional two nodes, the throughput dropped an additional 25 Mbits/s to 215 Mbits/s, but this was a rather insignificant drop in comparison to the switch from PTP to PTMP modes. Regardless of PTP or PTMP physical configuration, the M5 radios'

average and peak throughputs remained near 100 Mbits/s, which is likely a limit restricted by the devices' 100 Mbps Ethernet LAN port speed (Ubiquiti claims the M5 radios are capable of 150+ Mbits/s wireless transmission throughput). The tests also show that the iperf3 results were very similar to the readings from Ubiquiti AirOS webserver console.

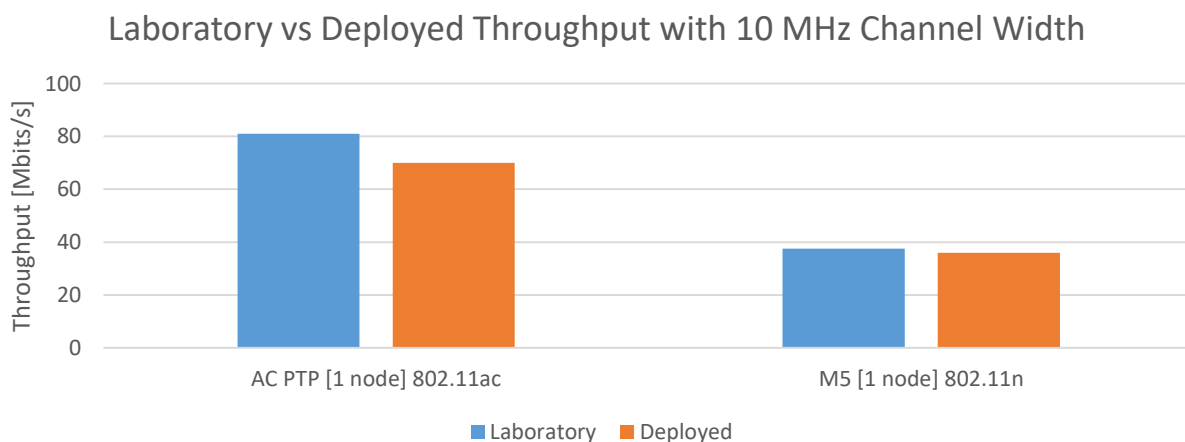
### 3.5.1.2 Deployed throughput test



**Figure 3.16:** Spectrum analysis from Ubiquiti web GUI from an installed device in downtown Montreal.



**Figure 3.17:** Comparison of Encrypted vs. Non-encrypted Deployed Throughput.



**Figure 3.18:** Comparing WiFi radios in laboratory and deployed throughput with 10MHz channel width.

After completion of the basic laboratory throughput experiments, the radios were deployed in downtown Montreal. Throughput tests were also conducted to determine the throughput and interference levels in

an urban deployment setting. To perform the tests, the pre-installed iperf3 tools in Ubiquiti wireless devices were used and results were collected through Ubiquiti AirOS webserver console.

It is noted that the channel width in the deployed radios was decrease to 10MHz to due to the interference sources in downtown Montreal. As shown in Figure 3.16, the publicly available 5GHz frequency band is very busy and a 40MHz channel width is very likely to suffer from severe interference from neighboring Wi-Fi radios. A quick throughput test shows that the throughput of AC radios will drop to 5Mbps using 40MHz channel width setup, this sever degradation is due to strong in-band interference. As the result, it is much better to choose narrower 10MHz channel width which will have less interference, and can deliver 30-45 Mbps for 802.11n and 70 Mbps for 802.11ac throughput.

As shown in Figure 3.17, throughput dropped significantly between lab experiments and deployment implementation. While there was a large variation on device throughput depending on radio location, all radios displayed much lower transfer rates than theoretical maximums. This seems to be a result of much higher interference, larger distances between radios, and a decrease in channel width after deployment. Radio throughput was tested with encryption both enabled and disabled but no significant effect on radio throughput was observed in both cases.

In order to provide a more accurate comparison between the laboratory experiments and the deployed radios, the laboratory results must be scaled down, as larger unimpeded channel widths will always have higher throughput levels than smaller widths. Per Shannon-Hartley Theorem, throughput increases linearly with channel width. As the channel width decreased from 40 MHz to 10 MHz, the original results can be scaled down by a factor of 4 to estimate the 10 MHz laboratory throughput speeds. Due to the LAN port on the M5 radios limiting the throughput, for a more accurate comparison, the advertised rates from Ubiquiti of 150 Mbits/s are scaled down. Results are averaged for radio type in PTP mode in deployment and can be seen in Figure 3.18 that the throughput results in the lab and in the deployed real location are comparable, 70-80Mbps for 802.11ac and 40Mbps for 802.11n radios with 10 MHz BW.

### 3.5.2 Zigbee Radios throughput tests

#### 3.5.2.1 Lab throughput test



Figure 3.19: Zigbee throughput test setup.

Table 3.18: Lab throughput test results on Zigbee radios.

	Unencrypted	Encrypted
1m	31 Kbps	9 Kbps
50m	14 Kbps	7 Kbps

The Zigbee radios were subjected to similar tests to the ones conducted for Wi-Fi radios. To test radio throughput performance, Digi XCTU software was used. This software can display the signal strength between a pair of radios and run a throughput test. In the throughput test, a payload of 230 bytes was selected, as this was the largest payload that allowed the test to function without freezing. The test was performed via loopback, in which the receiving radio's receive and transmit pins were shorted together so that it would transmit the same message back to the radio connected to the XCTU program as shown in Figure 3.19.

The throughput test was conducted with AES encryption enabled in which a four-byte encryption key was used as the passkey. The radios were first tested in the lab with about 1m distance separated between them and then tested in an urban environment to determine the effects of distance on throughput.

Table 3.18 shows the test results with the 2.4GHz radios. It is observed that there is a significant difference in throughput depending on whether the communications were encrypted or not. With distance, the throughput also drops considerably. On the radio datasheet, the Zigbee radio operates at an RF Data Rate of 250 kbits/s, however, this is the maximum bit rate over the air including all the headers and protocol overheads, and not the maximum throughput. It is also important to note that any distance larger than 50m, the 2.4GHz Zigbee radios cannot establish a stable connection, thus this 2.4GHz frequency band may not be suitable for communications in an urban environment due to the effect of signal degradation.

The 900MHz Zigbee radios also went through similar tests. However, as seen from their specs, there is only a marginal difference in terms of throughput when the radios are configured with and without encryption; thus, only tests with encryption enabled were considered

**Table 3.19:** Throughput test results on Zigbee 900MHz radio.

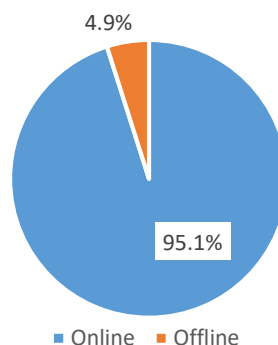
Distance	Throughput (peak/average)	Received Signal Strength	Packet Loss
<b>1m</b>	16.4 / 16.4 Kbps	-40dBm	0%
<b>50m</b>	16.4 / 16.4 Kbps	-40dBm	N/A
<b>340m</b>	12.3 / 11.75 Kbps	-61dBm	7.6%
<b>415m</b>	11 / 2.5 Kbps	-59dBm	33.3%

As shown in Table 3.19, the 900MHz Zigbee radios can reach much further distance than the 2.4GHz radios. As further reach is more suitable for urban area, it is interested to do more extensive tests on the radios with the results of the peak/average throughput, the received signal strength and packet loss are shown in Table 3.19. The results illustrate that the 900MHz radio is able to establish a stable connection up to 340m with a packet loss of 7.6%. However, any distance further than will degrade the signal quality significantly due to path loss and shadowing effect, which results in significantly higher packet loss.

It is note that in the current MSCPS, the 900MHz Zigbee radios were deployed at location BL1200 with approximately 42m link between them

### 3.5.3 Device Reliability

Overall Network Reliability



**Figure 3.20:** Device Reliability measured by Ping.

Through an implemented central management platform described in Chapter 4, statistics are logged to determine the reliability of all deployed devices on the network. A ping test is used to check whether a device is accessible from a management server every 20 seconds. Figure 3.20 presents the average percentage of time that each of the devices were reachable by ping.



During the 12-day measurement test, there was a period where there was a network crash on the Quartier des Spectacles network. This crash lasted around one day and affected four of the radios, two cameras, and two adapters. During this test, there was also a connection problem between one of the radio’s and its attached camera and adapter. This resulted in more than four days of lost connection to the camera and adapter.

The abovementioned network disruptions can happen relatively frequently. Due to installation locations being in a shared location with festival setup teams, the prototype smart city network devices occasionally get disconnected by external parties. . Some of the radios (Nanobeam 5AC models) also have random issues with Ethernet LAN ports which provides an appearance that a connected device disconnected when the real problem is on the radio. These particular radios will need to be replaced under the warranty. Wi-Fi interference sometimes results in brief disconnection, but the effect of this is limited.

3.5.4 Road surface and tree inspection vehicle field test

Table 3.20: Road surface and tree inspection vehicle field test storage results.

	Frame rate	Shutter time	Resolution	Storage consumption
Road surface 3D camera	2 fps	0.24ms	1024x768 (color)	9.6 Mbytes/s 35 GB/hr
Tree inspection 3D camera	2fps	0.24ms	960x600 (MONO 8 bit)	2.3 Mbytes/s 8.3 GB/hr



Figure 3.21: Footage from street inspection camera (top) and tree inspection camera (bottom) for both left and right channels.

The road surface and tree inspection vehicle stores the videos from the cameras and GPS logs locally on the Mac Mini. Due to the huge amount of data generated, the data offload can be done manually or via high-speed WiFi when it is available. The field test results are presented in Table 3.20.

It is noted that due to the power constraint, the current vehicle does not allow the installation of an UPS to back up the power for the Mac Mini for proper shutdown. Further developments can be done in the next steps for further optimizing the operation of the system.

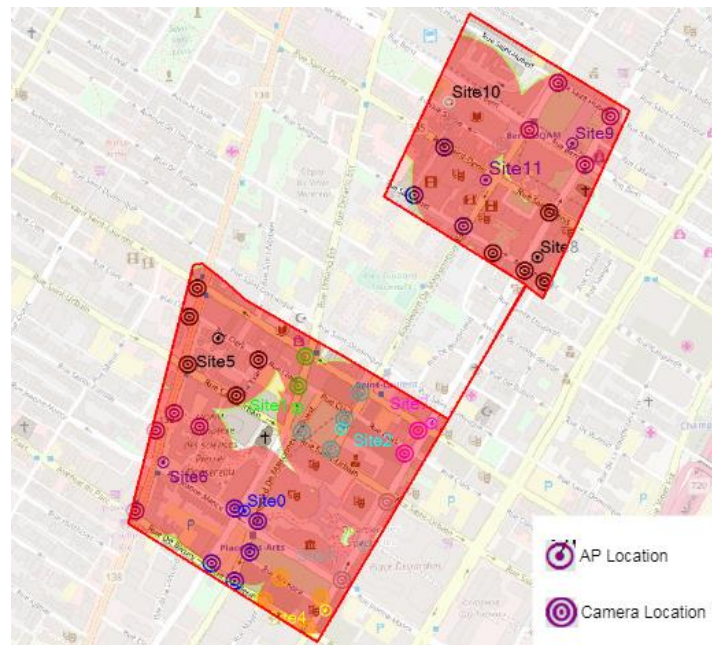
Figure 3.21 shows the footages from both the street and tree inspection cameras for both left and right channels. With the frame rate at 2fps, and vehicle speed at 40km/h (normal travel speed in an urban area), one frame is taken approximately every 5.5m. For a standard configuration, the GPS logs truck's location every second so the accuracy of the interested location can be in the vicinity of  $\pm 11\text{m}$  from the location obtained from the GPS log.

### 3.6 Planning estimation for Montreal downtown and the whole Montreal

With the pilot deployment in place, one natural question is how to expand this setup to the scale of a Smart City. In this section, the scalability of Smart City IoT infrastructure is investigated to demonstrate the issues expected when expanding deployment to city-wide scale. More details regarding the simulations can be found in Appendix J.

#### 3.6.1 Assumptions and the investigated scenarios

In this section, the main interest is on cameras as they are the most bandwidth-demanding type of sensor. Specifically, the focus is on the number of cameras and access points, and bandwidth requirements for supporting a Smart City wireless video surveillance network using 802.11n and 802.11ac-based technologies.

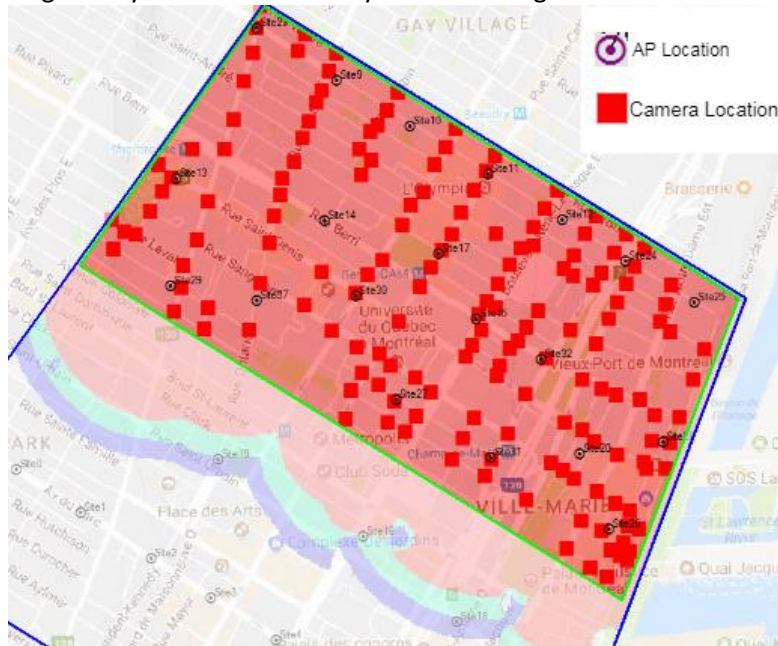


**Figure 3.22:** Signal strength coverage of 802.11n based wireless networks in Scenario 1.

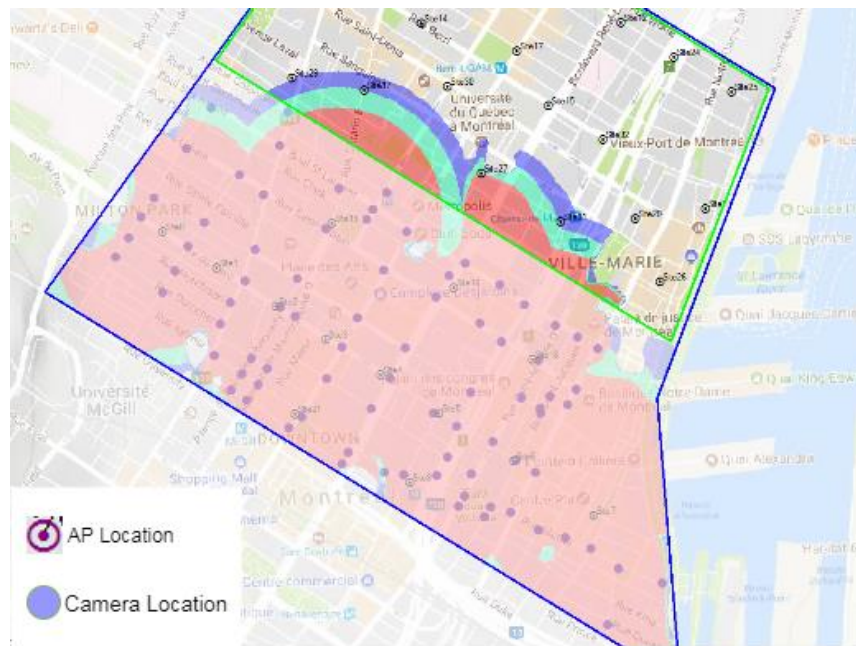
It is observed that in real-life installation, the location of the camera installation may not be in the close proximity of the fiber drop and the aid of wireless communication technologies may be needed to extend the communication to the target surveillance area. As a result, the network topology of wireless devices facilitating these connections is a major consideration for wireless deployment of Smart City surveillance application. In a full-scale deployment with many cameras deployed in all directions around a fiber drop,



a point-to-multipoint deployment where an access point, equipped with an omni-directional antenna, can serve all cameras within the vicinity is more desirable due to economical reason. In addition, signal coverage can be extended to farther locations by using single-hop or multi-hop *relay APs*, which are APs that are connected to gateway via a series of relay wireless bridges.



**Figure 3.23:** Signal strength coverage of high-density 802.11ac-based wireless networks in Scenario 2.



**Figure 3.24:** Signal strength coverage of low-density 802.11ac-based wireless networks in Scenario 3.

In this section, it is assumed that the cameras are placed at numerous intersections and junctions in a target area to record their vehicular and pedestrian traffic activities. Two main camera deployment patterns are investigated: high-density and low-density deployment pattern. In a high-density deployment pattern, a camera is installed at almost every intersection in the target area to provide near complete surveillance coverage of ongoing traffic, excluding some small crossroads where traffic activities are too

infrequent to justify the cost. In a low-density deployment pattern, only intersections with traffic light are monitored by cameras, reducing cost of installation and maintenance while still providing surveillance at major locations in the target area.

Due to the sheer number of possible target areas and the time limitation, *3 scenarios were investigated*. For the *first scenario*, a small expanded urban area from the pilot deployment area is considered, where *the exact locations of fiber drops are known*. This area covers the two regions around Place-des-Arts and Berri-UQAM Metro stations with the total area of approximately  $0.36\text{km}^2$  as shown in Figure 3.22. Camera deployment follows the high-density pattern, where one camera are installed on a street light pole at every intersection, excluding small alleys and intersections already in field of vision of other cameras. 802.11n-based wireless radios and APs are used to establish wireless network coverage in this scenario. Next, we broaden the scope to the slightly larger commercial district area surrounding Quartier des Spectacles with a similar urban planning attributes to maintain the correlation with the previous scenario. Figure 3.23 illustrates the area of *Scenario 2* which covers the northeastern  $1.6\text{km}^2$  half of the commercial district in which a *high-density deployment pattern* is considered. Figure 3.24 depicts the overview of *Scenario 3* which covers the southern  $2.1\text{km}^2$  half of the commercial district area in which a *low-density deployment pattern* is investigated. As the gateway locations are not known, it is assumed that fiber drops are located along a major road through the target areas every 200m. Two distribution patterns of fiber drop locations were considered in each scenario: Scenario 2 in which gateways are placed along a road near the edge of the target surveillance area, and Scenario 3 in which gateways are placed along a road going through the center of the target surveillance area. The wireless network for this scenario is also based on 802.11ac wireless technologies to contrast the deployment density in Scenario 1.

The deployments are simulated using Atoll network planning software based on installation parameters and equipment models obtained from our practical small-scale deployment in the previous section and the bandwidth consumption of each camera is assumed to be 20Mbps, which is the estimated average throughput to stream data at UHD resolution. As a result, an 802.11n wireless device is assumed to effectively support up to 5 cameras and an 802.11ac wireless device is assumed to support up to 13 cameras simultaneously.

### 3.6.2 Simulation results and discussions

As shown in Figure 3.22, the simulation result shows that in *Scenario 1*, there are 12 APs covering 41 cameras across the two areas, and no relay AP is needed. The camera and AP densities for 802.11n based deployment in this scenario are *113 cameras/km<sup>2</sup> and 33 APs/km<sup>2</sup>*, respectively. The result also reveals that *3 different wireless channels* should be provisioned to avoid interference between non-overlapping cells.

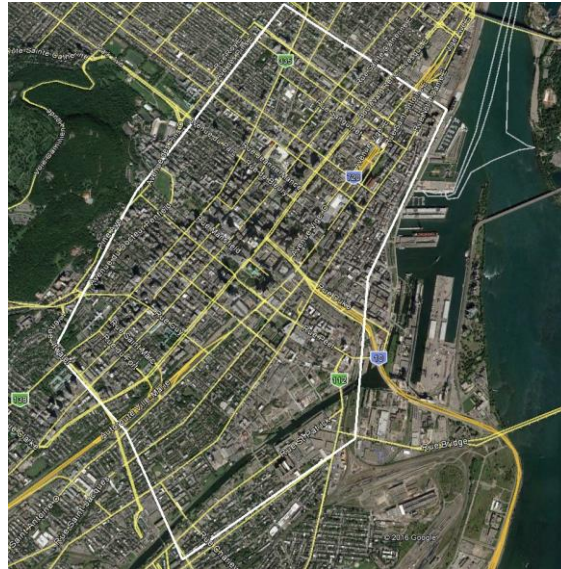
For *Scenario 2*, the use of relay APs is needed to provide the necessary coverage. The wireless bridges between relay APs and the fiber drops are spread out evenly such that each fiber drop serves no more than 2 relay APs, as the high traffic each relay AP delivers can overload the gateway. As shown in Figure 3.23, in total 7 base APs placing at the fiber drops and 13 relay APs are needed to support 161 cameras in this larger  $1.6\text{km}^2$  region. Thus the camera and AP densities for this high-density deployment scenario are *100 cameras/km<sup>2</sup> and 13 APs/km<sup>2</sup>*, respectively. In comparison to Scenario 1, the camera deployment density of Scenario 2 is very similar with 13% difference. This result provides an approximate basis to estimate the number of APs needed to support a high-density deployment using 802.11n and 802.11ac-based implementations. The density of 802.11n-based APs is much higher than 802.11ac-based deployments ( $33\text{ APs/km}^2$  vs  $13\text{ APs/km}^2$ , respectively), which is expected as 802.11n supports significantly lower data rate. This difference in AP density could nullify any cost advantage that 802.11n-based deployment has due to cheaper device cost, meaning that for a large-scale deployment, 802.11ac could be better both in terms of cost and minimizing device quantity.

For the low-density deployment in *Scenario 3*, as shown in Figure 3.24, relay APs are also needed but are placed more sparsely throughout to maximize the number of stations that each AP supports, with only 14

APs are used to cover 85 cameras in the 2.1 km<sup>2</sup> target area, which amounts to *40 cameras/km<sup>2</sup> and 7 APs/km<sup>2</sup>* in camera and AP densities respectively. It is noted that the uneven terrain and the narrow horizontal beam width of the omni-directional antennas could result in irregular coverage patterns and occasional blind spots where wireless signal is not strong enough.

It is observed that low-density deployment *reduces bandwidth requirement by 60%*, requiring only 40 cameras/km<sup>2</sup> as opposed to 100 cameras/km<sup>2</sup> in Scenario 2. Less APs are also required to provide sufficient coverage and throughput support for the cameras, as only approximately 7 APs/km<sup>2</sup> are needed.

#### Scale-up estimation



**Figure 3.25:** Downtown and Quartier de l'Innovation area overview.

Based on these preliminary results, *estimations on the number of cameras and APs, and bandwidth required to deploy over larger areas can be derived*. For example, the downtown Montreal and Quartier de l'Innovation region, which has about 9 km<sup>2</sup> in area as shown in Figure 3.25, has similar urban features and intersection distribution as the areas in our test scenarios. As shown in Table 3.21, based on Scenario 1 results, about 1005 cameras and 293 APs for a high-density 802.11n-based deployment, supporting roughly 20 Gbps total throughput would be required. Similarly, high-density 802.11ac-based deployment would utilize about 900 cameras and 117 APs, according to estimations of Scenario 2, which requires 18 Gbps in throughput. Finally, low-density 802.11ac-based deployment, with 360 cameras and 63 APs, would require 7.2 Gbps to support data transmission. Scaling up to high-density deployment covering *the whole island of Montreal (almost 506 km<sup>2</sup>), roughly 50000 cameras, 6000 APs, and 1 Tbps throughput could be estimated*.

**Table 3.21:** Summary of simulation results for Downtown Montreal and Quartier de l'Innovation area.

Scenarios (Total area: 9km <sup>2</sup> )	Estimated Camera and AP Density	Estimated number of Cameras and APs	Total bandwidth required
Low-Density Scenarios	40 Cameras/km <sup>2</sup> 7 APs/Km <sup>2</sup>	360 Cameras 63 APs	7.2Gbps
High-Density Scenarios	100 Cameras/km <sup>2</sup> 13 APs/Km <sup>2</sup>	900 Cameras 117 APs	18Gbps

While these estimates are preliminary, the huge throughput requirements raise nontrivial questions about how much resources in a given area can Smart City support and how to distribute networking resources to numerous devices and applications.

### 3.7 Conclusion

In this chapter, the summary of proposed tasks and actual achievements were presented and discussed. Then, the outlines of the network structure that were implemented and the device capabilities are presented. Within the project, devices with state-of-the-art, commercially available technologies are purchased and deployed whenever the budget allows to help the determination of functionality, capability limitations in realizing a Smart City. Throughput test results were also performed to illustrate the capacity of the network. It is shown that environmental and configuration factors have strong effects on wireless throughput rates. In downtown Montreal, it is shown that wireless data throughput may drop to below 20% of the manufacturer advertised speeds in some cases. This degradation in performance is mainly due to the destructive interference with public WiFi. In addition, while the Wi-Fi radios did not experience a speed decrease with enabled encryption, not all devices may behave the same way. The ZigBee Pro radios pushed less than 50% of the max throughput when AES encryption was enabled vs disabled. As a result, when deploying a large-scale Smart City network, it is imperative that conservative estimations of device capabilities should be considered and routine tests should be performed to highlight any performance degradation with time.

Moreover, it is expected that further research into communication technologies is necessary to improve the Smart City IoT structure design and performance. Better directed antennas, adaptive beamforming with MIMO technology, dedicated frequency bands are some promising solutions to decrease the interference level experienced in busy city sections and improve connection throughput and stability.

Finally, a simulation-based planning estimations for Montreal downtown and the whole Montreal are performed to give some estimation guidelines for expanding the deployment to a larger area. The numerical results also highlight the scalability issue in terms of bandwidth and hence storage and computing power for processing this sheer amount of data. For the full-scale deployment, the whole city should be divided into a number of hierarchical network sectors based on the connectivity availability and maximum bandwidth support. The data from the IoT network will be then routed to datacenters for storage and processing. The network architecture proposed in this pilot project follows this approach, match closely to the architecture in Chapter 2 and can be generalized to a bigger network. However, the hierarchical planning of network sectors is not addressed in this project as in depth data regarding fiber drop availability and application provisioning should be provided. This is an interesting and important research for the next steps.



# Chapter 4

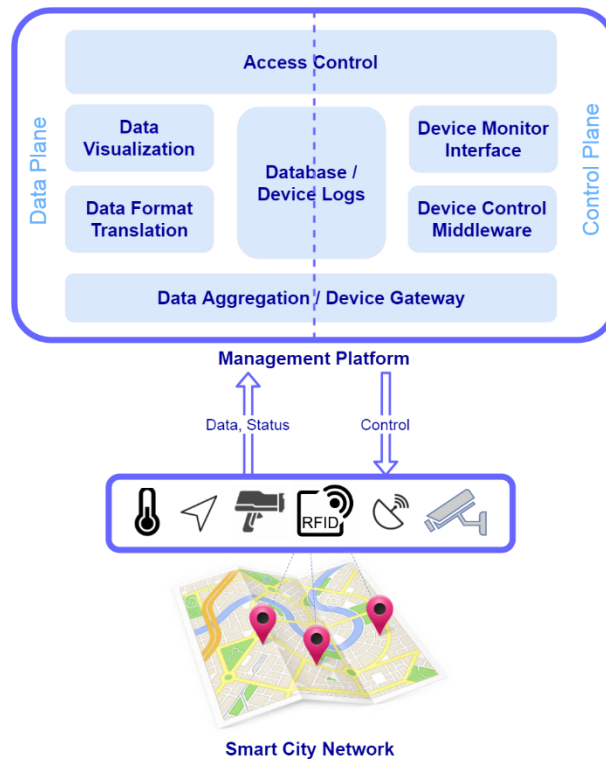
## Data Flow, Storage and Control Structures

In a Smart City, data and devices management is the key factor for the development of the whole system. This is the heart of the system that holds everything in the Smart City together as an integrated system, allowing data collection, device monitoring and control. Investigating this issue allows us to have a more in-depth understanding about the data and traffic characteristics which lays a foundation for any expansion of the system in the future. As a result, an efficient management structure that can be easily expanded is one of the most important considerations. Although many IoT platforms were already commercially developed, careful selection of the flexible and customizable multi-sensor-type and multi-data-communication-type IoT platform must be performed, and it should be noted that configuration of the already commercially available IoT platform for multi-type sensors will not be trivial, a system integration service may need to be sub-contracted besides the IoT platform purchase. As service/device provider “locked in” is one of the factor that should be avoided, a flexible, adaptable and customizable centralized data management structure is one of the most important components in a Smart City setup. In this chapter, two approaches for management structures will be investigated and discussed in the context of the current MSCPS setup: on premise and Cloud based mechanisms. For each approach, the management structures will be separated into two categories data and storage management (data plane), as well as sensor and device control management (control plane). A discussion will also be given to discuss the open issues to be further developed and studied.

### 4.1 The data management structure

In the scenario of a Smart City, as the number of sensors and devices increases, the volume of data becomes overwhelming and the huge number of devices make it very hard to coordinate different tasks. As a result, a centralized data management structure is one of the most important components in a Smart City setup. As shown in Figure 4.1, the basic management platform for a Smart City will receive both the data and status messages from the deployed network, archive this information (data plane), as well as will be capable of sending control instructions to reconfigure the devices (control plane). The basic functionalities of this platform consist of several services as illustrated below.

- **Data Aggregation/Device Gateway** is responsible for establishing connections to the devices and sensors for collecting sensing data and status report from the devices as well as acting as a proxy between the IoT network and the external network to receive and forward reconfiguration commands to the devices.
- **Data Format Translation.** In the context of a Smart City, there may be many types of devices from different manufacturers. Each of the device may produce raw data in different formats and hence, uniformly format the data before storage is needed for the ease of processing the data latter on.



**Figure 4.1:** Basic Data Management Platform for Smart City.

- **Database and Device Log** receive and archive sensing data and status reports from the IoT network. The database also stores the device identity information for locating the device as well as necessary authentication and encryption. For resilience purpose, the content may be distributed across many nodes in a redundancy manner.
- **Data visualization** represents collected data in terms of charts or maps for the ease of management.
- **Device Monitor Interface** provides a user-friendly interface to report the status, notifications or alarms regarding the health of the deployed devices as well as provides controls to reconfigure the devices.
- **Device Control Middleware** serves as a middleware between the user Device Control Interface and Device Gateway to mediate the requests from Device Monitor Interface, translating these requests into appropriate commands to the devices.
- **Access Control.** The whole idea of collecting and storing data is to make it available for future analysis by different applications. In addition, devices in the IoT network can be actuated remotely from other applications. Determine which application can access to which type of data or control which type of devices is very crucial to prevent misuse of information and services provided by the Smart City infrastructure.

To facilitate these functionalities, two main approaches can be identified based on the placement of the data management platform: **On-premise** and **Cloud-based approach**. In the On-premise approach, all the above functionalities are hosted on local computing facilities such as a data center. In this case, the operational, administration and maintenance of these facilities will be on the shoulder of the data center's operator. This approach offer the benefits of performance and data privacy. Since the data sink is placed close to its sources, the performance of data archiving in terms of bandwidth and delay can be guaranteed easily. This is especially advantageous for real-time data. In addition, since the data is stored locally, the operator has the complete control over the data, and its privacy. Privacy control is very important in the context of Smart City where thousands of video streams of citizens are recorded and stored. However, it

can be costly to setup, maintain and scale up this back-end infrastructure to support the growing Smart City IoT network and applications.

In Cloud-based approach, many of these functionalities are designated to the cloud computing facility. Cloud Computing brings about the availability of virtually unlimited storage, processing capacity and computing resources at low cost that can be virtualized and leased to users on demand. To catch up with the trend, many cloud providers such as Amazon, IBM, and Microsoft are racing to adopt their Cloud Computing platforms for delivering various IoT services over the Internet. Many motivations that drive the development of IoT and Cloud Computing integration are identified as below.

- **Interoperability:** Generally, IoT comprises of large number of heterogeneous devices and technologies that often needs to interoperate in seamless manner with each other as well as back-end servers. To cope with this problem, Cloud Computing often offers a unified platform to aggregate data and communication with IoT devices, which greatly simplifies the integration of new devices and streamlines data flows between IoT network and back-end management components.
- **Scalable and rapid deployment:** It is often not easy to scale up an on-premises setup. To increase storage and computing capacity, new physical servers have to be purchased and integrated into the existing system, a lengthy and potentially costly process. Resources have to be allocated for peak performance demands even if most of the time they stay idle. Cloud Computing platform enables rapid resource allocation and integration through well-defined APIs and automated processes, which can cope well with the dynamic demands of the IoT network.
- **Ubiquitous access:** Cloud Computing platform allows IoT applications to be accessed anywhere and anytime through web portals or built-in apps. It enables effective and cheap solution to connect, track, and manage IoT devices, data, back-end operations from anywhere at anytime.

A typical Cloud provider generally offers 3 service models: infrastructure as a service (*IaaS*), platform as a service (*PaaS*), and software as a service (*SaaS*). *IaaS* uses the underlying cloud-based hardware components to provide processing, storage, and networking resources such as virtual machines, virtual local area networks, etc. *PaaS* contains development frameworks built on top of cloud infrastructure virtual resources to provide services such as operating system, programming language execution, database, webserver, etc. Finally, *SaaS* delivers complete applications running on cloud environment. Using SaaS users don't have to worry about the installation, setup and maintenance of the applications. Due to its many benefits, the Cloud Computing is extensively studied and applied recently. However, Cloud Computing is not a one-size-fits-all solution for a Smart City Deployment. There are many limitations to deploy a cloud-based solution compared to an On-premises solution. Firstly, while often advertised as virtually unlimited resources, services are often subjected to arbitrary quotas and limits imposed by the service provider that are not present for an on-premises setup in order to prevent one user from draining all available resources and deny others of their usages. Secondly, compared to on-premises solution which is often implemented in a local data center, cloud-based solution is often implemented in a global network of data centers managed by the cloud provider, which are not necessarily geographically close to the deployment site. This results in higher delay and lower throughput in exchanging data with the cloud, which could impact real-time applications with strict timing requirements. Lastly, most Cloud Computing services are still in development with limited capability, serving more as a framework on which IoT applications can be developed.

However, to implement a complete Smart City IoT application, a lot of additional development and customizations must be done specifically targeted towards Smart City requirements. In this chapter, both On-premise and Cloud-based approaches will be investigated on the data plane to manage the storage, data flow and control of devices in the MSCPS. In each approach, the actual implementation details that were done during the time frame of the project will be described and supplemented by *Appendix E* (for data plane implementation) and *Appendix F* (for control plane implementation).





market, ranging from free open-source programs to costly proprietary enterprise services 3 candidates at different price range were selected for investigation.

- **ZoneMinder**<sup>6</sup> is a free open-source VMS developed on Linux platform by a community of volunteer developers. It provides a web-based user interface for user to add cameras to be monitored, set recording parameters, and view live or archived camera footages. ZoneMinder utilizes FFmpeg, an open-source universal video encoding program, to retrieve and decode video streams from cameras. This implementation prevents ZoneMinder from making changes to camera settings remotely. The video stream is decoded into MJPEG regardless of source video encoding format and the resulting video frames are stored as sequences of JPEG images, which can then be displayed by ZoneMinder integrated live monitor or transcoded to H.264 format and stored as an MP4 or AVI file. In addition, ZoneMinder provides an internal motion detection mechanism by analyzing the stored JPEG image sequences. Finally, ZoneMinder supports user-defined filter to sort and match recorded data for some automated functionalities, such as automatic transcoding of recorded still images to video format and removing old records when storage capacity reaches a defined threshold.
- **Xeoma**<sup>7</sup> is a commercial, cross-platform video surveillance software developed by Felena Soft. It provides functionalities such as recording, live monitoring, and motion detection as modules that can be combined in different arrangements as required by user. Xeoma also uses FFmpeg to retrieve and decode video streams from cameras, though it can archive data as transmitted by camera without transcoding. Thus, it does not support remote modification to camera configuration. Xeoma supports automatic filter to purge old recordings when storage capacity limit is reached.
- **XProtect**<sup>8</sup> is a commercial solution developed by Milestone Systems. It is highly integrated with camera firmware, allowing remote adjustment of camera settings from an internal interface. XProtect is well optimized, allowing functions such as motion detection to be hardware accelerated for faster processing. Video stream is decoded by a closed-source mechanism and stored in the proprietary PIC format, which can only be accessed and transcoded to other video formats (e.g., MKV, AVI) to by an associated proprietary media software.

#### 4.2.1.1.2 Performance Comparison:

The capability of each VMS candidate is investigated through 2 experiments. First, the performance of each software is evaluated when managing one camera video stream at various video resolutions to determine how much resources is required to process and archive video data. Second, how the performance of each VMS candidate change while managing an increasing number of streaming cameras is examined to verify their scalability for Smart City deployment. All tests were done on a virtual machine hosted on a Dell Precision T7600 server with allocated 8 Intel Xeon E5-2620 2.00-GHz CPU cores, 6 GB of RAM, and 1000Mbps Ethernet connection.

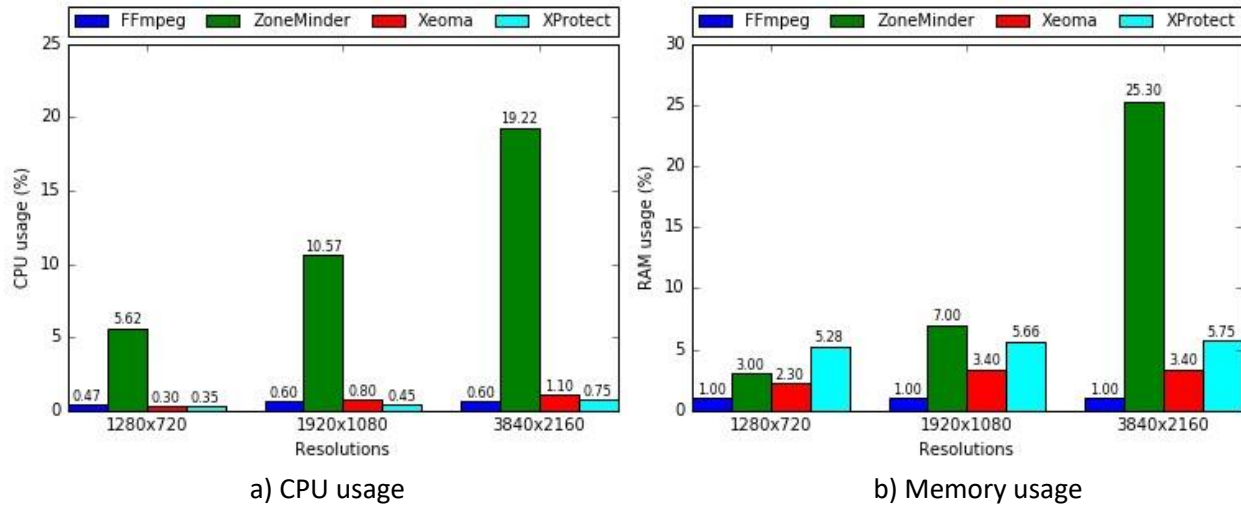
##### Singe Stream Performance test

Image resolution is one of the key factors that determines the amount of information in a video. A higher resolution video such as 3840x2160 (UHD) contains more data but also consumes more resources in processing and storing the content. To measure the resource consumption of each VMS solution, they are set to continuously record 10-minute video segments from a test video stream at 3 different pixel resolutions: 1280 × 720 (HD), 1920 × 1080 (FHD), and 3840 × 2160 (UHD). The frame rate and bit rate of the test video stream are kept constant at 24 FPS and 16 Mbps, respectively. The key performance indicators are the CPU, RAM usage and bandwidth usage of each candidate software. The video archive process is also investigated by measuring the storage size and the frame rate of the recorded videos.

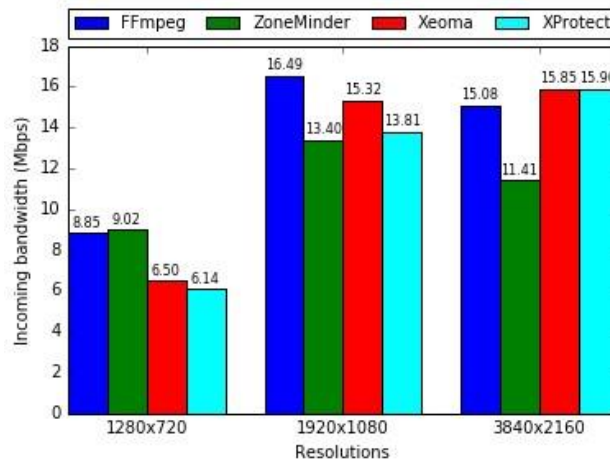
<sup>6</sup> ZoneMinder. Accessed 2017. url: <https://www.zoneminder.com>.

<sup>7</sup> F. Company. Xeoma. Accessed 2017. url: <http://felenasoft.com/xeoma/en/>

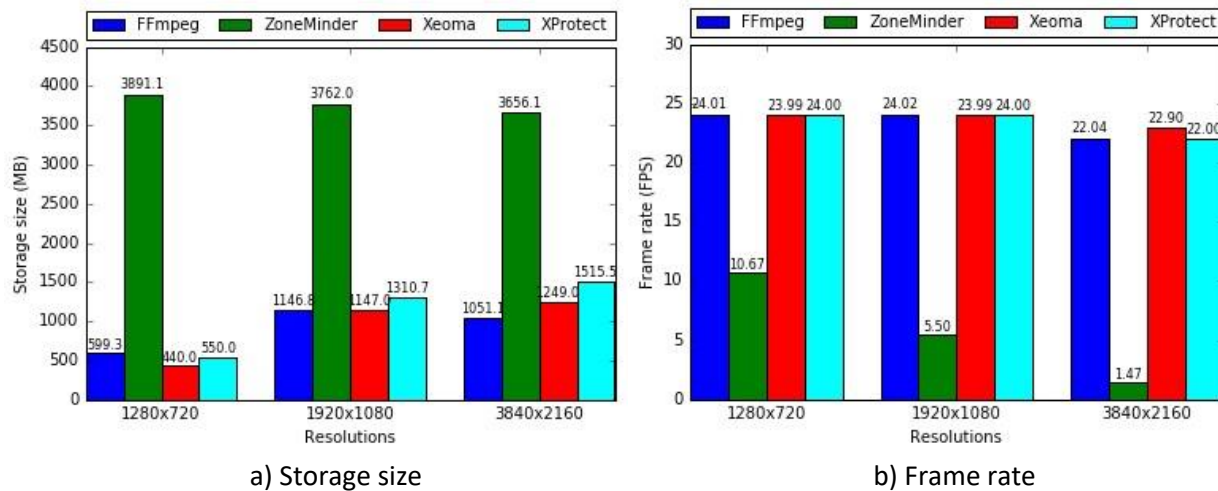
<sup>8</sup> M. Systems. XProtect Professional. Accessed 2017. url: <https://www.milestonesys.com/solutions/platform/video-management-software/xprotect-professional/>



**Figure 4.3:** CPU and memory consumption of the candidate software at different resolutions.



**Figure 4.4:** Network bandwidth consumption of the candidate software at different resolutions.



**Figure 4.5:** Storage size and frame rate of video archives by different solutions.

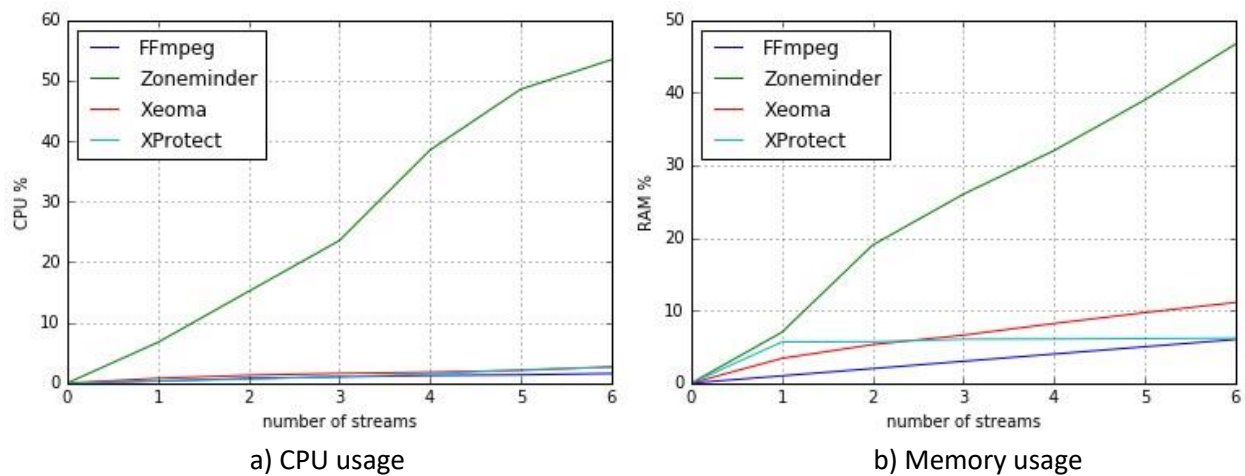
Figure 4.3 shows the CPU and memory consumption of the investigated software. In terms of CPU consumption, there is no significant difference between FFmpeg, Xeoma and XProtect. Regarding the

memory consumption, Xeoma and XProtect is observed to consume slightly more memory than FFmpeg, most likely due to some additional processing for functionalities that are not implemented in FFmpeg. Most notably, ZoneMinder consumes significantly more CPU and memory resources than any other candidates with increasing image resolutions. The higher resource consumption can be attributed to the video decoding process to MJPEG format that is implemented in ZoneMinder. It is important to observe that except for ZoneMinder all other software have roughly the same CPU and memory consumption as the resolution changes.

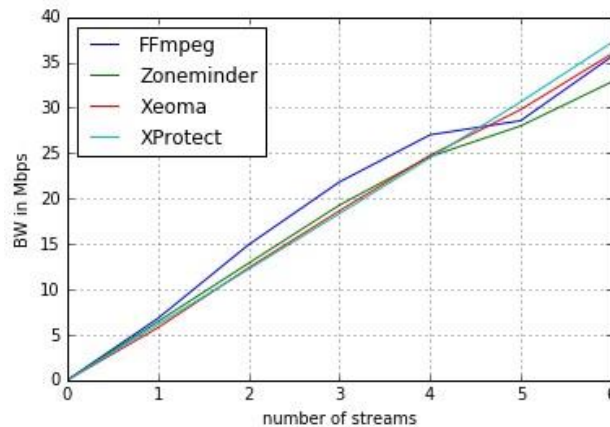
Regarding the bandwidth consumption, as illustrated in Figure 4.4, despite being based on FFmpeg, ZoneMinder only has similar incoming throughput at  $1280 \times 720$  resolution, and then lags behind due to the longer processing time required to the JPEG processing of individual image frames at higher resolutions. On the other hand, FFmpeg, Xeoma, and XProtect maintain their incoming bandwidth consumptions at close to the 16 Mbps maximum bit rate output by the camera.

The inefficiency of operating with MJPEG format extends beyond CPU and memory consumption for ZoneMinder. Figure 4.5 shows that video quality and storage space are also negatively affected. As image resolution increases, the larger size of each video frame requires ZoneMinder to use more time and resources to compress individual frames. This results in lower number of frames being processed in the same duration. Consequently, the frame rate of ZoneMinder video recordings suffers while the storage size remains much larger than archives of FFmpeg, Xeoma, and XProtect.

#### Multiple Stream Performance test



**Figure 4.6:** CPU and memory consumption of the candidate software at different number of streams.



**Figure 4.7:** Network bandwidth consumption of the candidate software at different number of streams.

As Smart City is expected to deploy a large number of cameras, it is interesting to see how the performances of the candidate software scale with the increasing number of devices. The result may be useful to evaluate whether these VMS solutions is suitable for Smart City deployment and may provide a basis for estimating resource requirement for planning a scalable Smart City VSN management layer.

To evaluate the scalability for Smart City, each VMS candidate solution manages up to 6 video streams from the 6 cameras deployed in the data acquisition layer. The video streams are fixed at  $1920 \times 1080$  resolution and 24 FPS frame rate as this is the highest configuration supported by all of our cameras. The bit rates are limited to 6 Mbps to avoid over-congesting the VPN connection without sacrificing too much video quality. The key performance indicators are CPU, RAM, and network bandwidth usage of each VMS candidate solution.

Figure 4.6a shows the CPU and memory consumption of each VMS candidate solution. As expected, the CPU usages of all 4 solutions scale almost linearly with the number of video streams. Similar to the previous experiment, there are no notable differences among FFmpeg, Xeoma, and XProtect. Likewise, ZoneMinder's rate of CPU consumption is much higher than others due to its decoding process as explained earlier.

Regarding the memory usage, it is interesting to observe that the initial memory usage of XProtect is higher than FFmpeg and Xeoma, but it increases by a negligible amount for each additional video stream. The results may imply that the video processing mechanism of XProtect is much more efficient with memory management, though it is more likely that 6 cameras is insufficient to evaluate the scalability of XProtect. In contrast, FFmpeg, ZoneMinder, and Xeoma's memory usages scale linearly with the number of video streams. The usage of FFmpeg is equal to XProtect at 6 streams while Xeoma bypasses XProtect after 3 streams. ZoneMinder, once again, consumes the most memory with almost 50% RAM usage at 6 video streams.

In terms of network bandwidth usage, as shown in Figure 4.7, all 4 VMS solutions uses almost similar amount of bandwidth and scale linearly with the number of video streams. The slight differences are more likely due to variation in routing patterns through the public Internet. This result indicates that the implementation of VMS software has little impact on the data throughput of the network.

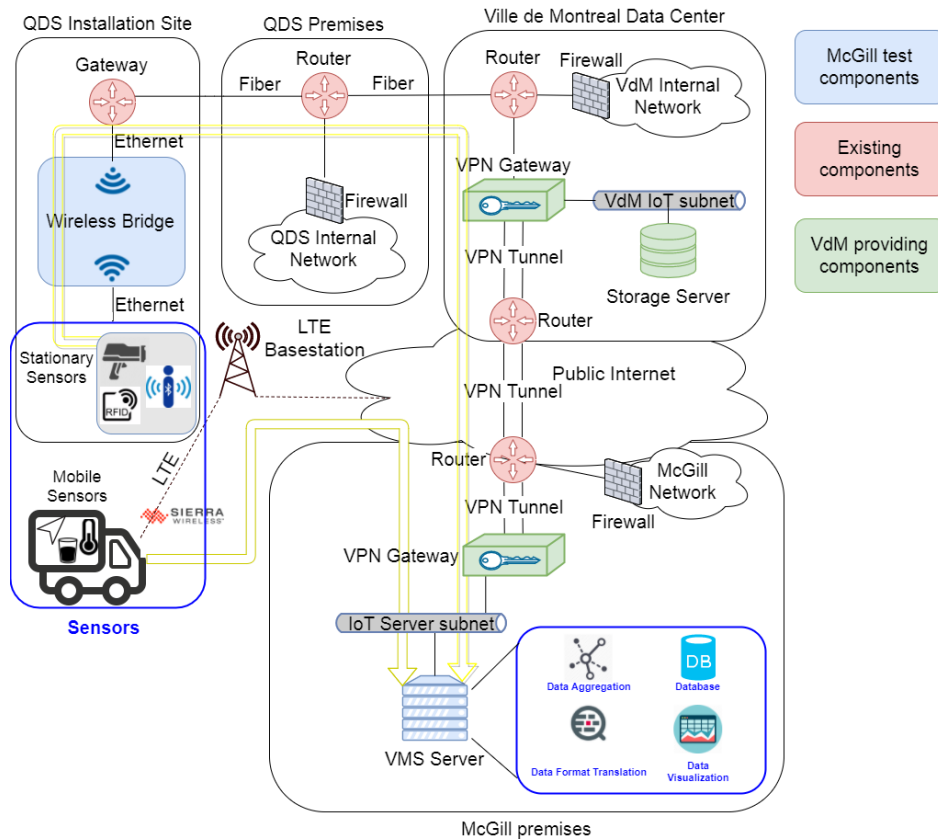
#### 4.2.1.2 Other types of sensors

In this subsection, an overview of a centralized data management and its components will be detailed first. Then, the implementation specifics for on-premises prototype central management platform will be described.

##### 4.2.1.2.1 Overview

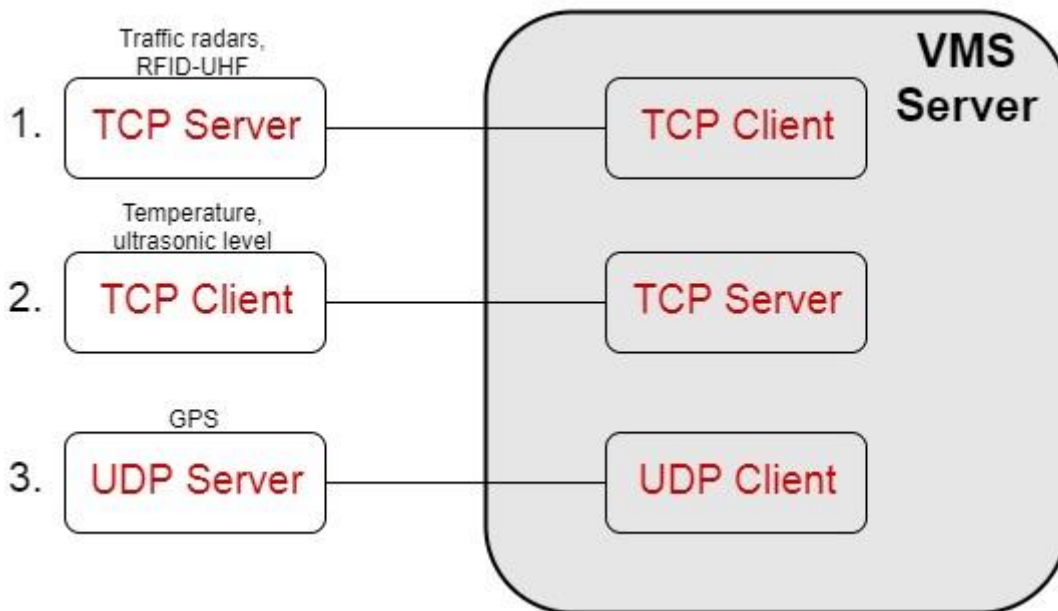
In the pilot deployment, there are different types of live sensors installed including radar sensors, temperature sensors, GPS sensors (etc.) that collect raw data which needs to be stored in a database for further analysis and visualization. However, each type of sensors has a different data format and the way to fetch/query the data is significantly different from one type of sensors to another. As a result, a common platform for aggregating and translating and storing the data to a uniform format is required.

Figure 4.8 illustrates the data flows and the four fundamental components of the on-premises data management platform: data aggregation; data format translation; the data base and the data visualizer. The *data aggregation* is in charge of connection establishment and raw data collection from different types of sensors in the network. *Data format translation* is responsible for parsing the data, reformatting them into a unified format and connecting to the database. The *database* stores the collected data for future processing. The *data visualization* provides a mean for users to visualize the collected data from the database. In the on-premises implementation, all of these components are deployed in a management server.



**Figure 4.8:** On-premises data management.

#### 4.2.1.2.2 Data Aggregation



**Figure 4.9:** Types of socket connections.

Due to the wide range of setups and types of sensors, sensor data are collected in one of the three methods as shown in Figure 4.9. Stationary devices, in particular traffic radars and RFID-UHF readers, have a static IP address, so TCP server is used for data collection because connection to these can be done from



many destinations simultaneously using TCP clients. Mobile devices such as temperature and ultrasonic level sensors cannot use the same approach because their IP addresses are constantly changed. As a result, those sensors send data through their TCP clients. These devices can only send data to one configured device at a time. In addition, Sierra Ailink's GPS sensor does not support TCP client mode and its TCP server mode doesn't work (due to firmware bugs), so in our prototype platform, the GPS send data through UDP server sockets.

#### 4.2.1.2.2 Data Format Translation

##### 4.2.1.2.2.1 Raw data format

##### Stationary sensors

- **Radar sensors.** Table 4.1 shows the raw data format of messages from radar sensors. Basically, there are four types of messages in which two are used for real-time and two are used for statistics purposes. It is noted that the speed values in the reports are scaled up by 10 so scaling down by 10 should be done to obtain the actual speed.

**Table 4.1:** Formatting of traffic radar reports.

Message type	Message formats
<b>Target detected report</b>	<p><b>\$RDTGT,D1,S1,L1,D2,S2,L2,...,Dn,Sn,Ln*CSUM&lt;CR&gt;&lt;LF&gt;</b></p> <p><b>Meaning:</b> This message type is use for real-time report and includes information about all the detected targets in a pre-defined period (currently set at 1s).</p> <p><b>D<sub>i</sub>:</b> The direction of the i<sup>th</sup> target (1 approaching, -1 receding).</p> <p><b>S<sub>i</sub>:</b> The speed of the i<sup>th</sup> detected target (Note: speed = S<sub>i</sub>/10).</p> <p><b>L<sub>i</sub>:</b> The detected level of the signal reflection from the target.</p> <p><b>CSUM:</b> message checksum.</p>
<b>Target counts report</b>	<p><b>\$RDCNT,D,S,L,aprcNT,rcdCNT*CSUM&lt;CR&gt;&lt;LF&gt;</b></p> <p><b>Meaning:</b> This message type is sent whenever a new valid target is detected.</p> <p><b>D:</b> The direction for the new counted target (1 approaching, -1 receding).</p> <p><b>S:</b> The speed for the new counted target (speed = S/10).</p> <p><b>L:</b> The detected level of the signal reflection from the target.</p> <p><b>aprcNT:</b> The cumulative counter for the approaching targets.</p> <p><b>rcdCNT:</b> The cumulative counter for the receding targets.</p> <p><b>CSUM:</b> Message check sum.</p>
<b>Approaching target statistics report</b>	<p><b>\$RDSTA,count,avgSpeed,minSpeed,maxSpeed,roadOCP,tmpCNT*CSUM&lt;CR&gt;&lt;LF&gt;</b></p> <p><b>Meaning:</b> The message is sent periodically at the beginning of each statistics report (currently set at 15 minutes) for <b>approaching</b> targets. <i>All values are relative to the time period from the last sent report.</i></p> <p><b>count:</b> The accumulated count during the report period.</p> <p><b>avgSpeed:</b> The average speed for all approaching targets in the report period.</p> <p><b>minSpeed:</b> The minimum of all approaching targets in the report period.</p> <p><b>maxSpeed:</b> The maximum of all approaching targets in the report period.</p> <p><b>roadOCP:</b> The road occupation percentage for the report period calculated as number of samplings with at least one valid approaching detected target divided by the total number of samplings in the report period.</p> <p><b>tmpCNT:</b> The temporary counter of all approaching targets in the report period.</p> <p><b>CSUM:</b> Message check sum.</p>
<b>Receding target statistics report</b>	<p><b>\$RDSTR,count,avgSpeed,minSpeed,maxSpeed,roadOCP,tmpCNT*CSUM&lt;CR&gt;&lt;LF&gt;</b></p> <p><b>Meaning:</b> The message is sent periodically at the beginning of each statistics report (currently set at 15 minutes) for <b>receding</b> targets. <i>All values are relative to the time period from the last sent report.</i></p>



	<p><b>count:</b> The accumulated count during the report period.</p> <p><b>avgSpeed:</b> The average speed for all approaching targets in the report period.</p> <p><b>minSpeed:</b> The minimum of all approaching targets in the report period.</p> <p><b>maxSpeed:</b> The maximum of all approaching targets in the report period.</p> <p><b>roadOCP:</b> The road occupation percentage for the report period calculated as number of samplings with at least one valid approaching detected target divided by the total number of samplings in the report period.</p> <p>tmpCNT: The temporary counter of all receding targets in the defined time period.</p> <p>CSUM: Message check sum.</p>
--	--

- **RFID-UHF readers.** As shown in Figure 4.10, the raw data from RFID-UHF reader consists of three values, separated by commas. The first value is the UID, second value is the timestamp and third is the antenna number. The timestamp is in a hex format. The first two bytes, converted to decimal represent the hour, the next two the minutes and the next four bytes represent the milliseconds. For example, 003901F4 in hex evaluates to 0hr:57min:0sec:500ms. Due to a firmware bug which causes falsified data, the current timestamp, as seen in Figure 4.10 is not valid. As a result, the timestamp values from the RFID-UHF readers in the database are automatically set to having a 'bad' status until it is fixed.

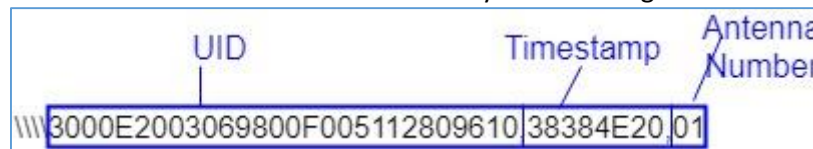


Figure 4.10: Raw data from RFID-UHF reader.

- **RFID-BLE readers.** As shown in Figure 4.11, the raw data from RFID-BLE reader is rather long and consists of many fields. Among the fields, it is important to point out the most important fields that are relevant to the applications in MSCPS are: Board ID (use to identify readers in a wide scale setup), IDD/MAC (ID of the BLE tag), Date and Time and RSSI (received signal strength).

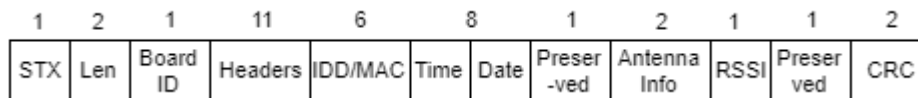


Figure 4.11: Raw data from RFID-BLE reader.

#### Mobile sensors

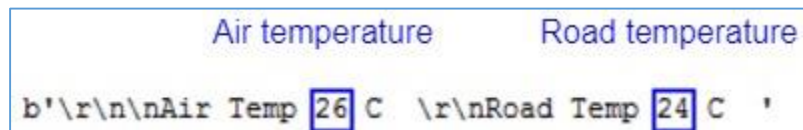


Figure 4.12: Raw data from temperature sensors.

- **Temperature Sensors.** Figure 4.12 illustrates the raw data from the temperature sensors including the ambient temperature and the road surface temperature data.

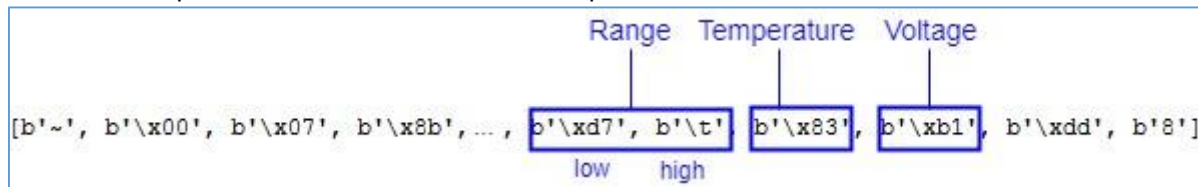


Figure 4.13: Raw data from Ultrasonic Level Sensors.

- **Ultrasonic Level Sensors.** The raw data sent from Ultrasonic sensors are encapsulated in a frame of 40 bytes. The raw data are 4 bytes from the sixth byte to the third byte from the end of the frame as illustrate in Figure 4.13. The sixth and fifth bytes from the end of the frame represents the range which are used to calculate the distance from the sensor to the salt surface. The actual distance in inches is calculated from the range (converted to decimal) and divided by 128, then subtracted by 28 (the sensor is placed 28 inches above the bottom of the salt trunk). The temperature reading in degree Celsius can be calculated from the value of the forth byte from the end (convert to decimal) by multiplying by 0.587085 and subtracted by 50. The voltage value is the third byte from the end, identifies the battery voltage and can be calculated by converting to decimal, subtracting by 14 and dividing by 40.

Time	Latitude	Longitude
\$GPGGA,200413.0,	4536.435400,N,	07335.222303,W,
\$GPRMB,05,2.9,-10.1,M,-33.0,M,*76		
\$GPRMB,259.9,T,259.9,M,0.0,N,0.0,K,A*23\r\n'		

Figure 4.14: Raw data from GPS sensors.

- **GPS sensors.** The raw data from GPS sensors is shown in Figure 4.14, including the timestamp in GMT, the latitude and the longitude.

#### 4.2.1.2.2 Unified data format

In this project, in order to unify the data formatting for the ease of access, the data format proposed by VdM was adopted. The collected data are stored in JSON files with the following values: *deviceID* and *Blocks*.

- **deviceID.** Each JSON file only stored data for a single device, identified by the deviceID value.the format for the deviceID is as the following:  
 $\langle \text{abbrev. of device type} \rangle \langle \text{last two digits} - \text{device address} \rangle p \langle \text{last two digits} - \text{device port} \rangle$   
 For example, radar data from address 192.168.10.51 and port 10002 can be found under deviceID "r51p02".
- **Blocks** value is a list of dictionaries. Each of those dictionaries corresponds to a specific value or a set of values. There is information about the format, the description of the values, the creation date-timestamp, the expiry date-timestamp, the unit and status of the value, and the value itself.
  - **Format** is ODNGF1 (Open Data Node Format Version 1) corresponds to the version of the unified data format.
  - **Desc** describes the data presented.
  - **CreateUtc** is the creation time and date of the data. The timestamp is created when the data is received at the management server.
  - **ExpiredUtc** is the expiration time of the data. Currently, "0000-00-00T00:00:00" is used, which indicated that the data is always valid.
  - **Unit** helps with the interpreting of the **Value**. If there are multiple values, the unit would be "array". If the multiple values are of different types, the unit would be "object".
  - **Status** is used to show if the data would be falsified. It could be "bad", "uncertain" or "good". The status is left empty for now as there is not yet a well-defined procedure for determining this.

#### 4.2.1.2.3 Database

The database to store collected data is deployed using MongoDB as it is open-source, scalable and is capable of distributed database implementation. The database is organized into four collections as the followings:

```

{
  "Format": "ODNF1"
  "Desc": "RFID tags data"
  "CreateUtc": "2017-05-17T10:58:23"
  "ExpireUtc": "0000-00-00T00:00:00" //data is valid at all times
  "Unit": "object"
  "Status": "good" //only good reports will be saved
  "Value": {
    "UID": ""
    "Timecode": ""
    "Antenna": ""
  }
}

```

**Figure 4.15:** Block structure of documents in RFID Collection.

**RadarRT Collection:** storing real-time reports from the traffic radars (Target detected and target count reports). The collection consists of JSON documents, each contains up to 1000 blocks for the ease of data access as well as management. The document name is defined as: *<name of the radar>+\_rt*, ex. *traf\_rad\_001\_rt*.

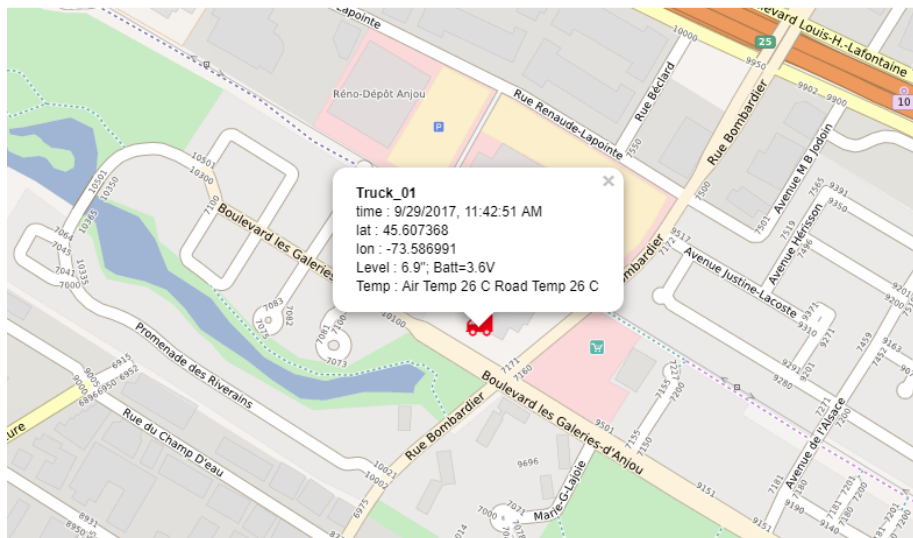
**RadarST Collection:** storing statistics reports from the traffic radars (Approaching and Receding Target statistics reports). The collection consists of JSON documents. The document name is defined as: *<name of the radar>+\_STA\_+<date>*, ex. *Traf\_Rad\_001\_STA\_2017-05-17*.

**Salt\_Truck Collection:** storing collected data from the salt truck. The collection consists of JSON documents, each contains up to 60 blocks for the ease of data access as well as management. The document name is defined as: *<name of the truck>+<date and time>*, ex. *anjou\_salt01\_2017-05-1710:58:23*.

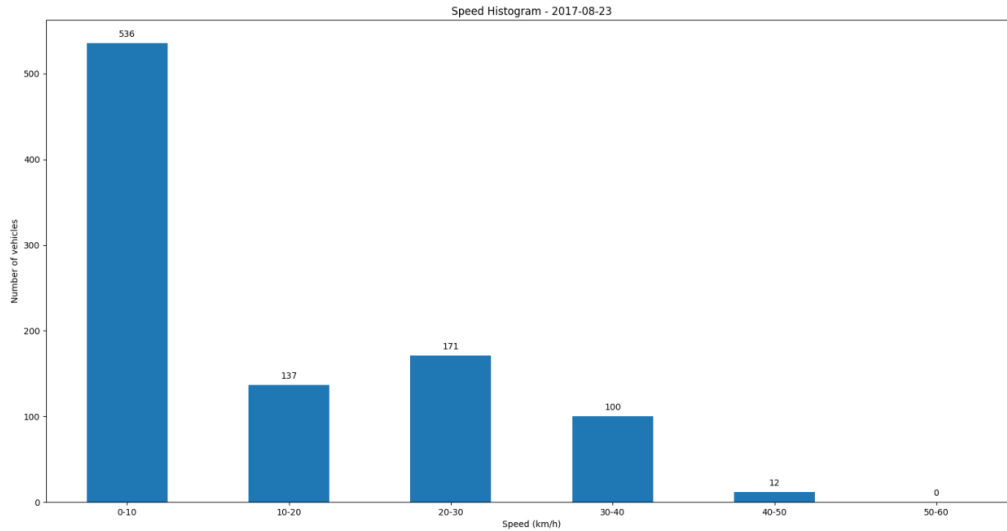
**RFID Collection:** storing collected data from the RFID readers. The collection consists of JSON documents. The document name is the name of the server, ex. *sch6\_1\_sv01*.

Figure 4.15 illustrates the block structure of the documents in the RFID collection. For illustrative examples of the block structure of other collections, please refer to Appendix E for further details.

#### 4.2.1.2.4 Data Visualization



**Figure 4.16:** Visualizing salt truck data.



**Figure 4.17:** Speed histogram visualization from radar sensor data.

For the salt truck, collected data can be visualizing in real-time by requesting the data directly from the Data Aggregation block and visualizing directly on Open Street Map. In this case, the location of the truck as well as the readings from its sensors can be displayed in real-time as illustrated in Figure 4.16.

Moreover, data can also be queried from the database by scripts and graphing using different tools. For example, Figure 4.17 presents the histogram of vehicle speeds from a radar sensor.

## 4.2.2 Control Plane

In this subsection, an overview of central management goals and deployed vendor central management tools will be detailed first. Next, the implementation specifics for on-premises setup prototype central management will be described.

### 4.2.2.1 Overview

The deployed physical network structure contains various types of devices, many of them produced by different manufacturers. Some of these devices, such as the Wi-Fi radios and IP cameras, have specially developed central management software. Some vendors provide these software platforms free of charge, while others require a subscription charge or product purchase.

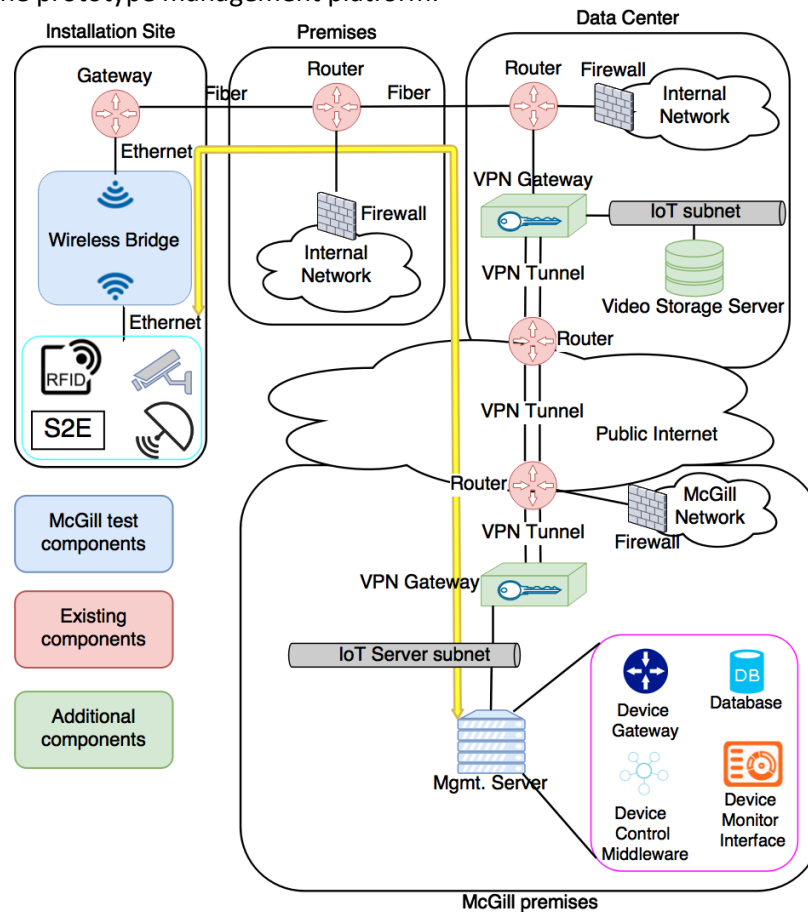
Vendor-made central management software from Ubiquiti, Axis, HikVision, and Panasonic were tried throughout the deployment. *Each software is only capable of managing devices from its own product line and type, and the features differ greatly from vendor to vendor.* Some software focuses more on an overview control to alert users of product failures or warnings, while others focus more on creating one console for viewing many device settings.

These observations motivate us to develop a prototype centralized management platform for the pilot deployment. The prototype platform is capable of monitoring all deployed devices from one location was developed after examination of commercial options. Throughout the design process, the use of standard protocols was emphasized to highlight inter-vendor capabilities. The central management platform prototype has a graphical user interface capable of remote rebooting and monitoring network status of devices.

As illustrated in Figure 4.18, the abstracted system is composed of *four fundamental components*: a device gateway, database, device control middleware and device monitor interface. The *device gateway* acts as a proxy to send control data from the user interface to the managed devices. The *database* stores the device information, including the authentication information to connect to each of the devices. *Device Control Middleware* mediates the requests from device monitor interface, translating these requests into appropriate commands to the devices. And the *device monitor interface* provides an easy-to use interface

for users to manage the devices. In the on-premises implementation, all of these components are deployed in a management server.

Due to the time limitation, only device availability status and device remote reboot features were implemented in the prototype management platform.



**Figure 4.18:** On premises device control management.

#### 4.2.2.2 Vendor platforms

This subsection will briefly present the commercial vendor software and features that are available for the deployed hardware, as well as the comments on each of the software.

##### 4.2.2.2.1 Ubiquiti Air Control 2

Ubiquiti AirControl2 is a free graphical user interface platform capable of monitoring Ubiquiti's Wi-Fi radios. It is possible to create automated alerts, configure select devices, and view relevant radio information such as online status, signal strength, throughput, as shown in Figure 4.19.

AirControl2 development does not appear to be a priority as it greatly lags behind their hardware developments. While AirControl2 can configure M5 series radios in batch directly within the console, it does not support this same feature for the full AC generation or the recently released AC2 generation. Note, a new beta was just released during this writing which provides management support to two types of AC devices.

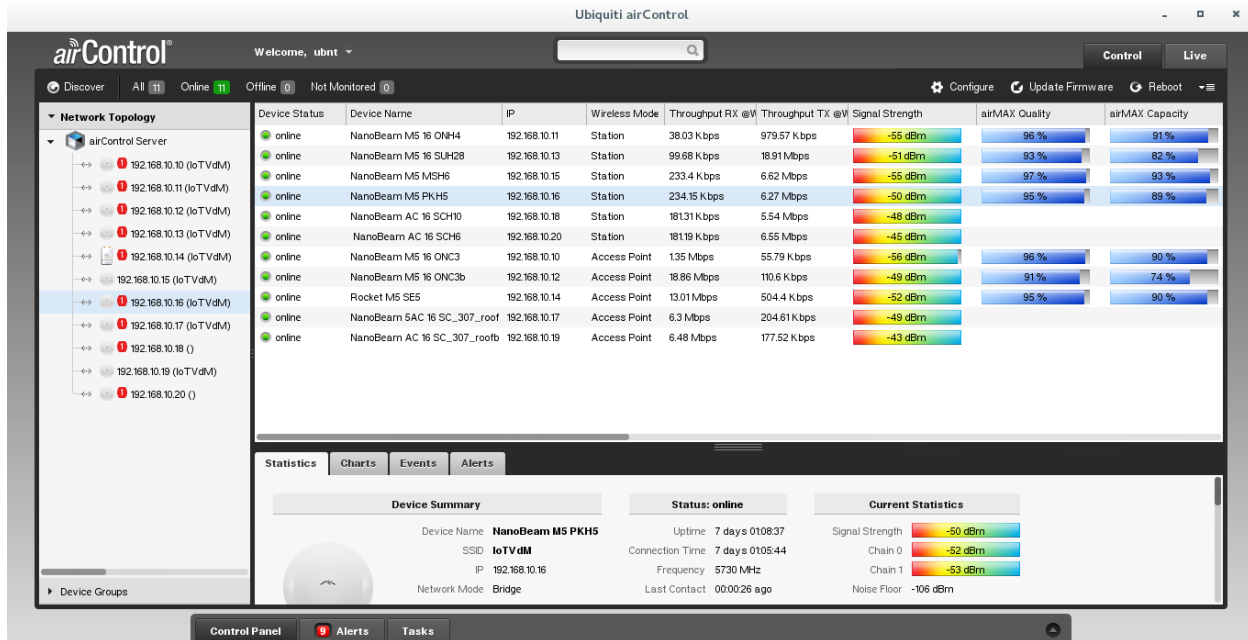


Figure 4.19: Ubiquiti AirControl2 GUI.

#### 4.2.2.2 Axis Camera Management Client

Axis Camera Management provides basic camera monitoring and management features. Through its graphical user interface software, it is possible to view camera online status, view and set alarms, and create configuration restore points as shown in Figure 4.20.

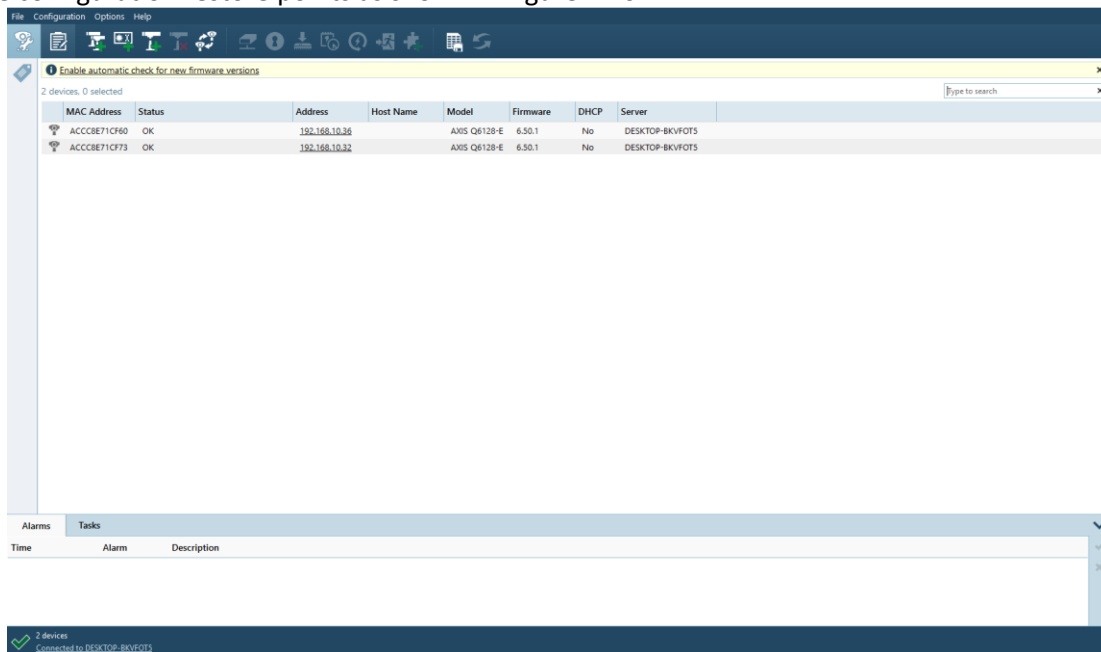


Figure 4.20: Axis Camera Management Client.

#### 4.2.2.2.3 HikVision iVMS-4200 Client

The iVMS-4200 software client is the central management tool developed for HikVision cameras. It has features that enable online status monitoring, camera configuration, alarm setup, and event management for programmed events as shown in Figure 4.21.



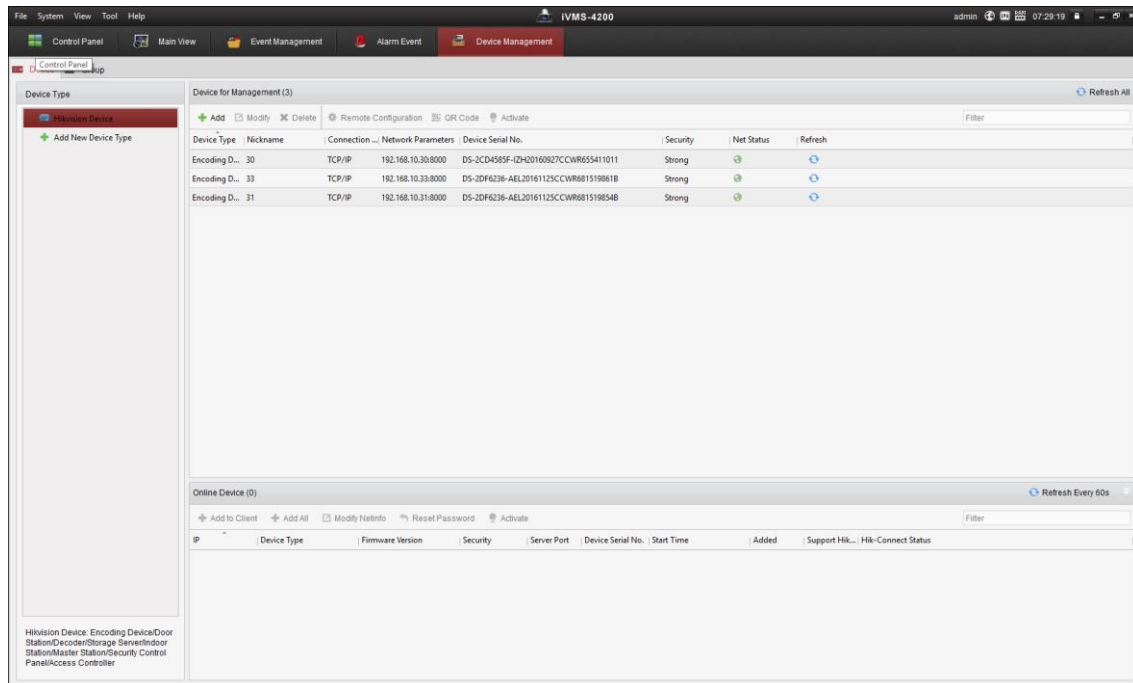


Figure 4.21: HikVision iVMS-4200 Client.

#### 4.2.2.4 Panasonic ASM200 Operation Software

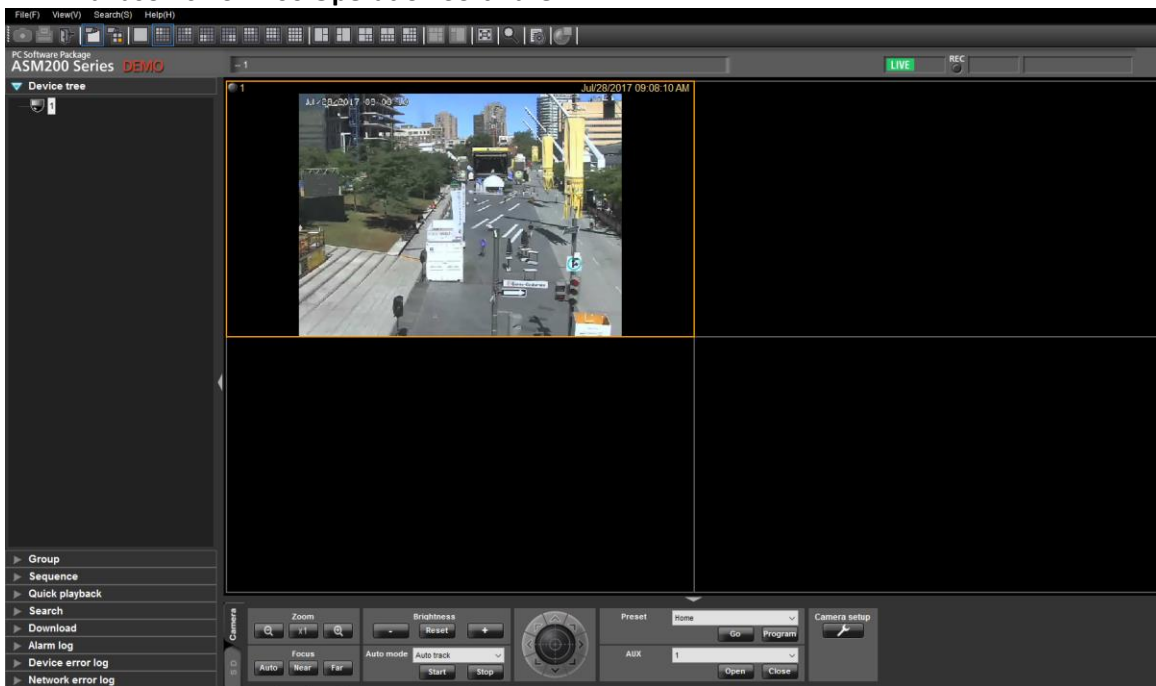


Figure 4.22: Panasonic ASM200 Operation Software.

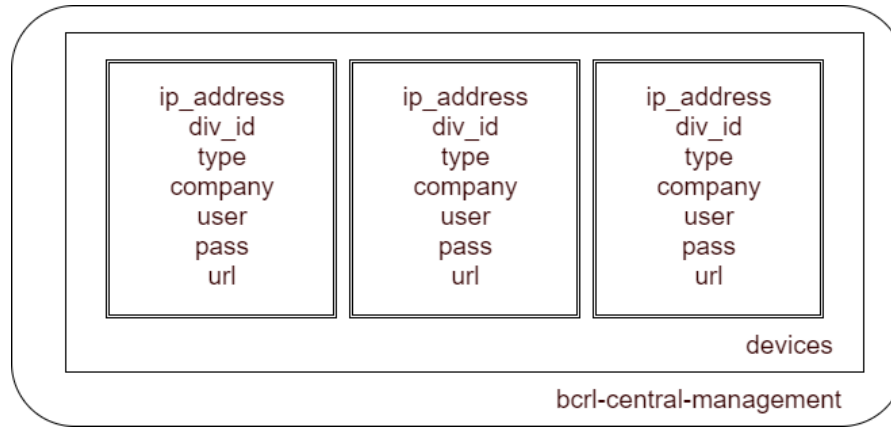
Panasonic offers a purchasable management tool for its cameras: ASM200 software. This software allows users to adjust camera settings in terms of viewing and recording parameters. Camera feeds can be viewed within the console. It also provides alarm, device, and network error logging as shown in Figure 4.22.

#### 4.2.2.3 Local Implementation Specifics



#### 4.2.2.3.1 Database

MongoDB was selected for information storage as it is open-source, scalable, capable of distributed database implementation and was a familiar system. For efficient information storage, the organization of the MongoDB database was set up as shown in Figure 4.23. A cluster called '*bcrl-central-management*' was created. Inside this cluster, a collection named '*devices*' was generated. This collection was designed to store the device information of each device on the network.



**Figure 4.23:** Device information fields and relation implemented in the database.

For each device in the '*devices*' collection, there are seven data fields: *ip\_address*, *div\_id*, *type*, *company*, *user*, *pass*, *url*. The field *ip\_address* is used to hold the ip address at which a device can be found. *Div\_id* also stores the *ip\_address* except for with each dot in the *ip\_string* replaced by an underscore for ease in environments such as JavaScript where a dot signifies a special character. Device descriptions such as S2E for serial to Ethernet devices, or *ip\_camera* for cameras are set within the *type* field. *Company*, *user*, and *pass* denote the manufacturer and the authentication credentials for the device. The *url* field is preserved for optional integration to a cloud integration system, in which case, the *url* field stores the device's Azure connection string.

#### 4.2.2.3.2 Device Control Middleware

The Device Control Middleware connects the Device Monitor Interface to the Device Gateway. In the on-premises setup, both the Device Monitor Interface and Device Gateway run on the same management server, hence, direct connection is used.

#### 4.2.2.3.3 Device Gateway, and Device Monitor Interface

In the on-premises implementation, since the prototype management platform was implemented on a single machine, the functionalities of Device Gateway, and Device Monitor Interface were merged into a single program to simplify and shorten the implementation. The implementations of the two features of the prototype management platform (device status monitoring and remote reboot) vary from device to device and will be described in the following text.

##### 4.2.2.3.3.1 Device Status Monitoring

Monitoring of device online status is accomplished through sending ICMP ping packets from the management server to the monitored IP device. The ping test is scheduled and performed every 20 seconds to maintain an active status log. This test is performed on all IP-enabled devices including all the Wi-Fi radios, IP cameras, and Serial-to-Ethernet adapters. For non-IP devices such as sensors, status monitoring using ICMP is not possible.

Two log files are kept, one for offline devices due to rebooting commands and one for offline devices due to other reasons. Each time an online status check (ping test) is performed, if a device does not return a ping response, a new line entry with a time stamp, and IP address (separated by commas) will be added to the rebooting or offline log file, depending on whether a reboot command was recently sent or not.

The logs are formatted into comma-separated-value (csv) files. This allows a user to import the data into an excel spreadsheet to manipulate data or parse based on values separated by commas and new lines.

```
2017:07:20:09:34:55,192.168.10.34
2017:07:20:09:40:16,192.168.10.16
2017:07:20:09:40:35,192.168.10.17
2017:07:20:09:40:35,192.168.10.18
```

**Figure 4.24:** Entries from csv offline log file.

#### 4.2.2.3.3.2 Remote Reboot of stationary devices

##### 4.2.2.3.3.2.1 WiFi Radios

The Ubiquiti Wi-Fi radios deployed in the physical network have Telnet and SSH capabilities. Due to SSH's encrypted communication method, SSH was used to manage radios. The radios' SSH client can perform reboots, speed tests, return hardware statistics and information; however, only the reboot features is implemented in the prototype centralized management platform.

When the management server receives a reboot command, it creates an SSH connection to the radio using the username and password taken from the MongoDB database.

##### 4.2.2.3.3.2.2 IP Cameras

All the IP cameras deployed in the pilot deployment have ONVIF protocol capabilities. ONVIF is a standard which was developed to maintain same management and camera control commands across any compatible camera. The use of ONVIF allows development of one method to perform a function, rather than many using vendor-specific APIs. The reboot ONVIF command as shown in Figure 4.25 is sent to the cameras via an https connection created through the Node.js npm 'request' library.

Request device.SystemReboot
<pre>&lt;?xml version='1.0' encoding='utf-8'?&gt; &lt;soapenv:Envelope   xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"   xmlns:tds="http://www.onvif.org/ver10/device/wsdl"&gt;   &lt;soapenv:Body&gt;     &lt;tds:SystemReboot/&gt;   &lt;/soapenv:Body&gt; &lt;/soapenv:Envelope&gt;</pre>

**Figure 4.25:** ONVIF command to perform reboot.

Cameras authenticate the username and password credentials before receiving the command. Most vendors use basic authentication, but Axis uses digest authentication. To make sure the authentication passes, the proper authentication technique must be performed.

By using the ONVIF standard, development time for implementing camera functions into a central management platform can be greatly reduced without having to worry about any specific manufacturer APIs or function calls. However, while developing software that ran ONVIF commands on cameras from different vendors, there were sometimes issues relating to the underlying software on the cameras themselves. *Cameras from different manufacturers had different parsers.* This sometimes resulted in an ONVIF command being properly parsed and executed by one camera, but resulted in an error for another. Thus, before software is rolled into deployment, it is *imperative to test* that the command can be executed and read by all camera vendors.

##### 4.2.2.3.3.2.3 Adapters

In the physical network, adapters are used to transform signals between communication protocols, such as from Ethernet to Serial. Some devices, such as traffic radars, are not connected via an IP addressing scheme, and are accessed via socket ports on an adapter. This includes traffic radars and RFID readers. Unfortunately, there does not exist a standardized command scheme to manage or control adapter devices. Vendors offer different methods of restarting their devices. USR IOT devices can be restarted via their webserver or UDP broadcast, Perle and Lantronix devices can be restarted via webserver, SSH, or other options.

Reboot of adapters was more difficult to implement than for other devices, and functioning integration to the management platform was not successful.

#### **4.2.2.3.2.4 UHF RFID**

To restart the RFID reader, the reset [0x64] command must be dispatched to the reader. This is accomplished through the use of the FEIG reader's C++ SDK. This reset ability was successfully tested with the SDK, but there was not enough time to integrate the functionality into the management platform interface.

#### **4.2.2.3.2.5 Radars**

The deployed radars do not have a command option to restart. Through discussion from the manufacturer, this firmware capability could be implemented, but would require the manufacturer to develop it.

#### **4.2.2.3.3 Remote Reboot of Mobile devices**

The remote restart procedures for mobile devices were investigated but were not implemented due to the lack of time.

##### **4.2.2.3.3.1 LTE Gateway**

The Sierra Wireless Airlink GX450 has control and configuration methods through its webserver, telnet, and SSH. Telnet and SSH management of this device is performed through sending AT commands. To perform a reboot, the 'Z' command is used.

##### **4.2.2.3.3.2 Temperature Sensor**

The temperature sensor and its adapter function as a single unit and are considered together for management purposes. To perform a restart of this temperature unit, the AT command '\*' must be sent. As the adapter is connected to a Serial-to-Ethernet adapter device, the '\*' AT command is sent to the appropriate port on the adapter.

##### **4.2.2.3.3.3 Level Sensor**

The level sensor is more difficult to configure than the other devices due to hardware constraints. For configuration, the manufacturer mandates a Digi Gateway to be used alongside the Massa M3 level sensor and use of their software. Reverse-engineering a solution may be possible for a central control interface, but would be time consuming and was not explored.

#### **4.2.2.3.3.4 Graphical User Interface**

A graphical user interface is designed for quick viewing of device statuses and remote rebooting of managed devices. The user interface is composed of three main columns, displaying each of the online, rebooting, and offline devices. A search bar was implemented for sorting capabilities. The interface is illustrated in Figure 4.26.

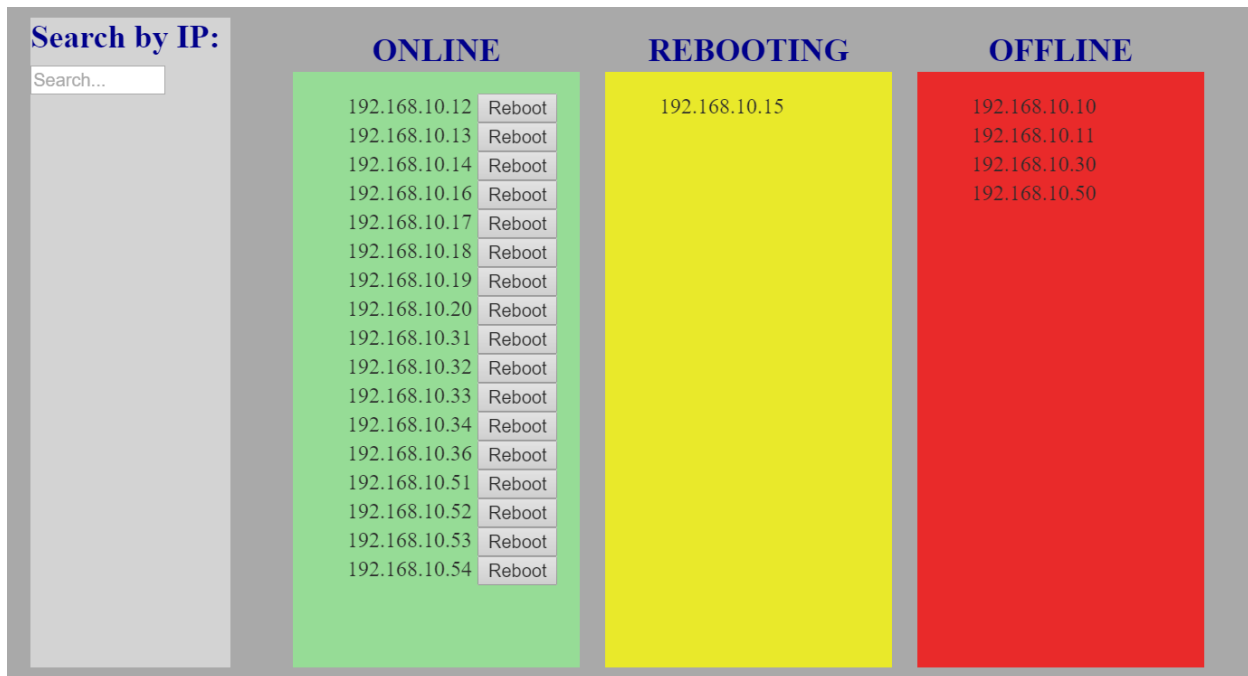


Figure 4.26: Central Management GUI.

### 4.3 Cloud-based data management structure for the MSCPS

#### 4.3.1 Introduction to Microsoft Azure

In this project, Microsoft Azure is used as an example cloud computing platform to illustrate the integration of cloud computing architecture and IoT network of MSCPS. Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through a global network of Microsoft-managed data centers. It provides all three IaaS, PaaS, and SaaS service models and supports many different programming languages, tools, and frameworks to collaborate with both Microsoft and third-party software and systems.

##### 4.3.1.1 IaaS service - Azure Resource Management and Provisioning

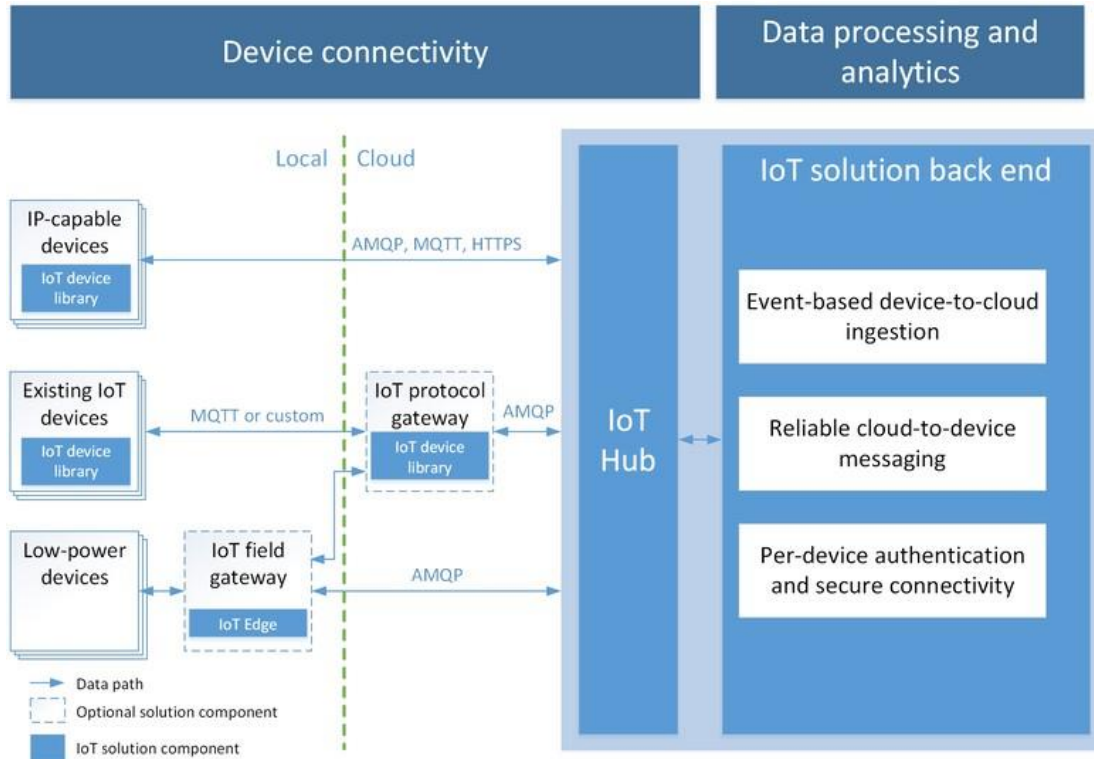
The basic components of the IaaS layer are the pool of virtual resources or assets provided by some of Microsoft service providers in the hardware layer: Microsoft.Compute supplies virtual machine resources; Microsoft.Storage provides storage account resource; and Microsoft.Web handles resources related to web applications. Various resources from these providers would be used to setup virtual machines, storage databases, and networking functionalities to develop the infrastructure for management and processing of IoT devices and data. Azure Resource Manager allows related assets to be grouped into the same resource group to be managed and monitored together.

##### 4.3.1.2 PaaS Layer - Azure IoT Hub and Media Services

###### Azure IoT Hub

To support IoT integration, Microsoft Azure presents IoT Hub, a fully managed service that facilitates IoT device connectivity with reliable and secure bidirectional communication to Azure back-end management services. Essentially, IoT Hub acts as a cloud middleware gateway that gathers incoming data and makes it available for further processing by other services in IoT solution as well as delivering messages to IoT devices as shown in Figure 4.27. IoT IP-capable devices can directly connect to IoT Hub using supported

application protocols like HTTP, MQTT, and AMQP. Those devices that do not support these protocols can communicate via gateways for protocol adaptations. For devices that cannot access Internet directly (i.e., devices that use industry-specific protocols on internal networks), a field gateway—usually some specialized equipment or a low-power computer that runs appropriate software—can be deployed at the edge remote site to facilitates communication using protocols supported by the devices.

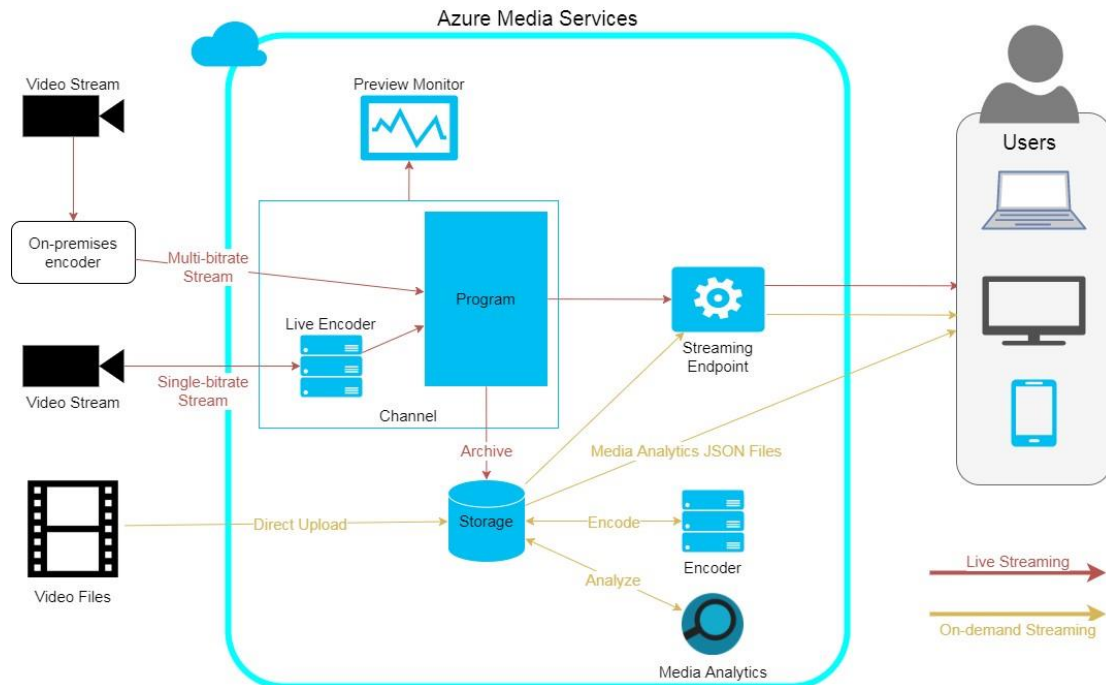


**Figure 4.27:** Overview of device connectivity and dataflow via IoT Hub.

In addition, IoT Hub also provides a number of services to monitor and manage IoT devices. First, Microsoft claims that millions of simultaneously connected devices can be supported as IoT Hub automatically scales up though the scaling rates vary by service tiers. Moreover, each device is registered and monitored individually to identify device connectivity issues (e.g., incorrect credentials, rejected messages) through detailed operation logs. This is achieved by the use of *device twins*, JSON documents that store state information, such as device-specific metadata, configurations, and conditions, of each device. Essentially, device twin acts a virtual representation of the corresponding IoT device, reporting device progress and status to back-end management system.

#### Azure Media Services

Since IoT Hub is mostly designed to handle message-based communication with sensors, which is unsuitable for data streaming from multimedia devices such as video cameras. Multimedia services is instead provided by Azure Media Services (AMS), an extensible cloud-based platform that enables developers to build scalable media management and delivery applications. AMS enables content providers to securely upload, store, encode, and deliver video and audio content in both on-demand and live streaming scenarios to various user devices (e.g., TV, PC, smart phones, tablets, etc.).



**Figure 4.28:** Overview of video streaming data flow via Azure Media Services.

As shown in Figure 4.28, multimedia content can be uploaded to AMS through two data paths: *on-demand streaming* and *live streaming*. In On-demand streaming approach, video files are directly uploaded to the cloud storage through AMS web portal or REST APIs, which can then be encoded to different formats and bitrates as required by content provider. On the other hand, live streaming allows video content to be delivered live to users through AMS in a streaming program. Each streaming program is contained in a *channel*, a pipeline for processing live-streaming content before delivering to users. Video data can be encoded using on-premises encoding software such as FFmpeg or cloud encoder provided by AMS. If encoded using on-premises software, the incoming stream passes through AMS to the stream endpoint and is distributed to users without any encoding. Cloud encoder allows a single-bitrate incoming stream to be encoded to different bitrates and formats to suit user requirements. Live streaming content is archived after program is finished, allowing users to access streamed data through on-demand streaming at a later time.

However, there are limitations to AMS when applying to Smart City applications such as video surveillance. Each streaming program *can only run continuously for up to 25 hours*, which means it is difficult to provide continuous video monitoring through AMS live streaming services. On-demand streaming is only compatible with complete video files, thus continuous video streams have to be processed elsewhere into individual files before uploading to the cloud. Live streaming are limited by AMS quotas per Azure subscription account, which reduces the scalability of this Azure platform. *Only 5 channels can be live simultaneously* and *only 1 input video stream per channel*, which *effectively limits support to 5 incoming streams at any given time*. At its current development stage, AMS is not yet suitable for large-scale scenarios with many input video streams such as Smart City.

#### 4.3.1.3 SaaS Layer - Azure Data and Media Analytics Services

##### Azure Stream Analytics

In Azure IoT platform, data processing for both storage and real-time analysis is provided by Azure Stream Analytics (ASA), a real-time event processing service that provides analytical capabilities on streaming



data from multiple sources, including devices, applications, and sensors. In the context of IoT, ASA is scalable, being capable of handling high data throughput of up to 1GB/second by partitioning the message stream to be processed concurrently by multiple readers. Additionally, ASA provides reliability with built-in recovery capabilities to maintain internal states, preventing data loss and maintaining business continuity in the event of failures. Finally, ASA is well integrated with many other Azure services, retrieving from IoT Hub for stream data and Azure storage for historical data. Analytical results produced by ASA can then be written to Azure storage, visualized for business intelligence, or further processed by other big data processing services.

### **Azure Media Analytics**

Similarly, media content uploaded to AMS can be analysed using Azure Media Analytics (AMA), a collection of speech and vision services that enables derivation of useful information from video files, including *face detection*, *motion detection*, and *optical character recognition*. *Face detection* can be used to aggregate and analyze reactions of people attending an event by detecting people's faces and their emotions such as happiness, sadness, and surprise. *Motion detection* identifies motion against stationary backgrounds in a video, recording timestamps and pixel locations of where the event occurs. *Optical character recognition* converts text content in video files into editable, searchable digital text, allowing easy indexing and searching for contents. The results of AMA services are generated in a JSON file, which contains metadata such as timestamps, pixel regions where objects of interest are detected, and confidence value of the detection. AMA is still in preview beta with and its analytical results still require further processing to be useful for visualizations and Smart City applications. Furthermore, the current AMA only works with complete video files stored in AMS storage, which is *not suitable for real-time applications*.

## **4.3.2 Data Plane**

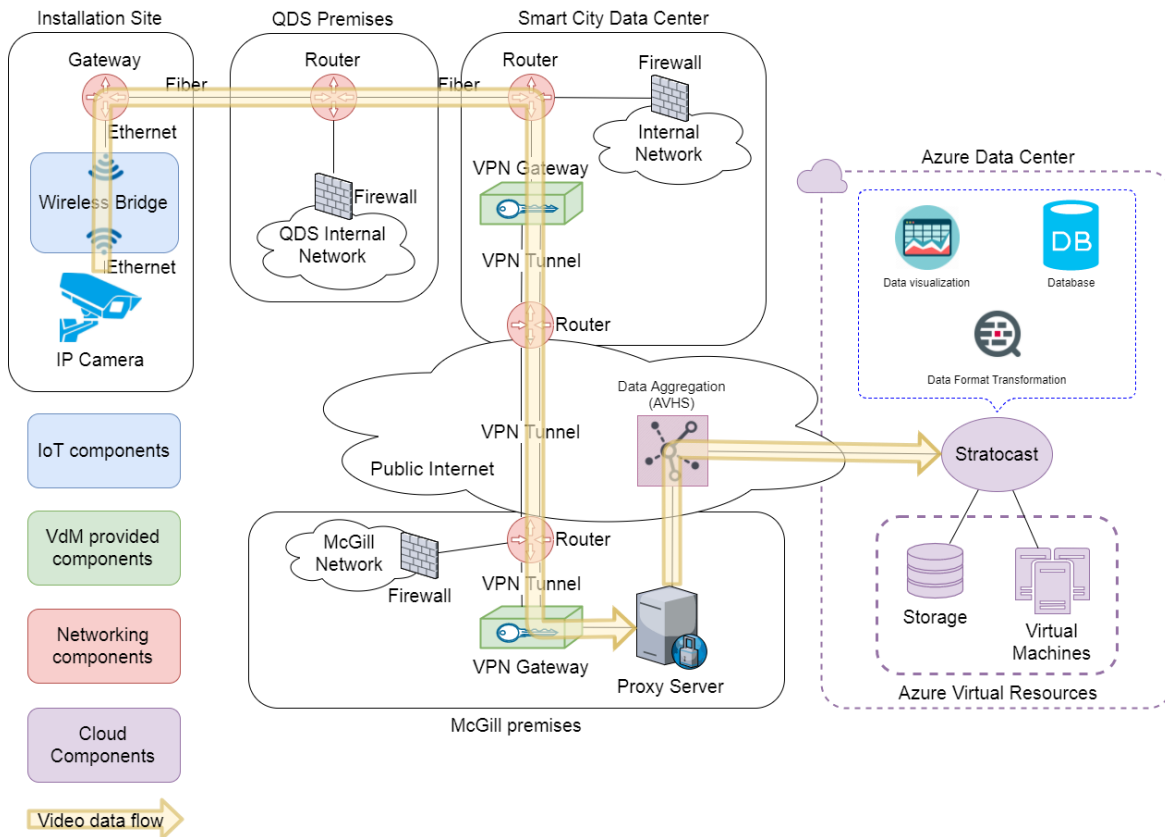
### **4.3.2.1 Camera**

#### **4.3.2.1.1 Data flow structure**

As discussed in section 4.2.1, with the current functionalities, Microsoft Azure does not have a compatible built-in platform for developing a cloud-based video surveillance application for Smart City. To provide an adequate comparison to the on-premises infrastructure setup, we investigate Stratocast<sup>9</sup>, an illustrative VMS application developed by Genetec, based on Azure. In line with the cloud computing paradigm, Stratocast enables VMS capabilities by eliminating the need for on-premises servers and storage, reducing traditional hardware costs and installation time. To achieve this, Stratocast utilizes Microsoft Azure cloud IaaS to support their proprietary video management and monitoring platform. Live video streams are uploaded directly to the cloud and stored in secure Azure distributed data centers, taking advantage of Azure reliable and scalable resources to ensure data protection. Being on the cloud also allows Stratocast to provide a centralized management solution accessible anywhere and anytime through the Internet from laptops, smart phones, or tablets. Stratocast services are offered in 3 separate packages, each with different video qualities (e.g., resolution, FPS) based on demand of clients. Additionally, Stratocast also offers edge recording where video recordings are stored directly on local storage devices such as NAS or SD cards, to reduce bandwidth consumption.

---

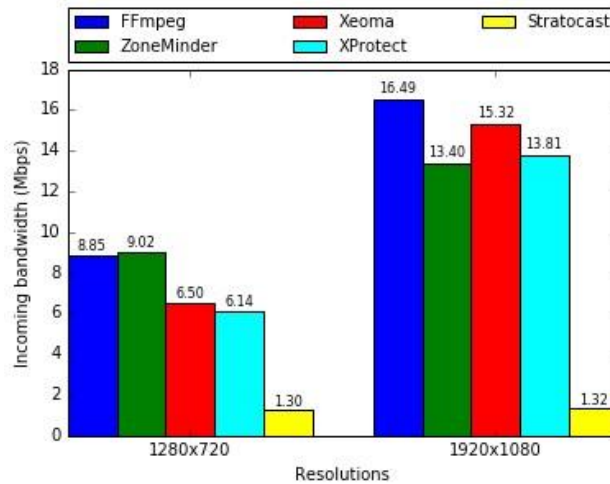
<sup>9</sup> G. Inc. *Stratocast*. Accessed 2017. url: <https://www.stratocast.com/>



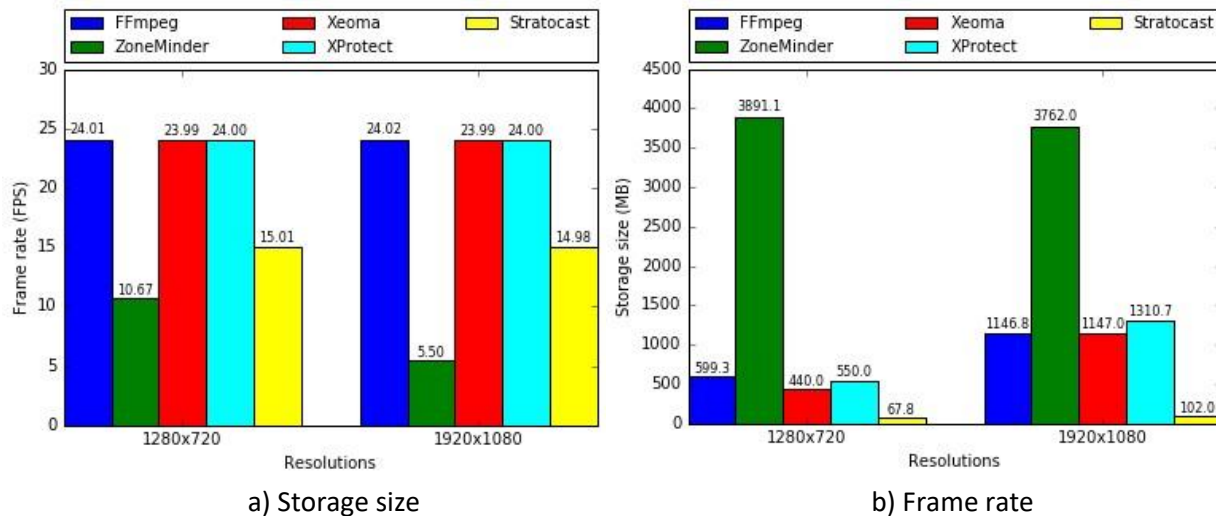
**Figure 4.29:** Video Data flow with Stratocast.

Figure 4.29 describes the Video data flow from the current pilot setup to Stratocast. Stratocast communicates with IP-cameras using cloud-based video surveillance middleware services developed by IP-camera manufacturers like Axis and Vivotek to register, monitor, and manage IP-cameras. Currently, Stratocast only supports certain camera models from Axis and Vivotek through their propriety middleware services namely Axis Video Hosting System (AVHS) and Vivotek Application Development Platform (VADP). In the experimental setup, there are 2 Axis Q6128-E IP-cameras that are compatible with AVHS middle ware service. Registration and communication between the IP-cameras and AVHS is done by a built-in function integrated in the camera firmware, which transmits an HTTP or HTTPS request to AVHS server to establish a TCP connection. However, as the cameras are on a private network for security reason, in order to enable communication with AVHS, a proxy is set up on McGill premises to relay outbound traffic from the private network to the public Internet, allowing the cameras to send requests and data to AVHS server. As communication is over the public Internet without VPN protection, only one IP-camera is registered to AVHS and Stratocast to minimize security risk to the rest of the system. In this case, *the AVHS acts as the data aggregation and Stratocast combines the data format translation, database and visualization functionalities.*

#### 4.3.2.1.2 Stratocast Performance Comparison



**Figure 4.30:** Bandwidth usage of Stratocast in comparison to on-premises solutions.

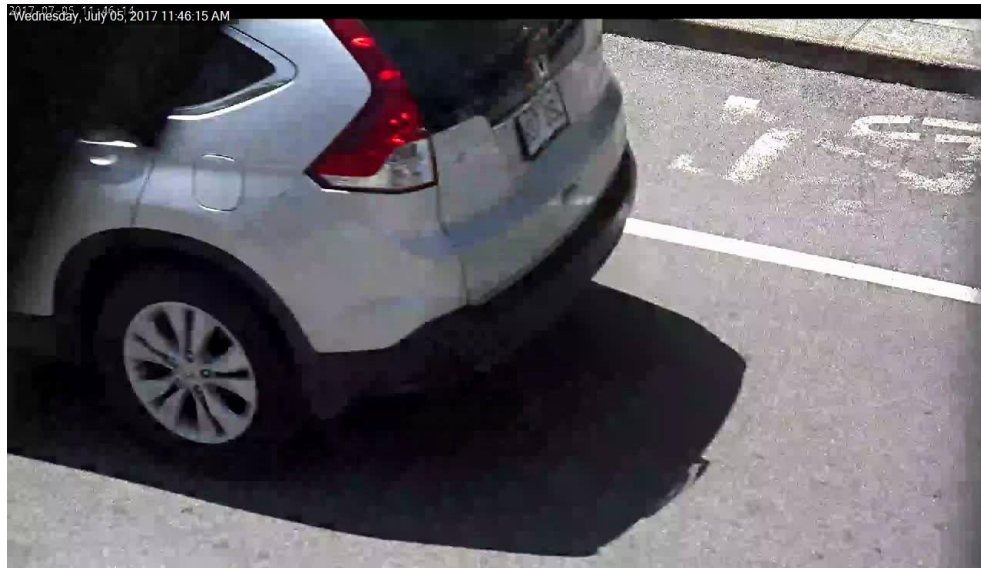


**Figure 4.31:** Frame rate and storage size of Stratocast in comparison to on-premises solutions.

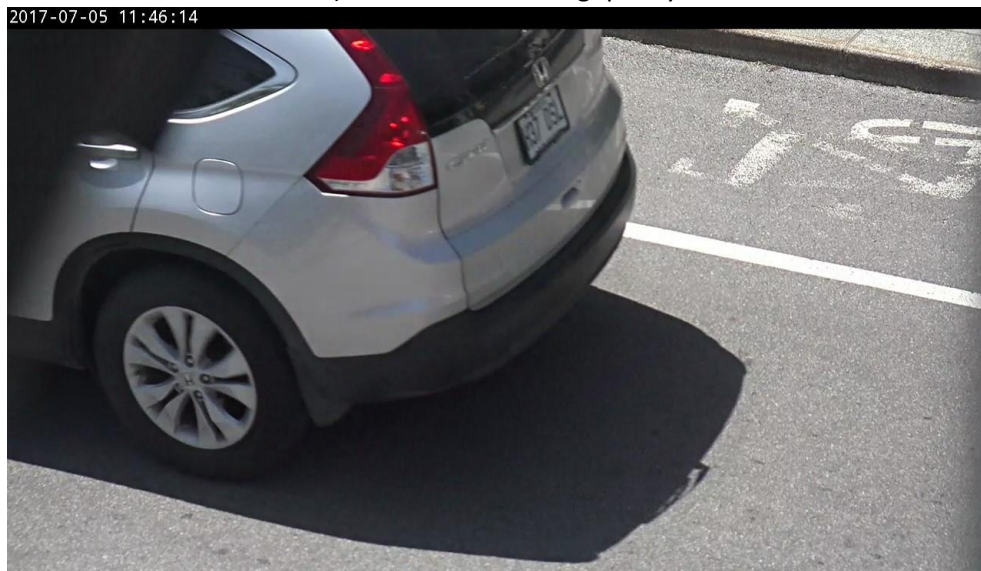
As only one IP-camera is connected to Stratocast, single stream performance of Stratocast is examined and compared it with the on-premises setup in the previous section. As Stratocast is hosted on cloud, hardware processing resources such as CPU and RAM usage are no longer a concern for the management application. Instead, the *bandwidth usage* to stream video data, *video recording size*, and *FPS* are investigated. As the highest resolution supported by Stratocast is  $1920 \times 1080$ , only results at two different resolutions:  $1280 \times 720$  and  $1920 \times 1080$  will be compared. The recording frame rate and bit rate of the test camera are set at 24 FPS and 16 Mbps, respectively (the same as in the previous section). Video recording frame rate and bit rate configurations on Stratocast are set at the maximum supported by the premium plan (highest package) at 15 FPS and 1.2 Mbps bit rate, respectively.

Figure 4.30 shows the network bandwidth usage of Stratocast in comparison with the on-premises VMS solutions. It is observed that while bandwidth usage of other solutions increase as resolution increases due to more data to be transmitted, bandwidth usage of Stratocast is throttled at around 1.30Mbps. This constraint is intended to restrict the bandwidth consumption of camera streams so that live view is possible. As a result, data has to be further compressed to meet the bandwidth requirement by Stratocast, leading in poorer video quality, as illustrated in Figure 4.32a.

Bandwidth is not the only constraint that Stratocast imposes on video traffic. Frame rate of video stream to Stratocast is also limited to 15 FPS for the premium plan, as depicted in Figure 4.31a. Unlike ZoneMinder, whose frame rate is inhibited by the use of poorly optimized implementation of MJPEG and the processing capability of its hosting server, Stratocast frame rate is restricted by its service provider. The reduced bandwidth consumption, combined with a lower frame rate, results in lower storage requirement for Stratocast recordings. As shown in Figure 4.31b, the storage size of a 10-minute recording for Stratocast is only 67.8 MB at 1280 × 720 resolution and 102.0 MB at 1920 × 1080, significantly lower than the recording sizes for other on-premises VMS solutions at the same resolution.



a) Stratocast recording quality



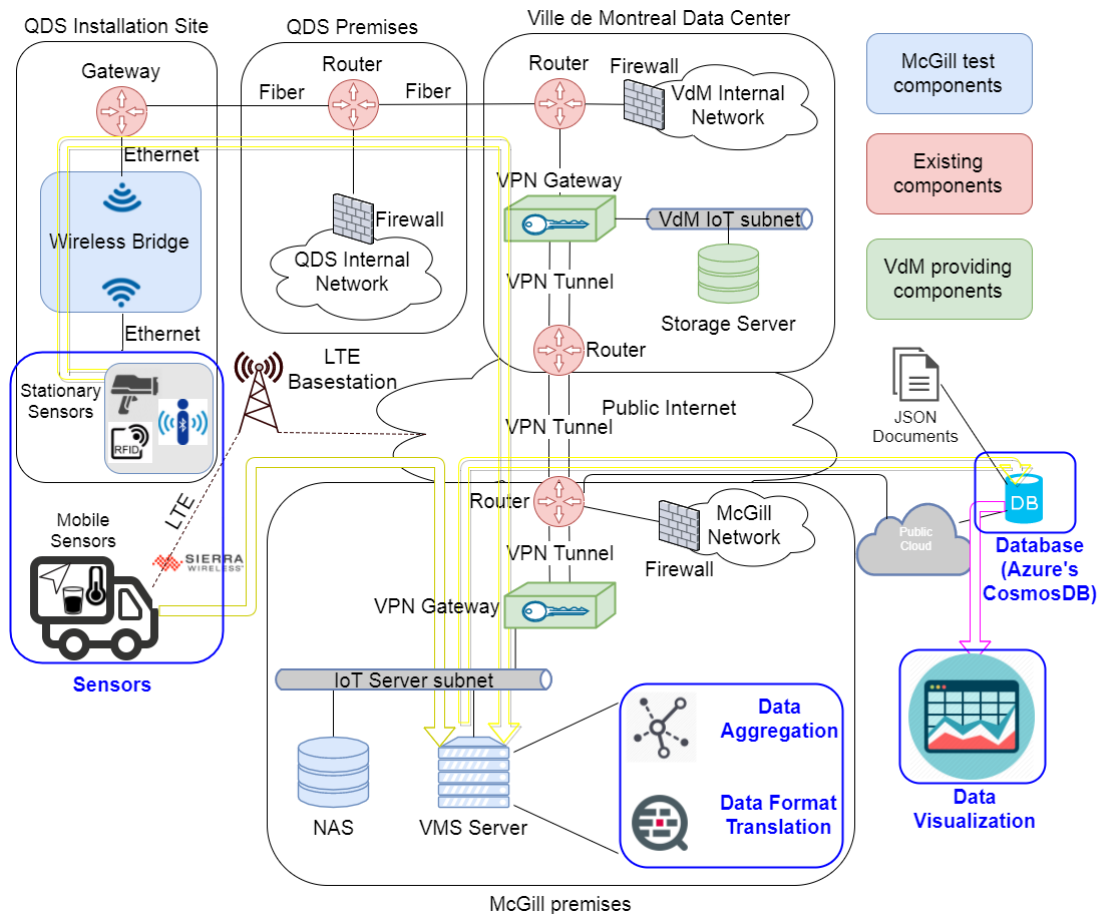
b) FFmpeg recording quality

**Figure 4.32:** Video recording quality comparison of Stratocast and FFmpeg.

#### 4.3.2.2 Other types of sensors

In this subsection, an overview of a cloud-based centralized data management and its implementation specifics will be described.

#### 4.3.2.2.1 Overview



**Figure 4.33:** Cloud-based data management.

For cloud-based data management, as illustrated in Figure 4.33 the Database is moved to the Cloud (using Azure CosmosDB) so that Data Visualization can be done from any device with Internet connection. The Data Aggregation and Data Formatting Translation are retained on-premises. It is noted that these two functional blocks can be moved to the cloud; however, Data Aggregation is built on-premises due to security reasons (prevent direct access from the Internet to the sensors) and Data Formatting Translation functional blocks, which can be implemented in Azure's IoT hub, is implemented on-premises for the ease of implementation and because of the time limitation.

#### 4.3.2.2.2 Data Aggregation and Data Format Translation

These two functional blocks are retained on-premises and hence, the implementation details are the same as the on-premises implementation.

#### 4.3.2.2.3 Database

In the cloud setup, Azure CosmosDB is used. In order to store data to the database, a url and a key (simple string) are required and can be accessed from the 'keys' page in Azure. In the current setup, the database has unlimited storage and can store multiple JSON documents.

The Database cost is based on a provisioned amount of Request Units (RUs) and the amount of storage used. Storage costs \$0.33 CAD per GB per month (\$167.19 per month for 500GB)<sup>10</sup>. The storage costs work as a pay-as-you-go system. RUs are Azure's custom unit for CPU, memory and IOPS (input/output operations per second) usage. The current database has 2500 RUs/sec provisioned (lowest amount for a

<sup>10</sup> Azure Cosmos DB pricing: <https://azure.microsoft.com/en-gb/pricing/details/cosmos-db/>



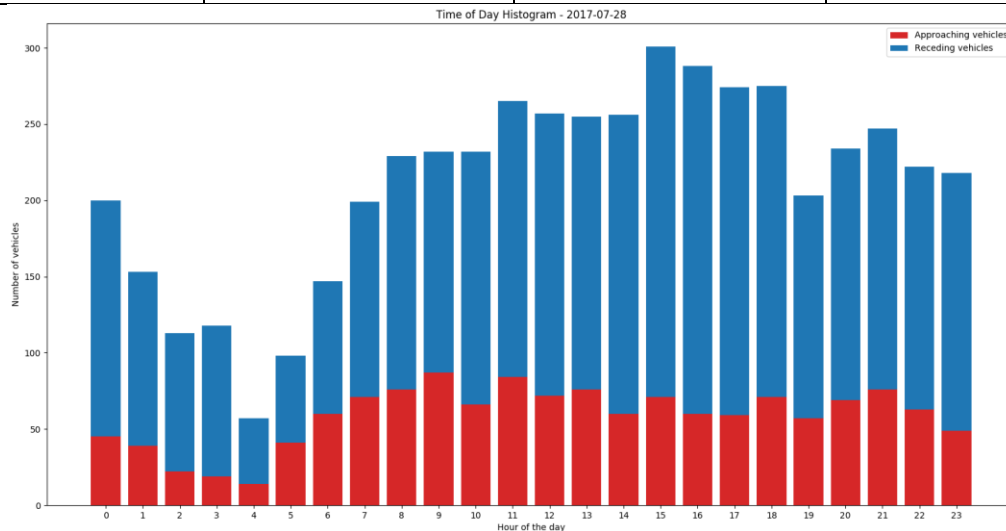
collection with unlimited storage) which costs \$199.02 CAD per month. For the amount of radars that are currently live, the above provisioned amount of RUs/sec is more than enough. Azure's CosmosDB has an upload speed of 1.51 MB/s from the East of Canada<sup>11</sup>.

**Table 4.2:** Amount of data upload from different types of sensors.

Sensors	Consumption (kB/hour)	Incoming data rate	Writing to database rate
Radars	~20,000 (approximation)	20 sentences per second	Once every ~3 min
Temperature	1	Once every 15 min	Once every 15 min
Level	1	Once every 15 min	Once every 15 min
GPS	2	Once every 15 min	Once every 15 min

**Table 4.3:** Data retention prediction for all currently live sensors

Sensors	1 Day	1 Month	1 Year
8 radars	3.840 GB	115.2 GB	1.4016 TB
1 Temperature sensor	24 KB	720 KB	8.760 MB
1 Level sensor	24 KB	720 KB	8.760 MB
1 GPS sensor	48 KB	1.440 MB	17.520 MB



**Figure 4.34:** Number of approaching and receding vehicle at different time in a day.

Since data storage is costly in Cloud-based setups, the data consumptions of different type of sensors are monitored. As illustrated in Table 4.2, traffic sensors have the most data recorded with approximately 1MB file being uploaded every 3 minutes. For data retention, Table 4.3 shows the required storage for all of the currently live sensors for 1 day, 1 month and 1 year.

#### 4.3.2.2.4 Data Visualization

In order to obtain the timeout and connection information of traffic radars, queries can be used directly on CosmosDB dashboard.

For visualizing data to charts, a user-interactive program was implemented for traffic radar, temperature and ultrasonic level sensors. For Traffic radars two types of graphs can be created: A time-of-day histogram which shows the number of cars (receding and approaching) passing by in the different hours

<sup>11</sup> Azure Speed test – Available online: <http://www.azure-speed.com/>

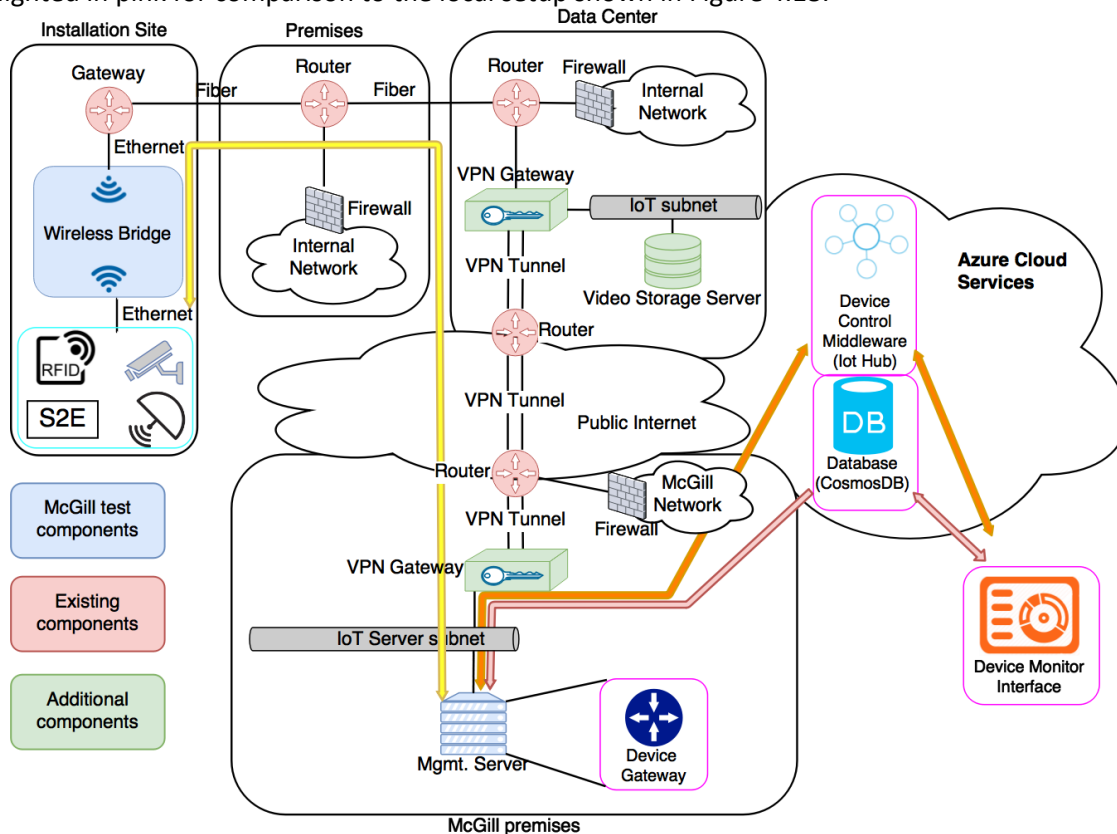


of the day as shown in Figure 4.34, and a speed histogram which shows the number of cars in different speed ranges. For temperature and ultrasonic level sensors, graphs that show the changes in the temperature or level value over the course of the day can be produced. All graphs are saved as .png files.

### 4.3.3 Control Plane

This section will focus on describing specifically how various tools were used and implemented for the cloud-based implementation. Specific device command implementations, the graphical user interface, and other aspects remain the same as in the local implementation and will not be discussed.

As shown in Figure 4.35, *three major changes* occur in the transition from the local setup to the cloud-based implementation. The *database* and *user connection manager* functional blocks are moved from within the on-premises management server to the cloud. This also allows the *user interface* to move from the single management server to any computer with internet access. These changes provide management capability from outside the IoT network and limitless scalability without physical hardware changes. A figure explanation of the cloud setup can be seen in Figure 4.35, with the five functional parts once again highlighted in pink for comparison to the local setup shown in Figure 4.18.



**Figure 4.35:** Cloud-based device control management.

#### 4.3.3.1 Database

The storage structure for the cloud implementation is kept the same as in on-premises setup. However, the MongoDB database in the on-premises setup was moved to the cloud and deployed using Azure CosmosDB. This can be done by creating a CosmosDB database with MongoDB as the database structure and then creating the same 'devices' collection and populating it with all device info.

#### 4.3.3.2 Device Control Middleware

In the cloud setup, the User Connection Manager is realized by the Azure IoT Hub, which will receive commands from the User Interface and forward to device gateway. The MQTT protocol is used for messaging to and from the IoT Hub, with symmetric keys being used as the authentication scheme.

#### 4.3.3.3 Device Gateway

Due to the physical network structure design and security constraints, devices are hidden inside a private network and most of them do not have direct access to the public cloud. This physical design leads to data and communication with the IoT devices needs to be routed to a Device Gateway that has access to the public Internet in order to use cloud based tools like Azure.

Managing devices through Azure IoT Hub is therefore accomplished through using this Device Gateway to listen and maintain multiple device connections to the central IoT hub and send the command on to the appropriate device using a node.js program. Command responses are returned to the Device Gateway, sent back to the IoT Hub and eventually the appropriate User Interface.

It is observed that there was a stability issue relating to the Azure IoT Hub Node.js. The Azure IoT Hub Node.js SDK appears to have a bug when using the symmetric key authentication scheme. Occasionally, an error is triggered by the MQTT client software that can cause the entire system to crash. The X.509 certificate authentication protocol does not have this error and to enable maximum security, should be used for future authentication anyway.

Another issue encountered was that occasionally certain device connection string connections are lost and become 'undefined'. This may also be related to symmetric key authentication problems, but has not been verified.

#### 4.3.3.4 Device Monitor Interface

In the cloud setup, the User Interface connects to the IoT Hub using MQTT protocol and symmetric key authentication for sending commands to devices. Otherwise, the implementation details (commands to send to each type of devices) are the same as in the on-premises setup and are omitted here to prevent duplications.

### 4.4 Discussions and challenges

#### 4.4.1 Comments of current deployment

In the current deployment, due to the time limitation of the project, only primitive setups were implemented to investigate the possibility of integrating data flow, storage and device management to both the on-premises and cloud platforms. In comparison to Figure 4.1, data aggregation/device gateway, format transformation, storage and part of the data visualization functions were implemented during the project regarding the data plane. The collected data were re-formatted in the compatible formatting proposed by VdM before storing to the database. Regarding the control plane, the device gateway, device control middleware and device control interface were implemented for demonstration. However, only device availability status check and remote reboot features for selected devices were implemented in the prototype management platform.

In this project, access control functionality is not implemented due to the time restriction and also it is important to have the access control policies clearly defined before any implementation take place. This function block should be implemented in the next steps.

The benefits and drawbacks between the on-premises and cloud setup were also investigated. Cloud setup requires administrators to either allow the devices to communicate directly to the Internet or having a data aggregated in some intermediate servers before pushing the data to the cloud. This setup may impose some security threats which needs to be considered. With respect to the on-premises solutions, it is observed that free open-source solutions tends to have limited capabilities than the commercialized software, which may hinder the scalability of the network. In addition, on-premises solutions require constant maintenance and upgrade of the infrastructure which may slow down the deployment. On the other hand, Cloud computing approach, with on-demand resource usage, allows for quick deployment of applications and services with ubiquitous access. However, Cloud computing is not a one-size-fit-all

solution for a Smart City deployment. Firstly, Cloud resources are normally subjected to quotas and limits imposed by the service provider which may hinder the scalability of the network. Secondly, in comparison to the on-premises solution which is implemented in local data center, cloud-based solution is often implemented in global network of data centers managed by the cloud provider, which are not necessarily geographically close to the deployment site. This results in higher delay and lower throughput to transmit data to the cloud, which could greatly impact real-time applications. Lastly, most Cloud Computing services for IoT are still in development with limited capability, to implement a complete Smart City IoT application, additional development must be done targeted towards Smart City requirements.

#### **4.4.2 Challenges**

##### **4.4.2.1 Scalability**

With the scale of a vast network such as in a Smart City, scalability becomes the top issue in terms of data management. First of all, *storage capability and bandwidth* requirements in such a big network is very critical. Among the types of sensors, cameras are the most bandwidth demanding devices. With a network of ten thousands of cameras, the accumulated bandwidth is in terms of Tbps. How to route and store these flow of traffic is a big question, especially when real-time monitoring is required. Moreover, *connection management* also imposes a problem with the scaling of the network. With many thousands of connected devices, the management platform has to be able to handle that many transactions, parse the raw data, process them and store them at the same time, which demands a powerful computing platform to fulfil this demanding task.

##### **4.4.2.2 Virtualization structure for Smart City Infrastructure**

In a context of a Smart City, the smart infrastructure of Smart City has to be designed to be used as a public platform or to be shared for various applications from various parties such as police, fire fighter departments, hospital, governments as well as for private companies. This design is not only to maximize the benefits of this infrastructure; deploying a separate infrastructure for additional applications is cost ineffective, redundant and to some extents would limit the operations of the existing network due to interference and bandwidth contention. Moreover, in Smart City, it is not possible to predict all the future applications to be able to support them at the time of deployment. As the result, the concept of virtualization is brought into the picture to provide the abstraction of the physical resources so that different applications can utilize the same physical infrastructure for added value at the same time. However, the realization of such idea is not easy as it requires complex design and communications both at the physical layer and the application layer.

##### **4.4.2.3 Data access control**

As a Smart City grows, its number of sensors and accordingly data and applications also increases. Consequently, an important issue is how to control the access to this voluminous amount of data and devices. Data and device access is the key to derive new knowledge and applications but also impose public safety and privacy threats. For instance, misuse of actuators such as traffic lights or road signs can cause multiple threats in public safety. Strict rules of who can access to which may not be the best solution but implementing a more flexible and adaptable way of *data and device access control depending on the usage contexts* (location, time, social events, etc.) may not be straightforward.

#### **4.5 Chapter summary**

In this chapter, the data flow, storage and control structures of the deployed MSCPS are described. Although many IoT platforms were developed, most of them can only support a limited number of device manufacturers and models. As service/device provider “locked in” is one of the factors that should be

avoided, a flexible, adaptable and customizable centralized data management structure is one of the most important components in a Smart City setup. The implementation details for data and storage management as well as the device control management are illustrated for both cases of on-premises and cloud setups. Regarding the data and storage management, it is observed that both on-premises and cloud-based solutions have their own problems in terms of scalability, storage, bandwidth and delay to support the full-scale, vast amount of devices. In terms of device control management, with a wide range of device types from different manufacturers, it is hard to implement a unified management platform for device control and re-configuration. The chapter also highlighted some of the key challenges need to be studied before a full-feature management platform is feasible.

# Chapter 5

## Security

### 5.1 The importance of IoT security, examples and common threats

#### 5.1.1 The importance of IoT security:

As the deployment of IoT becomes popular and IoT devices become abundant, it opens up a new dimension for hackers. On one hand, hackers can make use of these sensors to collect or modify information. It does not sound much in hacking into someone fitness tracker or a light bulb, but putting in a right context these will become more serious. For example, hacking into someone fitness tracker would allow the attacker to know the trajectory and the current position of the person accurately, in real-time. In IoT systems with actuators, the threat is even more serious as the attacker can take over road displays or send modified sensing information to trigger an actuator. Moreover, taking over an IoT sensor can help the attacker step by step compromise the whole IoT system considered. On the other hand, if the information is not useful enough for the hacker, the device itself can be used as a bot in the botnet of the hackers [60], waiting for the right time to launch a Distributed Denial of Service (DDoS) attack. With billions of connected devices floating around, this threat will only grow. A prime example of this kind of attack is the DYN DDoS attack in Oct, 2016 [61]. In this attack, the attacker used a new weapon called the Mirai botnet, which comprised of hundreds of thousands of IoT devices such as digital cameras and achieved the immense rate of tetra bits per second [62].

Looking back at Figure 2.5, it is apparent that the only difference in terms of architectural design between the IoT-based Smart City and other ICT systems is the presence of the Data Acquisition layer which consists of sensors with different capabilities. As the result, *the main focus of this chapter is on the Data Acquisition Layer* and security measures as security at other layers can be realized using available standardized mechanisms. The security breaches in IoT-based networks can be classified into two categories:

- **Falsified data injection.** In this category, the attackers try to *inject fake data* into the network. There are many ways that the attackers can execute this attack such as masking a camera or tricking a sensor to report an incorrect readings and as the IoT network in a Smart City is normally physically exposed, this is surface of attack is quite easy to perform. This type of attack can be dangerous if the falsified data can directly trigger an actuator which can cause a domino effect on to the operation of the whole system. As the sensor itself can be malfunctioned due to environmental conditions, it is merely impossible to differentiate between the two cases. In order to mitigate the effects of falsified data, *outlining data detection mechanisms* can be employed to highlight the data that is derived from the normal values and depending on the context, the outlined values can be eliminated. In addition, *data correlation methods* can be implemented to combine data from many sources to lessen the impact of the falsified data before any action is taken.
- **Device/System penetration.** In this type of attack, the attacker *tries to take control of the IoT devices*

in order to collect private data, manipulate device operations and actuators, or to execute further penetration into the Smart City IoT system in the Management and Application layers. This type of attack is more severe than the previous categories since the types of actions that the attacker can perform is wider and have more serious effects on the IoT system. In this case, the infected devices have to be quickly identified and eliminated, i.e. all communications with it should be terminated, to prevent further infections in the network. Due to the nature of the IoT network, this type of attack can be performed locally in the physical vicinity of the device or remotely via communication infrastructure, even from the Internet.

As illustrated above, the later type of security breach is more significant and will be the focus of this chapter. For the first category, outlining data detection and data correlation methods can be employed to eliminate/mitigate the fake data.

### 5.1.2 IoT security vulnerability examples

In this subsection, through showing various examples of currently available IoT devices which have security vulnerabilities, common security threats are illustrated to identify devices that should be avoided. Representative examples will be given in the area of consumer devices first, then, illustrations of the vulnerable devices in smart cities contexts will also be discussed to give a more insightful taste.

#### **Consumer devices:**



**Figure 5.1:** A smart device running modified firmware [59].

At DEF CON 2016, two researchers from a reputable security company illustrated the security vulnerabilities of some of the widely available smart locks [63]. Their research shows that many of the available smart locks do not integrate good security measures and can be easily hacked. Many of the devices under test *do not support encryption* and credentials are communicated in plaintext. Some of the others do support encrypted communication but still *vulnerable to replay attack* due to the lack of using random values to differentiate between sessions.

In [64], the authors investigate some household devices to show some security vulnerabilities in these devices. In one of the devices, the study shows that the communication is entirely in *plain text*. More seriously, the devices communications were conducted *without authentication* to ensure that the commands were coming from a legitimate device. These two issues are of major concerns as they can be easily exploited by an attacker.

In [65], the authors illustrate a method to gain access to both consumer device (a smoke detector) and an industrial device (a smart meter). The attack makes use of the physical access to the devices to *open a*



*UART console* which consequently helps to extract the login information and changing of the devices parameters.

Perhaps, the most comprehensive report was done by Symantec in one of its security responses in 2015 [59]. The authors analyzed 50 smart home devices that are available and the findings are quite fearful with various serious security issues in majority of these devices. First, although the use of *weak passwords* is not really a new issue, many vendors do not force the user to change the default login credential, do not enforce the use of strong passwords or even do not allow the use of such complex passwords, leaving the users with only a 4-digit PIN number for login. Moreover, most of the devices *don't lock users out after a number of failed login attempts*, making it possible to perform brute-force attack. Some of the devices *does not utilize encrypted communication* which allows the attacker to conduct man-in-the-middle attack, replay attack to login, control the device or even upload new firmware to the devices. In addition, most of the tested devices *do not use encrypted nor digitally signed firmware updates*, which makes it easy for an attacker to generate and install malicious firmware. Some of the tested devices *do not verify the digital certificates* and approve all connections as long as they are done over HTTPS and none of the tested devices perform *mutual authentication*, leaving only the server authenticating with the client.

#### **Smart city devices:**

Recent years witness the mushroom of the number of smart cities around the globe [66]. However, although the huge amount of money that governments and cities' authorities put in these projects, it was not long that several security holes were found and published.



**Figure 5.2:** Hacking into a road display<sup>12</sup>

In [67], the authors explore the vulnerabilities in transportation systems using RFID cards. The investigated systems including car parking tokens, bike renting cards and bus-tram-metro cards. The research highlights several issues in RFID card configuration as well as system structure that allows the attacker to abuse the system by cloning the cards for free rides, and parking.

In [68], the authors reveal several security vulnerabilities which can be exploited in smart city's infrastructure. The types of devices under test ranging from smart terminals in government offices, airports and bike rental facilities to street cameras. The study shows that the terminals that allow user interactions are not well implemented and attackers can *escape the kiosk mode* quite easily through several techniques to enter commands and change configurations. It is also shown that many of these devices store sensitive data such as login credentials in *clear text*, making it easy for retrieving the logins,

<sup>12</sup> Hacking into a road display - [https://www.washingtonpost.com/news/morning-mix/wp/2016/06/06/somebody-keeps-hacking-these-dallas-road-signs-with-messages-about-donald-trump-bernie-sanders-and-harambe-the-gorilla/?utm\\_term=.dad2542fe0c0](https://www.washingtonpost.com/news/morning-mix/wp/2016/06/06/somebody-keeps-hacking-these-dallas-road-signs-with-messages-about-donald-trump-bernie-sanders-and-harambe-the-gorilla/?utm_term=.dad2542fe0c0)

passwords, payment details or further penetrations. The authors also tested speed cameras installed on the streets and found that the cameras' streams are *open without any password protection*; moreover, the authors showed the possibility to change the camera configuration to disable detection functions on certain lanes, making the trustworthiness and privacy protection of these cameras questionable.

**Table 5.1:** Common Security threats and countermeasures [70]

Attack	Description	Countermeasures
Data alteration and modification	Malicious users can break integrity of exchanged data by alerting, deleting or fabricating its content	Public Key Infrastructure (PKI) can be used to counter such attacks.
Masquerade attack	Malicious entities duplicate valid identities to achieve their goals.	Identity certificates issued by trusted authorities are used to identify entities. Certificate revocation list mechanism can be utilized to expel compromised certificates.
Replay attack	Malicious entities repeatedly send valid data gathered maliciously to gain access to the system.	A system can keep a cache of exchanged messages to compare with new messages.
Man-in-the-middle attack	The attacker intercepts the communication between two entities, while the entities think they have direct communication.	Cryptographic solutions.
Sybil attack	Impersonation attacks	Explicitly bind an identity to participating entities in a system using PKI based identities and cryptographic solutions
GPS spoofing	Attacker can make entities believe that they are in different position. GPS time synchronization can also be affected.	Implement plausibility rules such as using location and signal strength of satellite to detect any malicious and abrupt changes.
Broadcast and Transaction Tampering	Broadcast tampering injects of false data into the system by malicious users using broadcast messages. Transaction tampering could tamper transmitted data or create an alternate reply to unsuspecting users.	Usage of authentication and digital certificates.
Eavesdropping and Traffic Analysis	Attacker tries to interpret the transmitted data.	Encryption and cryptographic solutions
DoS	Jamming, spamming, flooding	No effective countermeasure
Malware	Infect the infrastructure and its component with malicious software such as Trojans, viruses, worms.	Signing of the software, keeping the software up-to-date.
Brute Force	Attacker uses all possible combinations of keys to gain unauthorized access to the system/data.	Using of strong cryptographic algorithms.
Timing attack	Attacker tries to compromise a system by analyzing the time taken to execute cryptographic algorithms.	Use time-stamps and digital signatures.
Conflict Collision	RFID tags try to transmit data simultaneously.	RFID anti-collision techniques can be used.
Node security	Several methods can be used such as node insertion	Trust management techniques can be used to ensure the trustworthiness of nodes and communication.
Security attacks on devices with limited computation and storage resources	Many different attacks can be performed.	Use of lightweight cryptographic solutions such as Tiny SA
Routing attacks	Many different attacks can be performed such as sinkhole attacks	Use of secure routing protocols.

In [69], the authors show how traffic sensors can be manipulated using wireless connection. Using the *open Bluetooth* connection, the authors demonstrate that sensor's firmware can be changed remotely. Moreover, the studies also show that configuration settings of these devices as well as the related data content can be altered through several techniques, making readings through these devices unreliable. Through the above examples, it is shown that although each and every new technology and innovation bring new benefits, they also come along with new challenges and problems. In the context of Smart City, dealing with security is a hard problem. Since sensors are exposed to the environment, these devices have

a significant disadvantage of lacking of physical security. With so much complexity and interdependency, Smart Cities exhibit a large attack surfaces and it is difficult to make sure that everything is secure. Another important issue is that while common security threats and countermeasures are already identified as shown in Table 5.1, due to the wide range of capabilities of IoT devices in Smart City, not all features may be implemented. Moreover, as the new technologies in Smart City are not quite mature, thorough security tests may not be done properly by the vendors. To add more complexity into the picture, in smart cities, there is usually coexistence of old and new devices and technologies, assuring both interoperation and security at the same time is a difficult task.

## 5.2 Guidelines and best practices for Smart City planning and implementation with regard to security aspects

In a Smart City, the diverse infrastructure and the interconnections between different systems under different administration make security a hard task to be accomplished. In this section, we don't have the ambition to cover detailed information in security testing or assessment procedure in a Smart City but rather, an overview of the key factors that should be aware of in the planning, implementation, operation, maintenance and disposal of Smart City's technologies. This part is summarized based on the materials in [71], [72].

### 5.2.1 Key security requirements

In a Smart City, functionalities are what bring the added value needed but security is what will help a solution survive in a long term. In the context of Smart City, weak security is a serious issue as it could cause large-scale damage, even affecting the security and privacy of a national and its citizens. As a result, Smart City solutions should comply with key basic security requirements as the following

- *Strong cryptography to protect data, both at rest and in transit.* Data in Smart City are of critical importance in the privacy and the safety of its citizens, as a result, all data communications in Smart City should be protected by strong cryptography mechanisms to guarantee their confidentiality, integrity and availability. Moreover, for data that are stored for processing or archiving, protecting them with appropriate cryptography mechanisms is also important. For instance, some of the well-known encryption algorithms are Advance Encryption Standard (AES), Triple Data Encryption Standard (3DES), RSA...
- *Authentication capabilities.* Access to all systems in a Smart City should be guarded by authentication mechanisms. Beside the basic username/password authentication, enhanced authentication mechanisms such as one-time passwords, digital certificates or multi-factor authentication should be supported for future proof. For example, one-time password tokens, X.509 digital certificates, and biometric authentications are some of the industrial solutions in this aspect.
- *Authorization capabilities.* Privileges and permissions should be properly assigned for accessing functionalities and performing actions in a Smart City.
- *Automatic and secure update of software and firmware.* Software and firmware updates should be delivered in an automatic and secure way. Some examples of proposals for securing firmware updates including encryption of firmware, using digital signatures...
- *Auditing, alerting, and logging capabilities.* All systems should facilitate auditing and logging of events. Logs should be stored securely against tampering.
- *No backdoor/undocumented/hardcoded accounts.* These kinds of accounts cannot be removed or disabled and may have passwords that cannot be changed allowing compromising the deployed system easily. Removing these accounts is important and should be enforced in the contracts with the vendors.
- *Non-basic functionality disabled by default.* Only basic functionality should be enabled by default, and the rest should be provisioned when needed.

- *Fail safe/close*. In the case of a system malfunction or crash, the system should remain secure, for example, does not reveal the cryptographic keys, login credentials or specific communication implementations.
- *Secure by default*. Smart City's device should come with a secure set of configurations out of the box instead of depending on the users to enable these security functions.

Moreover, a Smart City deployment usually stretches across multiple sectors and understanding of the interconnectivity and dependency between different sectors is important to prevent large-scale damages. Figure 5.3 illustrates an infrastructure dependency matrix [73].

Sector	Element	Energy & Utilities					Services		
		Electrical Power	Water Purification	Sewage Treatment	Natural Gas	Oil Industry	Customs and Immigration	Hospital & Health Care Services	Food Industry
Energy & Utilities	Electrical Power		L			M			
	Water Purification	H				M			
	Sewage Treatment	M	H			H			
	Natural Gas	L				L			
	Oil Industry	H	L						
Services	Customs & Immigration	H	L	L	L	L		L	
	Hospital & Health Care Services	H	H	L	H	H	M		H
	Food Industry	H	H	H	L	M	M	L	

Key: **H** High **M** Medium **L** Low

**Figure 5.3:** Infrastructure Dependency Matrix.

### 5.2.2 Security considerations in Implementation

When the Smart City's devices are deployed, it is important that the deployment is done in a secure way. Some of the best practices should be considered are shown as the following.

- *Enable strong encryption*. All communications should be properly protected against unauthorized eavesdropping, interception, and modification. Encryption keys must be well protected.
- *Secure system administration*. Avoid using a single administrator user to perform all actions on all systems. A common practice is to use different administrator users and passwords at different permission granularities. For example, there should be separate accounts for monitoring and configuring of the devices.
- *Use of strong passwords*. Password policy must be defined for password strength and duration of validity. Other strong authentication mechanisms such as one-time password, digital certificates or multifactor authentication are recommended to be used to replace legacy username/password whenever possible.
- *Remove unnecessary user accounts*. Test/default accounts should be removed. Specific accounts used in the implementation process should be removed after the process is completed. These accounts should be documented for the ease of identification.

- *Disable unused functionalities and services.* Disabling of unused functionalities and services reduces the attack surface and prevents possible attacks that abuse weakness in those functions and services. Examples of these unnecessary connections/services include Bluetooth, telnet, SSH, UPnP, USB etc. Legacy HTTP login page which will send credentials in clear text should also be disabled. Anonymous logins or streaming should be disabled as well.
- *Enable auditing of security events.* Security events and device's logs should be enabled and monitored to identify ongoing attacks and breaches. Normal operating thresholds for each device and system type should be identified and documented. Based on these baselines, variance should be determined to trigger an alert and the criticality of that alert. In this case, a customized automated auditing software should be required.
- *Anti-tampering, anti-vandalism mechanism.* Devices should be protected against unauthorized physical access for modification, vandalism or device stealing. Moreover, masking of devices is also recommended to hide the specific model or label on the device which may help the attacker to decode the communication protocol or exploit specific bug easier.

### 5.2.3 Security considerations in Operation and Maintenance

Once implemented, continuous support, tracking and monitoring of components of a Smart City are required.

- *Monitoring.* Regular monitoring of system logs or other available mechanisms is required to track the stability of the services, suspicious activities, abnormal behavior, performance hooks or any other service-threatening events.
- *Patching.* Vendors are expected to collaborate on deploying the latest security patches. Patches are expected to be tested in the lab environment first. Firmware updates should be delivered in a secure manner with encryption and digital signature.
- *Regular assessment and auditing.* Vulnerabilities and exploits are announced and published everyday so being ready to respond is imperative to protecting the Smart City infrastructure. Testing of smart services is expected to run continuously, especially after applying new changes to the system.
- *Protection of logging environment.* All logs should be safely transmitted and stored. The integrity and protection of the evidence is critical to investigate misbehavior, track incidents and do legal liability.
- *Access control.* Monitoring who, when and how someone has access to smart device systems is critical to prevent unplanned changes, tampering or downtime.
- *Compromise reaction and recovery.* It is important to create detailed procedure manuals that define what must be done if a smart city system is compromised. These procedures should include certificate revocation, key re-initiation, and system isolation and clean up, as well as how to follow up on the incident in order to understand how the system was compromised and develop appropriate actions that will prevent it from happening again.

### 5.2.4 Security considerations in disposal of Smart City devices

Although deployment of Smart Cities is a recent trend, it is not early to think about the decommissioning of these devices in a near future. Regarding the security aspects, it is important to develop specific policies for securely disposal of Smart City's devices as important security related information may be leaked through these decommissioned devices. Several factors need to be considered when performing this task.

- *Avoid repurposing of devices.* This action could leak sensitive design, software, password or cryptographic information, which could create threats to the currently devices in services.
- *Securely erase data on system storage.* This is a good practice to apply, the destruction of storage may be required to assure the safe and quick disposal of critical data.
- *Vendor replacement.* During maintenance and support duties, vendor's personnel could replace a hardware. It is important that hardware that is removed from live environments is well protected



and vendors are expected to provide secure technology disposal as a part of their services and maintenance contract.

### 5.3 Summary of security features that are enabled in the pilot deployment

In this subsection, the summary of security features that are enabled in the MSCPS are presented. This subsection is further supported by *Appendix G*.

In the pilot deployment, the supported security features are investigated but many of the current configurations are not configured to maximize security abilities due to the purpose of testing and making quick configuration changes. For example, some extraneous services such as enabled webserver are very helpful. Once the centralized device management platform is fully developed, these services can be disabled to harden the security in the network. Unencrypted HTTP service remains enabled on many cameras due to a problem with viewing the camera videos with it disabled. This problem could be fixed by firmware updates.





Security on the data plane is limited to what is commercially available. Unfortunately, few devices at the present can encrypt data stream. None of the cameras have the ability to send encrypted video and none of the deployed sensors such as traffic radars, temperature sensors and other devices deployed have the option to send data encrypted. However, sensing data can be still protected from tampering by using the encryption capability of radios devices in the Data Acquisition Layer and the VPN setups from the gateway to the database in the Communication Layer.

It is important to note that security spans across the whole Smart City system and hence cannot be treated as an isolated component. As a result, security has to be integrated in all steps of the Smart City Deployment from the design and after an architecture and management structure are finalized, the objectives of security relevant steps can be clarified.

#### 5.3.1 Radios

In the MSCPS, wireless communication is the primary way to bring data to the fiber drops, before going to the database. On the data plane, the wireless devices support several mechanisms for encryption and the highest security features are chosen to be configured. On the control plane, beside the secure SSH access control, other unsecured methods such as HTTP and SMTPv1 are still enabled for the ease of testing and quick configuration changes. After the pilot setup, those unsecured communications should be disabled.

**Table 5.2:** Security features that are enabled on Radio devices.

		 <b>Ubiquiti NanoBeam NBE- 5AC-16</b>	 <b>Ubiquiti NanoBeam NBE- M5-16</b>	 <b>Ubiquiti Rocket M5</b>	 <b>Digi Xbee 232 Adapter S1 Pro</b>
<b>Data Encryption</b>	Capabilities	WPA2-PSK, WPA2-EAP	WPA2-PSK, WPA2-EAP, WPA-PSK, WPA- EAP	WPA2-PSK, WPA2-EAP, WPA-PSK, WPA-EAP	AES
	Currently Enabled	WPA2-PSK	WPA2-PSK	WPA2-PSK	AES







	Deployment Suggested Guidelines	WPA2-PSK	WPA2-PSK	WPA2-PSK	AES
<b>Secure Device Configuration Protocols</b>	Capabilities	SSH, HTTPS	SSH, HTTPS	SSH, HTTPS	N/A
	Currently Enabled	SSH, HTTPS	SSH, HTTPS	SSH, HTTPS	N/A
	Deployment Suggested Guidelines	SSH	SSH	SSH	N/A
<b>Insecure Device Configuration Protocols</b>	Capabilities	Telnet, HTTP, SNMPv1	Telnet, HTTP, SNMPv1	Telnet, HTTP, SNMPv1	RS232
	Currently Enabled	HTTP, SNMPv1	HTTP, SNMPv1	HTTP, SNMPv1	RS232
	Deployment Suggested Guidelines	disable	disable	disable	RS232
<b>Authentication Capabilities</b>	Capabilities	Digital certificate for HTTPS, RSA Keys for SSH, Basic Password	Digital certificate for HTTPS, RSA Keys for SSH, Basic Password	Digital certificate for HTTPS, RSA Keys for SSH, Basic Password	N/A
	Currently Enabled	RSA Keys for SSH, Basic Password	RSA Keys for SSH, Basic Password	RSA Keys for SSH, Basic Password	N/A
	Deployment Suggested Guidelines	RSA Keys for SSH	RSA Keys for SSH	RSA Keys for SSH	N/A

### 5.3.2 Cameras

Regarding the Cameras, none of the deployed camera supports secure video streaming. As a result, only the available insecure RTP and RTSP protocols are enabled. It is noticed that some cameras have options for allowing anonymous users to view the live video as discussed in section 5.1.2. This feature has been disabled and should be disabled for any newly deployed cameras to prevent unauthorized viewing of camera streams. For Axis cameras, Owner Authentication Key (OAK) can be used for communication with Axis Video Hosting System (AVHS) propriety middleware and access to the video streams globally, provided the AVHS service is turned on at the cameras and Internet connection is available for the cameras. OAK can be obtained alongside with the serial number in the packaging of devices.

On the control plane, HTTP, HTTPS and several versions of SMNP protocol are enabled for configuration and testing purposes. However, after the pilot project, it is recommended that only HTTPS and SMTPv3 are enabled for the best security measures. For authentications, passwords are currently in used. In addition, digital certificate and HTTPS can also be used (the detail procedure for setting up digital certificates with a private Certificate Authority is shown in Appendix G).




**Table 5.3:** Security features that are enabled on Cameras.

					
		<b>Axis Q6128-E</b>	<b>HikVision DS-2DF6236-AEL</b>	<b>HikVision DS-2CD4585F-IZH</b>	<b>Panasonic WV-SW598A</b>
<b>Secure Video Streaming Protocols</b>	Capabilities	N/A			
	Currently Enabled	N/A			
	Deployment Suggested Guidelines	N/A			
<b>Insecure Video Streaming Protocols</b>	Capabilities	RTSP/RTP	RTSP/RTP	RTSP/RTP	RTSP/RTP
	Currently Enabled	RTSP/RTP	RTSP/RTP	RTSP/RTP	RTSP/RTP
	Deployment Suggested Guidelines	RTSP/RTP	RTSP/RTP	RTSP/RTP	RTSP/RTP
<b>Secure Device Configuration Protocols</b>	Capabilities	HTTPS, SNMPv3	HTTPS, SNMPv3	HTTPS, SNMPv3	HTTPS
	Currently Enabled	HTTPS	HTTPS	HTTPS	HTTPS
	Deployment Suggested Guidelines	HTTPS, SNMPv3	HTTPS, SNMPv3	HTTPS, SNMPv3	HTTPS
<b>Insecure Device Configuration Protocols</b>	Capabilities	HTTP, SNMPv1, SNMPv2c	HTTP, SNMPv1, SNMPv2c	HTTP, SNMPv1, SNMPv2c	HTTP, SNMPv1
	Currently Enabled	HTTP, SNMPv2c	HTTP, SNMPv2c	HTTP, SNMPv2c	HTTP, SNMPv1
	Deployment Suggested Guidelines	Disable	Disable	Disable	Disable
<b>Authentication Capabilities</b>	Capabilities	Digital certificate for HTTP(S), Digest Password	Digital certificate for HTTPS, Basic Password	Digital certificate for HTTPS, Basic Password	Digital certificate for HTTPS, Basic Password
	Currently Enabled	Digest Password	Basic Password	Basic Password	Basic Password
	Deployment Suggested Guidelines	Digest Password	Basic Password	Basic Password	Basic Password

### 5.3.3 Adapters

For the deployed adapters, no data encryption is supported. On the control plane, SSH, HTTP and SMTP protocols are currently enabled, but for the best security hardening, only SSH is recommended. It is noted that the USR IOT adapters are not capable of secure configuration.


**Table 5.4:** Security features that are enabled on Adapters.

		 <b>USR IOT USR N-520</b>	 <b>Lantronix ED2100002-01</b>	 <b>Perle IOLAN SDS2 W</b>
<b>Secure Device Configuration Protocols</b>	Capabilities	N/A	SSH, HTTPS	SSH, HTTPS, SNMPv3
	Currently Enabled	N/A	SSH, HTTPS	SSH, HTTPS,
	Deployment Suggested Guidelines	N/A	SSH	SSH
<b>Insecure Device Configuration Protocols</b>	Capabilities	HTTP	HTTP, Telnet, SNMPv1	HTTP, Telnet, SNMPv1, SNMPv2
	Currently Enabled	HTTP	HTTP, SNMPv1	HTTP, SNMPv2
	Deployment Suggested Guidelines	HTTP	Disable	Disable
<b>Authentication Capabilities</b>	Capabilities	Basic password	Digital certificate for HTTPS, RSA Keys for SSH	Digital certificate for HTTPS, RSA Keys for SSH
	Currently Enabled	Basic password	RSA Keys for SSH, Basic password	RSA Keys for SSH, Basic password
	Deployment Suggested Guidelines	Basic Password	RSA Keys for SSH	RSA Keys for SSH

### 5.3.4 LTE Gateway

In the MSCPS, Sierra LTE gateway is mounted on a mobile vehicle for communication purpose. In the data plane, data encryption is supported via VPN feature; however, in the current setup, VPN is not enabled. For best security hardening, VPN can be used to prevent eavesdropping and data tampering. Regarding the control plane, several protocols, such as SSH, HTTPS, Telnet and HTTP, are left open for configuration and testing purpose, for the best security hardening, only secure protocols should be enabled such as SSH and HTTPS.

**Table 5.5:** Security features that are enabled on Adapters.

		 <b>Sierra Wireless GX450</b>
<b>Data encryption</b>	Capabilities	VPN
	Currently Enabled	Disabled
	Deployment Suggested Guidelines	VPN
<b>Secure Device Configuration Protocols</b>	Capabilities	SSH, HTTPS
	Currently Enabled	HTTPS
	Deployment Suggested Guidelines	SSH, HTTPS
<b>Insecure Device Configuration Protocols</b>	Capabilities	Telnet, HTTP
	Currently Enabled	Telnet, HTTP
	Deployment Suggested Guidelines	Disable
<b>Authentication Capabilities</b>	Capabilities	Basic Password, RSA Keys for SSH
	Currently Enabled	Basic Password
	Deployment Suggested Guidelines	Basic Password, RSA Keys for SSH

### 5.3.5 Non-IP Devices

The non-IP devices deployed include RFID readers, traffic radars, temperatures sensors, and ultrasonic level sensors. These devices do not support any specific security implementations. As a result, those devices have to rely on the security measures in other devices such as WiFi encryption or VPN to prevent eavesdropping and data tampering.

### 5.3.6 Data at rest Security

After data have been transmitted from the sensors to the data center, they are stored for further processing. While being at rest, the data can be tampered with, can be extracted for bad purposes and hence must be protected as well.

For data stored in on-premises storage, encryption for data at rest depends on the management applications as well as the storage device. Most free or inexpensive on-premises applications, such as Zone Minder and Xeoma, do not support directly encryption of video recordings, which are stored in playable multimedia file formats (e.g. MP4, AVI, etc.). Instead, the NAS that archives the recorded data can support AES-256 encryption standard to ensure data security.

On the other hand, enterprise software often includes built-in encryption. For example, XProtect only provides recording server encryption for XProtect Expert or Corporate versions, which supports AES-256 encryption at 2 different levels: Light and Strong. Light encryption only encrypts the header of video data while Strong encrypts all parts of video data at the cost of higher CPU processing requirement. In the pilot deployment, we use XProtect Essential+ version, which does not support any encryption. Likewise, MongoDB Enterprise Advanced supports AES-256 in CBC mode by default, while other available options

are available including GCM mode as well as FIPS mode for FIPS-140-2 compliance. Our on-premise MongoDB deployment software is a free version and does not support these encryption procedures. Cloud-based storage typically provides integrated encryption-at-rest by the cloud service provider. Microsoft includes Azure Storage Service Encryption (SSE) as an option for their IaaS services, such as Azure Storage. SSE automatically encrypts as new data is written to storage. Higher level services built on top of Azure, such as CosmosDB and Stratocast, instead have encryption enabled by default. CosmosDB does not provide control over encryption options to make it easy to use for users. Stratocast also delegates its data security to Azure infrastructure, instead focusing on securing data in transit for all connections between users and the cloud server.

## 5.4 Challenges

Given that it takes the Internet several decades to be mature, and it is still evolving; the development of IoT standards is still in its infancy and there are still many challenges to be overcome.

- **Physical security of hardware and firmware for IoT sensors.** Most of the standards in IoT focus on the communication aspects for the data transport in IoT network, leaving an unexplored area of research in physical security of IoT devices. In the context of Smart City, being exposed to the environment, IoT devices have a significant disadvantage of lacking of physical security. Having physical access to the device, an adversary can extract the security materials or the stored data on the device as well as gaining insights on the detail implementation for further exploit and penetration. Therefore, solutions for protecting the IoT sensors at the physical level still needs some further developments.
- **Secure configuration and re-configuration of IoT sensors.** During the life cycle of an IoT device, it is needed to be reconfigured and updated regularly to catch up with the functionality requirements and to apply security patches. Due to the numerous devices in a large scale deployment, automation of the update process is mandatory. An adversary can exploit this batch operation to inject malicious modifications or Trojan into the devices' firmware. As a result, a secure mechanism for reconfiguration of the IoT sensors is of top priority before any large-scale IoT deployment is feasible.
- **Efficient support for public keys and Digital certificates.** Although many developments in the field of microprocessors, the use of public keys is believed to be computation burden on these constrained devices. However, the use of public keys and digital certificates would open up the possibilities for mutual authentication, ease of security provisioning and isolation of contaminated IoT devices. For instance, this measure enables device authentication to prevent malicious devices from joining the network. As a result, a light-weight efficient support of public keys and Digital Certificates on constrained devices requires further investigations.
- **Efficient public key management infrastructure and online verification.** One important issue in IoT is the lack of appropriate key management mechanisms and online verification for the validity of certificates. These schemes would allow the ease of securely manage users and devices on a large-scale.
- **Multi-factor authentication for constrained devices.** In the context of Smart City, physical access may not be always restricted, especially for those devices that provide direct services to the citizen. Providing a means for efficient and convenience authentication of the device, especially in the installation phase is an important task to guarantee the joined device is trusted. Multi-factor authentication is a promising solution for this issue but how to deploy this method and manage identities for constrained devices in a large scale is still an open question.
- **Data Analytics and Artificial Intelligent (AI) – based Security.** Security is a very dynamic domain, new threats and types of attacks are evolving every day. To cope up with those attacks that are not yet documented, new mechanisms of security should be developed using data analytics and AI to quickly identify the new attack and come up with a counter measure to protect the system and its data.

---

## 5.4 Conclusion

In IoT, the issue of security is a very crucial topic. Due to the constrained IoT devices with low complexity and low power consumption, security hardening these devices is hard. In this chapter, the importance of IoT security is presented with illustrated security breaches in both consumer and smart city context to highlight the issue. Guidelines and best practices regarding security aspects in the context of Smart City are also discussed. Security features that are enabled in the pilot deployment are then summarized and commented upon. It is observed that while vendors make sure to secure open communication channels, such as wireless data transfers, few vendors secure data directly when it is collected, i.e., at the sensors. None of the cameras, traffic radars, or other sensors have the ability to encrypt the sensed video or data. In addition, key IoT security challenges to be studied is also discussed. With the versatility of IoT in a wide range of application, especially in the context of Smart City, security threats should be considered seriously in all phases of implementation and operation. Moreover, due to the evolvement of security attacks on the horizon as well as the infancy of IoT world, significant efforts have to put on developments of effective methods to proactively and aggressively address these security threats.



# Chapter 6

## Data Analytics – Example Functions and Results

*Data analytics by itself is a big field and solutions in this area are normally either condition, context, or application specific to achieve an acceptable performance. A generic “one solution fit all” is not yet possible. The aim of this chapter is not to explore the whole area, do extensive testing or to give a concrete solution but rather to present some promising potentials of data analytics based on well-established algorithms and the possibility of provide some data analytic applications on top of the deployed hardware.*

### 6.1 Motivations, benefits, and applications

#### 6.1.1 Motivations and benefits of data analytics for Smart City

With a fully deployed Smart City, the volume of collected data is expected to be very huge, far beyond the processing capability of humans for gaining the insights of the data and taking timely reactions. As an illustrative example, it is predicted that a city populated by 1 million people will produce 180PB of data everyday by 2019 [85]. To add more complexities into the picture, it is remarked that the collected data can also be different in their criticalities, for instance, some type of data may need a more stringent latency requirement than the other and these requirements can vary dynamically depending on the context [86]. As a result, monitoring and analyze these wide range and big volume of data is impossible for human and automated analytic functions are required to help to grind the raw data in order to provide more useful information from the massive collected data.

With the useful information extracted from the data analytics functions, the Smart City can benefits in several ways including economic, environment as well as social aspects.

- **Efficient use of resource.** With a high concentration of population in cities, the resources to support sustainable development become either scare or expensive. As a result, the efficient utilization of these resources is important. With a good resource monitoring system that can absorb and analyse data across the entire city, it is easier to point out the waste points. Based on this result, a more balance strategy of distributing resources can be applied to reduce cost, energy and natural resources consumption.
- **Interconnectivity and collaboration.** In the context of Smart City, there are many government parties that are involved in the operation of the city. Facilitates with data analytic functions, city government bodies can interconnect and collaborate with each other to be seen from the citizens’ view as a unified body which can save time, automate paper works and bring more convenience to its people.
- **Higher levels of transparency and openness.** With appropriate data analytic functions in place, the management and control of the data flows between various applications could be enhanced. This

improved data management in turns will levitate data interoperability and openness to a higher level. Information transparency offers the citizens and government bodies the opportunities to use data more effectively.

- **Better quality of life.** The ultimate goal of realizing Smart Cities is not to show off some cutting edge technologies but to improve the quality of life of its citizens. Facilitate by data analytics, better services, less waste environment, efficient transportation will allow Smart City citizens to have better decisions and hence a more efficient model for living and work.

### 6.1.2 Promising applications from data analytic applications

With many potentials that data analytics can bring about, many promising applications can be improved/realized in Smart City. Some of the representative examples of promising applications that data analytic can introduce are illustrated as below.

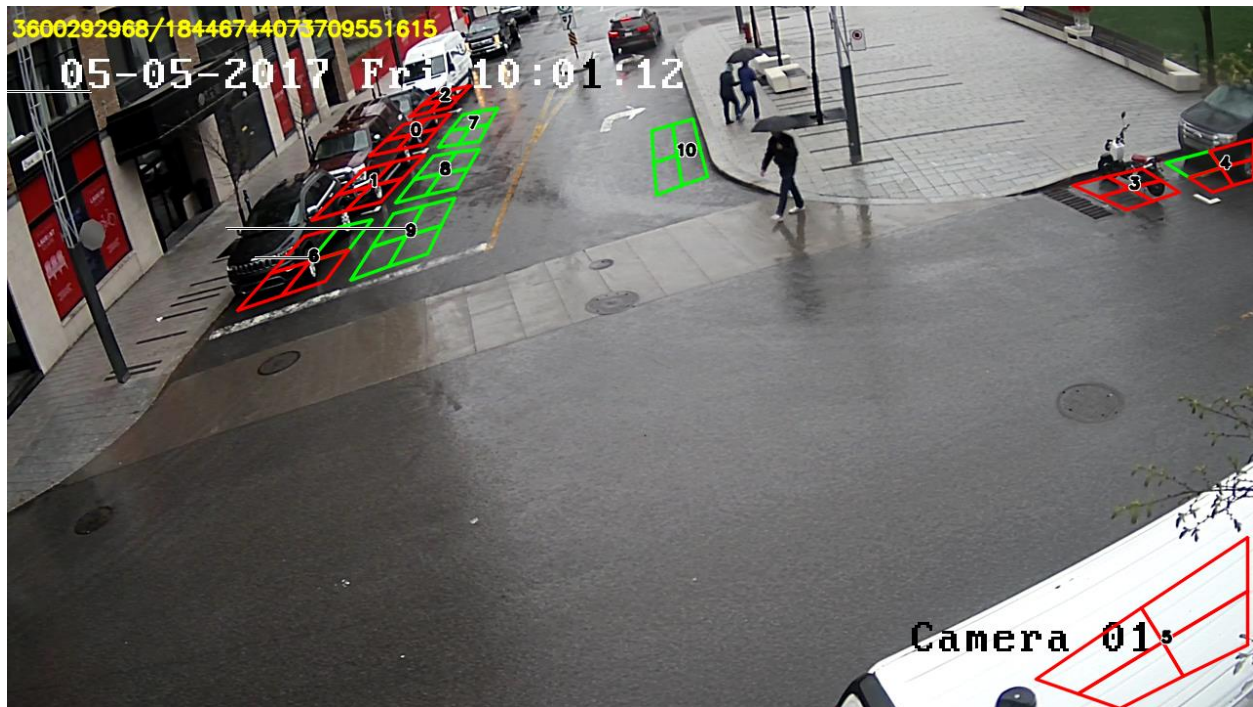
- **Smart Healthcare.** With sensors attached or even implanted into patient's body to collect information and sending data back for diagnosis. The collection of information is automated in real-time and the sensors can collect many types of data at the same time which can provide doctors with updated and complete set of information for monitoring the patient's health status. With the aid of data analytic functions, the data collection can be incorporated with various analytic processes to provide a constant monitoring and flag potential health issues either on daily or on demand basis. In this way, patients with a less severe condition can be monitored remotely at home or even at work, maximize the utilization of health care infrastructure so that more patients can be taken care of.
- **Smart Transportation.** Transportation is one of the most critical issue in every city around the world. Data analytic applications can help to recognize traffic patterns in real-time. These patterns can help to reduce congestions by predicting traffic conditions and adjusting traffic lights accordingly. For long-term planning, data analytics can help to optimize road infrastructure, guide the open of new roads. Information about free parking places can also be updated so that workers and shoppers can plan their travel accordingly.
- **Crowd Control.** With events and festivals happen frequently to attract tourism and promote commercial trading, crowd control has become a substantial concern among city governments. When so many people are concentrated in a small place, it is important to understand the crowd and predict its movements and actions. In this case analytic applications can help to estimate the number of people that are in the crowd, classify them, predicting the potential/dangerous trends in order to provide insight information to meet the crowd needs such as food, drink and to ensure their safety with timely emergency responses.

## 6.2 Example data analytic functions and results

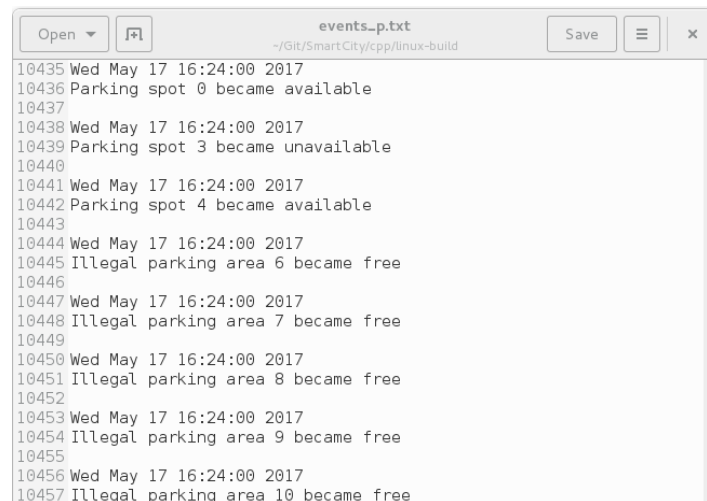
In the scope of the project, two analytic applications were developed to demonstrate the possibility of data analytic in the context of the pilot deployment. The two developed applications include parking place occupancy and vehicle counting, and crowd counting. The parking place occupancy and vehicle counting application allows user to define legal/illegal parking places and based on the analysis of the video footage to determine if a specific parking place is occupied or not. Another functionality of the program is to count the number of vehicles/pedestrians that enters/leaves an intersection. The second application is crowd counting to count the number of people in a scene. Using image processing algorithms, both the applications functions well with good accuracy. This part will give the summary of the results from these two applications as well as camera the impact of specifications on the results. The detail of the implementations will be described in *Appendix H and I*, attached with this main report.

### 6.2.1 Parking place occupancy and vehicle counting application

#### 6.2.1.1 Parking place occupancy



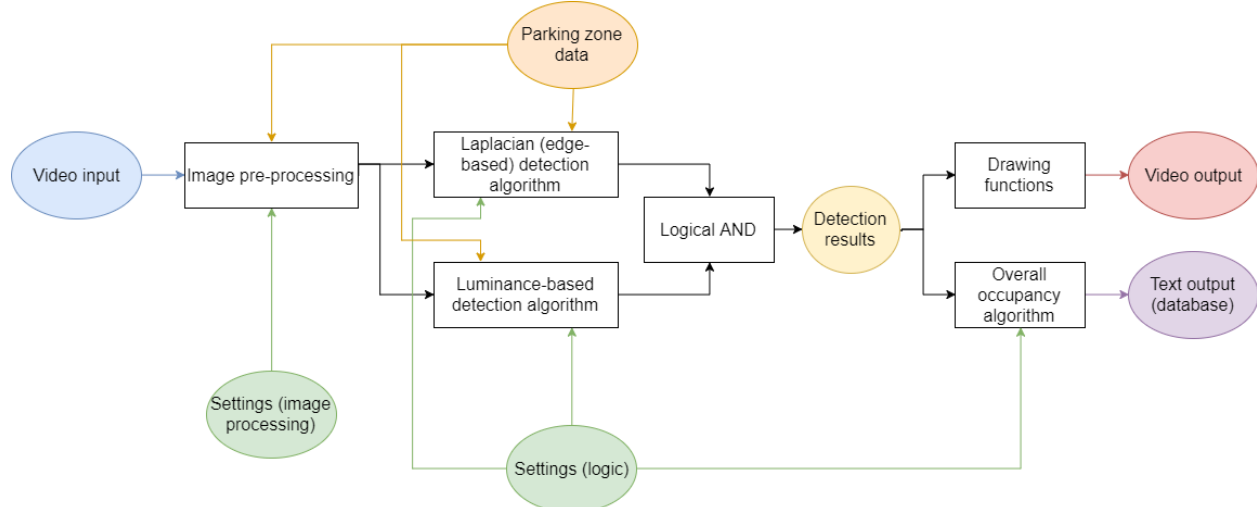
**Figure 6.1:** Parking place occupancy output frame.



**Figure 6.2:** Parking detection log file.

The parking place occupancy application is built based on the OpenCV library [88] to determine if a parking place is occupied or not. In addition, it allows users to define a certain parking place is a legal or illegal parking place for the ease of management. As shown in Figure 6.1, the parking detection software presents the user with a window displaying the current frame of the stream being analyzed, with an overlay of the defined parking zones and their associated ID in the center. Each individual zone is highlighted green if detected as free and red if occupied. In addition, the statuses of the parking zones are marked alphabetically as “U” if unoccupied and “O” if occupied. These statuses are updated at regular time for near real-time detection. As illustrated in Figure 6.2, the program also logs any changes in the parking state of a given zone to a text file along with a timestamp (date and time) and the characteristics of the parking place (legal/illegal parking place). This is a precursor to writing a full function that writes to a database.

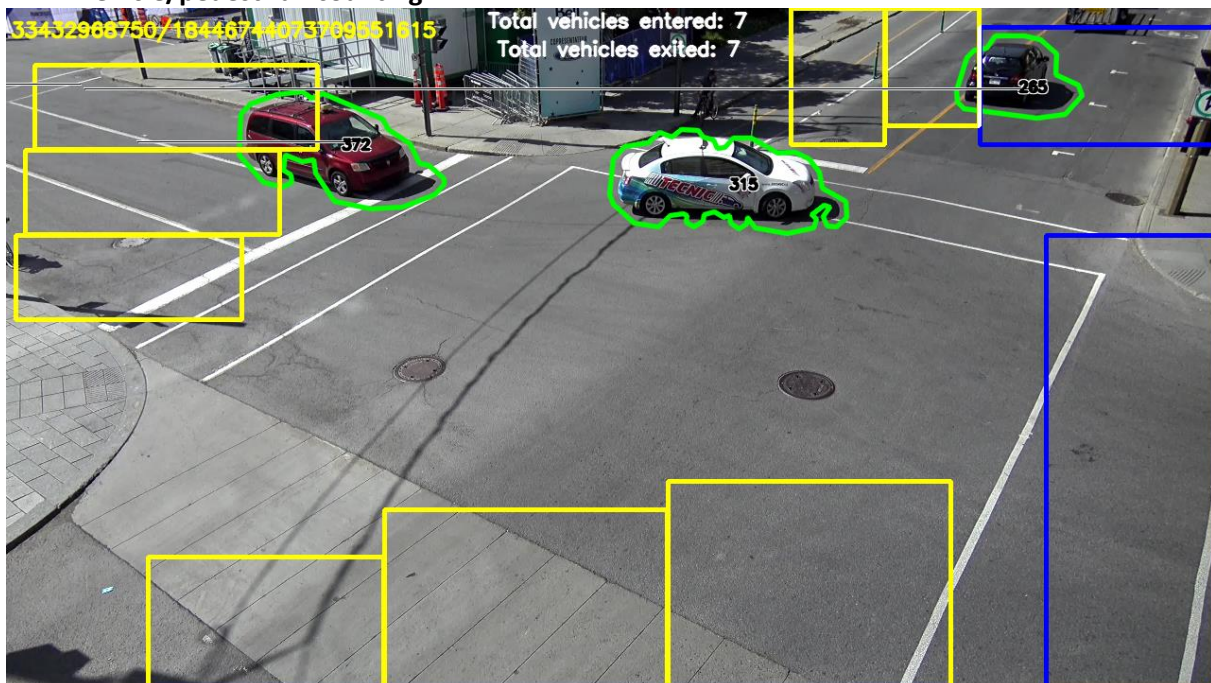
The overall image processing structure of the program is illustrated in Figure 6.3. The program consists of three phases: pre-processing to de-noise and enhance the image, occupancy detection to detect the states of the defined parking zones and result presentation to draw the results to the screen and to the output text file. To enable accurate detection, two image processing algorithms were implemented and combined: the Laplacian detection algorithm and the Luminance-based detection algorithm.



**Figure 6.3:** Parking detection image processing block diagram.

The reliability of the program was tested in typical conditions (from early morning to late evening, with good weather conditions). The testing footages consisted of 14 short clips of an intersection getting from the pilot deployment project with 11 parking spots each. The performance of the parking software is defined based on the number of incorrect detections that last more than a second. The testing results show that the program is able to achieve the **accuracy of 98.2%**. More details of the implementation and more extensive testing results of the program is presented in Appendix H.

#### 6.2.1.2 Vehicle/pedestrian counting



**Figure 6.4:** Vehicle counting application.



The vehicle counting application was also built based on OpenCV library to identify objects of interest (pedestrians or vehicles), track them across the frame and count their number as they enter or exit the intersection. As shown in Figure 6.4, the software allows users to define the entering/exiting lanes in an intersection and is able to track and count the number of vehicles that enters and exit the intersection. In addition, the application also records two types of events to a log file, a precursor to writing this data to a database. Any time a pedestrian or vehicle exits the frame, its point of origin and exit are recorded in order to track not only how many interested objects enter and exit but also the trajectory they follow. Furthermore, at regular, user-defined intervals, the status of all designated entrance/exit zones is also written to the log file.

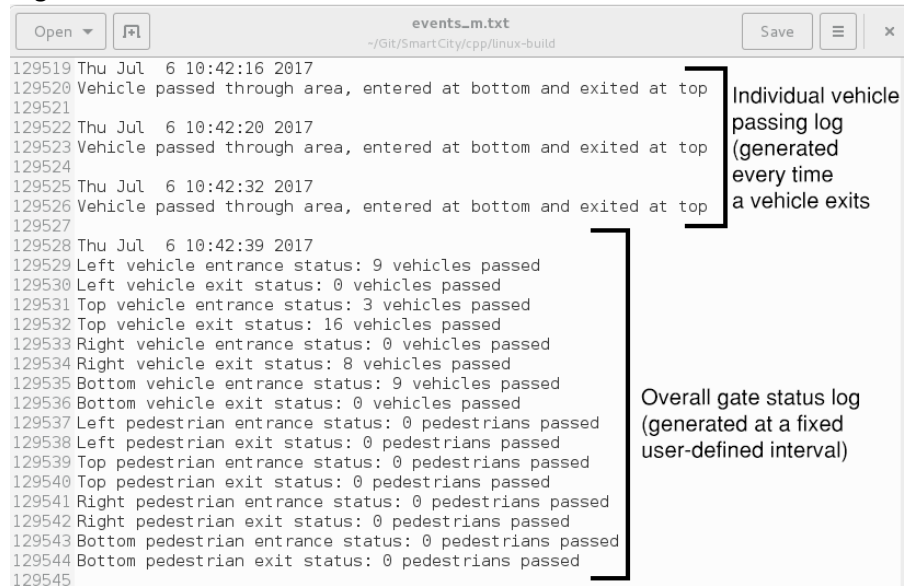


Figure 6.5: Vehicle counting log file.

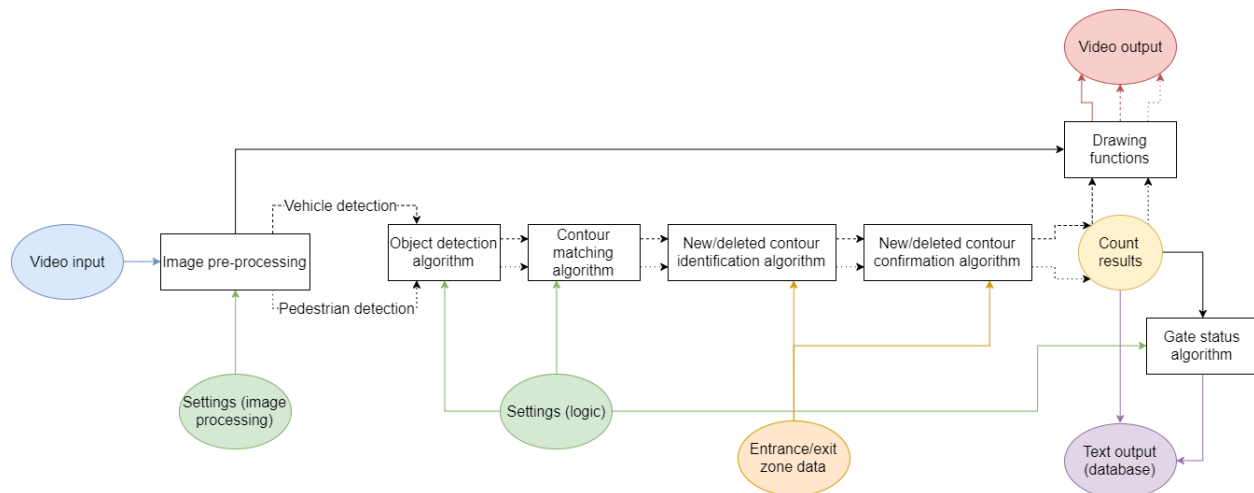


Figure 6.6: Vehicle counting image processing block diagram.

The overall video processing structure of the program is illustrated in Figure 6.6. The program consists of three phases: pre-processing, object detection and tracking, and result output. The pre-processing phase is performed to isolate moving objects and make them easier to identify. This phase outputs two different frames: one for vehicles and one for pedestrians, with different parameters for each to optimize detection for each case. Then, an object detection algorithm detects contours of objects in the frame, and a contour

matching algorithm allows tracking a given object by linking its contours between frames. For counting, a new/deleted contour identification algorithm identifies contours that have possibly entered or exited the frame and marks the entering/exiting locations, and a new/deleted contour confirmation algorithm confirms that the contours have indeed entered/exited the frame and generates the appropriate counts. The count results are combined with image pre-processing results and are displayed as video output to the user. They are also written to a database: some data is written immediately, other, periodically as presented earlier.

The reliability of the program was tested through three five-minute clips in two intersections for vehicle detection and one ten-minute clip for pedestrian detection, all of which are the footages from the pilot deployment in Montreal. All clips were recorded in typical conditions (morning and day with good weather conditions). To quantify the performance of the implemented software, the counting accuracy was measured. The test results show that the counting program is able to achieve the **accuracy of higher than 90%**. More details of the implementation and more extensive testing results of the program is presented in *Appendix H*.

#### **Impact of camera specifications on Parking occupancy and Vehicle/Pedestrian counting application**

In video analytics applications, accuracy of the applied methods can be greatly affected by several factors such as camera setup, resolution, etc. For parking occupancy and vehicle/pedestrian counting applications, comments on the camera placement will be discussed first and then the accuracy of the applications is tested on different resolution and frame rate. Last but not least, the real-time capability of the applications will be also investigated.

- **Camera setup**

#### **Parking occupancy application**



**Figure 6.7:** Example of good camera setup for parking detection application.

For parking occupancy application, camera setups have to satisfy many constraints. The most important factor is the camera angle, it is best to position the camera so that it has an approximately top-down view to avoid perspective and obstruction issues. It should have framing such that every parking spot to be analyzed is fully visible for the best detection. An example of a good camera position can be illustrated in Figure 6.7.

#### **Vehicle counting application**

For applications that focus on the accuracy of vehicle counting, it is best to install one camera on each street and detecting vehicles that pass directly underneath as shown in Figure 6.8. This setup would ease the detection of objects and also require no perspective compensation. However, this setup requires many cameras, each for each of the streets which is costly. Furthermore, since video footage comes from



many different cameras, tracking an object across the whole frame and thus obtaining statistics on the trajectory of the objects is not straight forward.



**Figure 6.8:** Camera looking down a street.



**Figure 6.9:** Top-down view.

If only one camera is available, the best view, a good setup is to look at the entire intersection from the top as shown in Figure 6.9. This setup would solve the problem of vehicles overlapping and being difficult to identify, and should minimize the impact of shadows. It also requires no perspective adjustment, since there is no perspective issue, and the distortion introduced by the fisheye effect is minimal. Furthermore, if both parking and motion detection are to be done with the same camera, this perspective will work well for both. However, mounting the camera directly above the intersection may not always be possible. Also, detecting pedestrians with this perspective will be tricky because the top-down view would make their contour area small and vary significantly.

The next suggestion would be an isometric view, such as shown in Figure 6.10. This setup requires little perspective compensation, supports pedestrian detection and may also be the most feasible camera

perspective setup in an urban area and to use for multiple purposes. However, the accuracy achieved in this setup is expected to be worse than the above two suggestions.



Figure 6.10: Isometric view.

- Resolution and frame rate

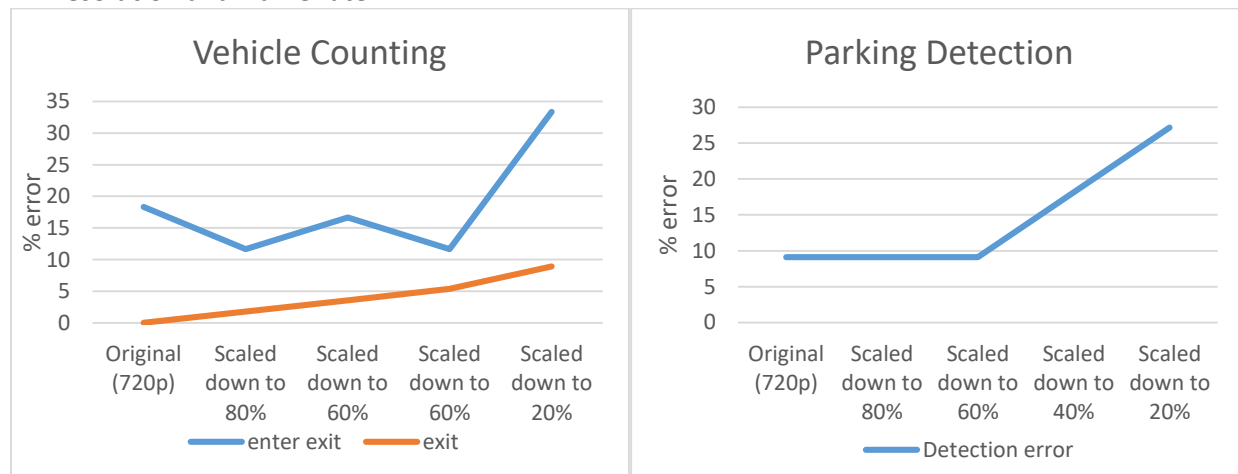
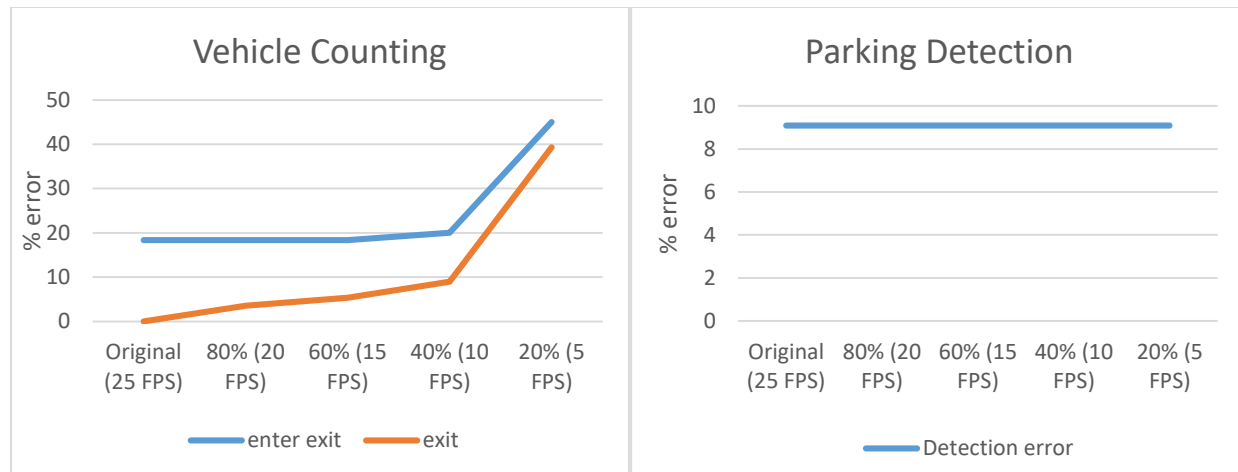


Figure 6.11: Error detection of the two applications at different resolutions.

The impacts of resolution and frame on the accuracy of the two applications are investigated in three recorded videos and representative results are shown in Figure 6.11. It is shown that as the resolution is scaled down, more details are lost and **the error increases with the reduction in resolution**. However, it is interesting to observe that **the parking detection is less sensitive to the resolution and the performance only degrades after aggressive scaled down**.

Regarding the impacts of frame rate on the two applications, representative results are shown as in Figure 6.12. It is observed that **the frame rate does not have a great impact on Vehicle counting application, however, when the frame rate is reduced significantly, it becomes hard to track the objects and the error rate shoots up**. Regarding the parking detection, the results show that the application is **not sensitive to frame rate** as object tracking is not required.

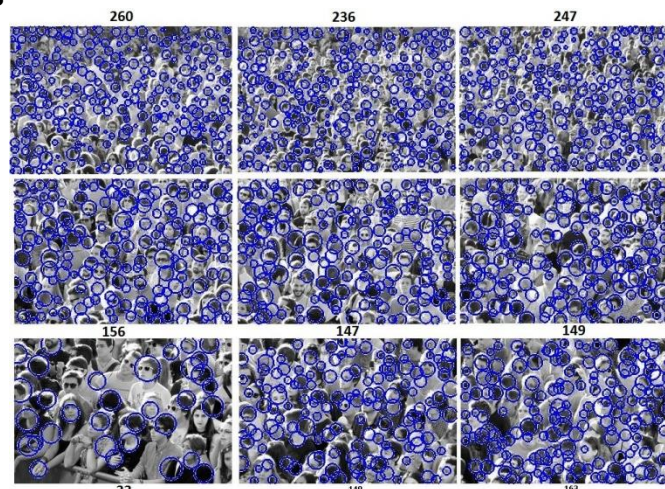


**Figure 6.12:** Error detection of the two applications at different frame rates.

- **Real-time capability**

Real-time capability of the algorithms is also tested on a machine with a 4-core 3.6GHz Intel Core i7-4796 CPU. The results show that **the vehicle counting application can achieve the frame rate of 10FPS while the parking detection application can only achieve up to 4FPS**. This results illustrate that the tested video analytic application are only suitable for near real-time application.

### 6.2.2 Crowd counting



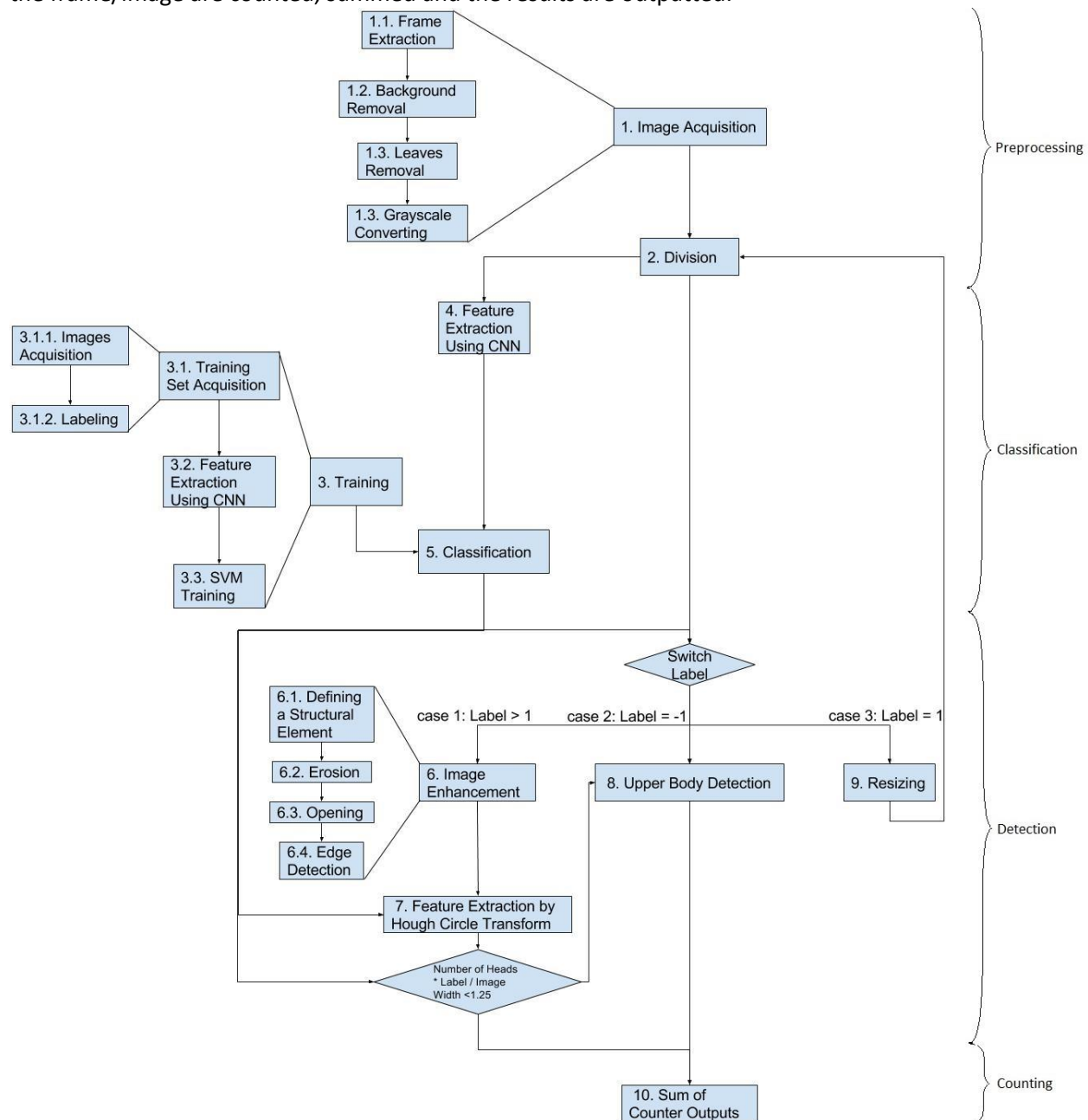
**Figure 6.13:** Heads found by the proposed method.

A crowd counting application was also implemented based on Matlab to estimate crowd size in a distorted image or video. In the application, a new technique is proposed based on a machine learning method which can automatically decide the size of people's heads in different parts of the image. As shown in Figure 6.13, the application divides the image into sections and can adaptively count the number of people's head in the crowd in each of the sections with different distorted perspectives.

The overall processing structure of the program is illustrated in Figure 6.14. In the proposed method, the video/image is first preprocessed to remove the background and leaves to eliminate areas with no people in the frame which can cause false detections in latter steps. Then, the frame is divided into 3 by 3 small patches. For each small patch, AlexNet [90], a trained machine learning model, is used to obtain a range of head size in that patch. Depending on the result head sizes, different strategies will be applied. For heads with medium size, an image processing method based on circle detection is used to detect the



heads. When the heads are large, upper body detection (detects heads and shoulders) is used instead for improved accuracy. If the heads are small, the image patch is scaled up by 3 and the patch is recursively processed again (divide, detect head size, detect heads/upper body). The found heads in each patch of the frame/image are counted, summed and the results are outputted.



**Figure 6.14:** Block diagram of the proposed method.

The proposed method were tested with 4 images (3 images of very crowded scenarios and one image is of a less crowded scene), and 2 videos (one of a very crowded scene and another of a less crowded one). All of the images/videos are at HD resolution (1280x720). The results are compared with the number of people in those images which is counted manually. The algorithm is also compared with other 3 methods in the literature. The results show that the proposed algorithm performs quite well with **errors around 20%**, outperforms the other methods in the crowded scenarios with. However, in less crowded case, other

small objects may be misinterpreted as heads which leads to large error rate. More details of the implementation and more extensive testing results of the program is presented in Appendix I.

#### Impact of camera specifications on Crowd counting

In video analytics applications, accuracy of the applied methods can be greatly affected by several factors such as camera setup, resolution, etc. For crowd counting application, comments on the camera placement will be discussed first and then the accuracy of the application is tested on different resolution. Last but not least, the real-time capability of the application will be also investigated.

##### • Camera setup

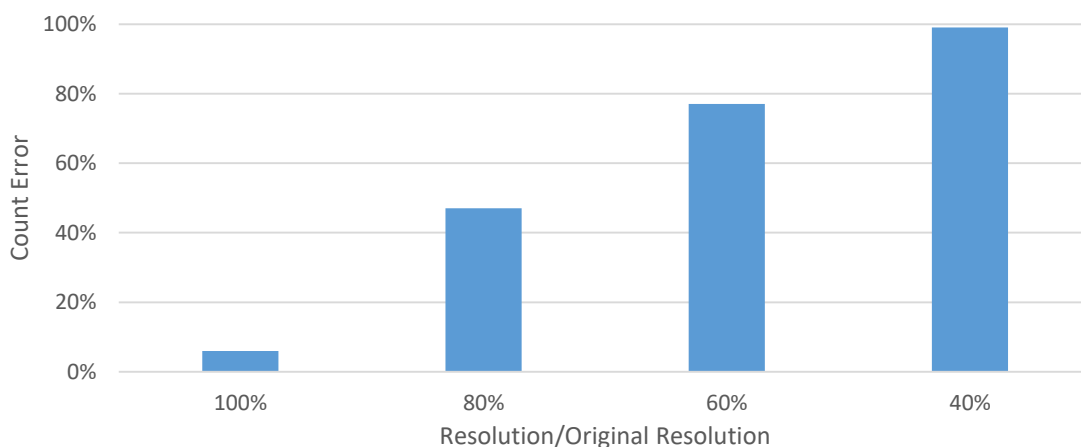


**Figure 6.15:** Expected camera setups (left) and camera setup with too high inclination (right)

For crowd counting application, the camera should be installed above people's heads and shoot down (not necessarily vertical) on the crowd scene. However too high inclination can cause a very low accuracy due to obstructions. Figure 6.15 illustrates both cases when the image taken by camera with expected installation and with the inclination too high.

##### • Resolution

Another important factor is the resolution of the image. A lower resolution image/video can lead to smaller heads, which makes the detection more difficult. When the heads are really small, false detection tends to increase with lower resolution image/video as the heads may not be recognizable. The impact of resolution was tested with the crowd estimation method and the result shows that the **error increases to 47% when the image is scaled down to 80%** and 99% when the image is scaled down to 40% of the original resolution as shown in Figure 6.16.



**Figure 6.16:** Crowd count error percentage of the proposed algorithm.

##### • Real-time capability

Real-time capability of the algorithm is also tested on a machine with a 4-core 3.6GHz Intel Core i7-4796 CPU. The results show that the proposed algorithm takes about **32 seconds to process 1 image and about 35 seconds for a video frame**. The results show that the tested video analytic algorithm are not suitable for real-time application.

### 6.3 Challenges

As shown above, data analytics have a vast potential with useful applications and will be one of the key component of a Smart City, constantly making new meanings from the raw data. However, the deployment of data analysis is still slow due to many challenges in the design, development and deployment. These difficulties includes the availability of necessary tools, real-time requirement, and the representation, accuracy of the collected data, as well as the cost and accessibility of data. Some of the key challenges to the deployment of data analysis are presented as below.

- **Data quality and formatting.** In a Smart City, data are generated from a wide variety of sources with different characteristics and formats. Regarding the formatting, new data formats are being introduced every day which may be greatly different or even incompatible with existing formats. Some of the data are even unstructured such as images, videos, server logs etc. In order to best utilize these data, a proper classification and structuring of data formats is needed. In addition, a Smart City can be implemented in different stages by different teams which can cause inconsistency in data structure. These issues make it difficult to retrieve and digest the data automatically for useful analysis.
- **Data quality.** Although the analytic applications are making progress every day, they still requires some proper setups with the sensor to get the best performance for examples, camera setup may work well in some setup while getting a lot of errors with other setup. Since the setup requirements may different from one application from another may not be the same, raw data from a single setup may not have the same quality for all the analytic applications but deploy dedicated sensors for each applications may be very costly. As a result, there is a trade-off between the cost and the data quality collected.
- **Centralized vs. distributed data analysis.** Due to the vast amount of data generated at the edge of the network, where sensors reside, the traditional approach of transporting all of these data to a centralized data center may not be suitable [87]. Moreover, many applications require stringent latency for reactions in terms of ms which makes it infeasible to realize such applications with the traditional centralized data analysis. To accommodate these new applications and data requirements, data analysis functions should be placed near the edge of the network which gives rise to the distributed data analysis mechanisms. As a result, one of the most fundamental questions for storing and processing data in the coming decade is on where, when and how to distribute the computation, communication, control and storage.
- **Real-time analytics.** Support for real-time or near real-time analytics is important in public safety and transportation Smart City applications, especially for those applications that deal with videos. With the currently available algorithms, the trade-off between accuracy and processing time has to be considered before the implementation. As a result, future studies are required to propose new innovative methods of data analytics that can both improve the accuracy and reduce the complexity.
- **Data and information sharing.** Sharing of information enables the collaboration and interconnectivity but it also imposes many challenges. Since each government entities has its own confidential information, sharing the data may not be easy without compromising the privacy conditions. In some cases, preprocessing of data before sharing should be done to prevent access to the raw data. For instance, a count number of traffic passing a cross section may need to be done and return to a request for data instead of allowing direct access to the raw video footage.



## 6.4 Conclusion

This chapter investigates the data analytic aspects for Smart City's applications. First, the general introduction to data analytic is provided to show the motivations, benefits, and some representative applications data analytics in a Smart City context. Then, two illustrative example applications which were implemented during the project were briefly described, including an application for parking lot monitoring and vehicle/pedestrian counting, and an application for crowd counting. The overall structures, outputs and results of these applications were also presented. The chapter closes with the key challenges of implementing the data analytic applications.

# Chapter 7

## Conclusions and Recommendations

This chapter provides a brief summary of the project on Smart City pilot deployment in Montreal. Then, recommendations for further developments in the next steps will be also outlined. For ease of understanding, this chapter is structured according the layer architecture in Chapter 2 and in each layer, discussion will be given on what have been achieved in this project, what are the possible extension/expansion in the next steps and what needs to be further studied. Since Security spans across all layers, a subsection is dedicated for discussion on Security. Table 7.1 summarizes of the main points presented in this chapter.

**Table 7.1:** Summary of the project and recommendations on what can be done next and be further studied.

<b>Data Acquisition Layer:</b>	
Work done:	<ul style="list-style-type: none"> <li>✓ Demonstrate the feasibility of integrating different types of sensors, devices and technologies to the Smart City infrastructure.               <ul style="list-style-type: none"> <li>• Stationary sensors: cameras, traffic radar sensors, RFID readers.</li> <li>• Mobile sensors mounted on moving vehicles: temperature, GPS, level sensors, 3D and high speed cameras.</li> <li>• Communication devices: WiFi bridges, several types of adapters.</li> </ul> </li> <li>✓ Test of different PHY/MAC protocols and topologies.</li> <li>✓ Demonstrate the feasibility of all-IP network support.</li> </ul>
Possible extension/expansion:	<ul style="list-style-type: none"> <li>➤ Integrate more types of IoT sensors:               <ul style="list-style-type: none"> <li>• Public services: trash can, fire hydrant</li> <li>• Environment monitor: humidity, noise, air/water quality sensors</li> <li>• Public safety: emergency button</li> <li>• Transportation: smart parking meter, vehicle tracking using RFID</li> </ul> </li> <li>➤ Integrate large-scale wireless sensor networks</li> <li>➤ Integrate actuators for automatic response/ remote control</li> <li>➤ Implementation of gateway devices for interfacing with limited feature devices.</li> <li>➤ Investigate the possibility of using dedicated spectrum for Smart City Wireless communications.</li> </ul>
R&D topics:	<ul style="list-style-type: none"> <li>❖ Intelligent network access mechanisms for highly dense sensor networks.</li> <li>❖ Minimum-interference frequency allocation and routing techniques.</li> </ul>

<b>Connection Layer:</b>	
Work done:	<ul style="list-style-type: none"> <li>✓ Investigated the feasibility and implementation of data connection through Fiber and LTE networks</li> <li>✓ Studied network planning estimation for downtown &amp; Montreal areas.</li> </ul>
Possible extension/expansion:	<ul style="list-style-type: none"> <li>➤ Deploy/expand a larger Smart-City/IoT network to have a more accurate network planning estimation.</li> <li>➤ Investigate the possibility of uploading data on Montreal Public WiFi network.</li> </ul>
R&D topics:	<ul style="list-style-type: none"> <li>❖ Distributed/hierarchical structure to accommodate the massive amount of data for a <i>full-scale</i> Smart City, e.g., broadband mmW communications</li> </ul>
<b>Management Layer:</b>	
Work done:	<ul style="list-style-type: none"> <li>✓ Implemented a prototype management platform based on both on-premise and cloud-based approaches for <ul style="list-style-type: none"> <li>• data collection and storage (data plane), and</li> <li>• control and monitoring (control plane)</li> </ul> </li> </ul>
Possible extension/expansion:	<ul style="list-style-type: none"> <li>➤ Expand the functionalities of <ul style="list-style-type: none"> <li>• Data Plane management: real-time monitoring, alarm system design, more interactive GUI.</li> <li>• Control Plane management: real-time log display and filter, more control functions: view device health (bandwidth, CPU, RAM usage), system health check, reset interface..., alarm system design... <ul style="list-style-type: none"> <li>○ Access control management</li> </ul> </li> </ul> </li> </ul>
R&D topics:	<ul style="list-style-type: none"> <li>❖ re-configurable structure for mass number of devices: device settings, mass firmware update.</li> <li>❖ Virtualized architecture for Smart-City IoT network to support multiple service providers</li> </ul>
<b>Application Layer:</b>	
Work done:	<ul style="list-style-type: none"> <li>✓ Implemented a scaled-down data center in BCRL at McGill.</li> <li>✓ Developed a demo application for street parking space monitoring and vehicle/pedestrian counting.</li> <li>✓ Developed a demo application for crowd counting.</li> </ul>
Possible extension/expansion:	<ul style="list-style-type: none"> <li>➤ Investigate/develop more data analytic functions: <ul style="list-style-type: none"> <li>• Data correlation</li> <li>• Automatic reactions based on collected data</li> <li>• More computer vision applications: fire, injury, accident detection...</li> </ul> </li> </ul>
R&D topics:	<ul style="list-style-type: none"> <li>❖ Distributed structures for signal processing and data analytic</li> <li>❖ Context-aware intelligent services</li> <li>❖ Real-time analytics</li> <li>❖ Data sharing structures</li> </ul>
<b>Security Issues:</b>	
Work done:	<ul style="list-style-type: none"> <li>✓ Implemented VPN for securing data transit through public networks</li> <li>✓ Investigated Security features of the currently deployed devices</li> </ul>
Possible extension/expansion:	<ul style="list-style-type: none"> <li>➤ Define &amp; implement privileges for database, device and management platform access</li> <li>➤ Secure data transfer from the sensors</li> </ul>
R&D topics:	<ul style="list-style-type: none"> <li>❖ Hardware and firmware security structure for IoT smart sensors</li> <li>❖ Secure reconfigurations of IoT Smart sensors.</li> <li>❖ Data Analytics and Artificial Intelligent (AI)/Machine-Learning – based Security</li> </ul>

## 7.1 Data Acquisition Layer

**Work done:** In this current project, the feasibility of integrating different types of sensors, devices and technologies to the Smart City infrastructure has been demonstrated. A wide range of devices have been deployed and tested in the pilot deployment:

- *Stationary sensors:* include several types of cameras, traffic radar sensors, RFID readers.
- *Mobile sensors:* Temperature, GPS, level sensors, were mounted on a salt trucks to monitor the ambient and the road surface temperatures, the trajectory of the vehicle and the salt level in the container. In addition, the second vehicle were mounted with high speed and 3D cameras for road surface and tree inspection purpose.
- *Communication devices:* include WiFi bridges and several types of adapters.

For wireless data transport, several PHY/MAC protocols and topologies were implemented and tested including Point-to-Point and Point-to-Multipoint/Star topologies as well as an enhanced media access protocol based on TDMA on WiFi Technologies. In the pilot project, several types of gateways were deployed to support the integration of non-IP sensors to the all-IP network.

**Possible extension/expansion:** For the next steps, the *integration of several other types of IoT sensors* can be considered. For public services, installing sensors on public assets such as trash cans and fire hydrant can helps to optimize the service and transportation cost to the city. Besides, a wide varieties of environment sensors, which are not considered in the pilot project, can be investigated such as humidity, noise, and air/water quality sensors. The data collected from these sensors can provide valuable insights about other important environmental aspects of the city. For public safety, deployment of emergency buttons could be useful in the case of injuries, accidents or robberies on the street. For transportation, sensors such as RFID can be used to track the trajectory of stolen vehicles around the city; besides, IoT-enabled smart parking system is also a very promising application for a crowded city like Montreal. Moreover, in this project actuators were not considered for deployment, which should be investigated in the next steps to provide the possibilities of automated response or remote control.

In addition, the *integration of large-scale wireless sensor networks* can be a cost-effective solution to collect low-rate data from a wide area like a City and should be investigated in the next steps. Besides, the implementation of gateway devices to interface the network of limited feature devices with the all-IP network could be studied. Furthermore, long-range low-power communication technologies such as LoRaWAN and SigFox have not been investigated in this project. These technologies promise very long range communication for a dense network of several thousands of devices with a simple star topology which is very suitable for applications in a city-wide scenario.

Another important theme to be studied is the possibility of *using a dedicated spectrum* for Smart City wireless communications. As the public WiFi is overcrowded with interference, uncontrollable and unpredictable with time, dedicate a spectrum for Smart City communications is needed to guarantee the availability of the infrastructure.

**R&D topics:** In the context of a Smart City where highly dense networks with a massive number of devices deployed, since many nodes want to use the same shared medium, channel access is likely a potential system bottleneck. The issue becomes more complex when the connected devices have different characteristics in terms of traffic demands, power demands and quality of services. In this context, an *intelligent and efficient media access protocol* has to be designed to adapt to the dynamicity of the network with nodes joining and leaving at any time but still can maintain a high throughput and fairness

in the network. As many of the nodes in the network are battery-operated, the protocol has to be also energy-efficient and low-complexity.

In addition, while wireless medium offers the flexibility for placement of devices, having many of them deployed in a close vicinity makes interference a serious problem. Too much interference can be the main factor that limits the overall network capacity. In this case, traditional adhoc routing protocols is not sufficient and a research of *adaptive frequency allocation and routing techniques to minimize the interference effect* is needed.

## 7.2 Connection/Communication Layer:

**Work done:** Regarding the connection layer, the current project investigated the feasibility of communications between the on the field devices and the management platform using fiber networks and LTE cellular network. Various types of devices were integrated into the network that have data collected through these two communication infrastructure. It is shown that for the stationary devices and high throughput sensors (such as cameras), fiber connections are the best choice due to their high bandwidth and stability. For mobile devices, LTE can be a very good option to consider due to its wide coverage. However, due to its moderate throughput and high cost, it is only suitable for low capacity applications.

In addition, in this project, a studied has been conducted to estimate the network planning for downtown of Montreal and then expanded to the whole Montreal area. The study is based on simulation tools since the exact fiber drop location is not available. The study shows the estimated bandwidth requirements for different setups and deployment areas which also highlight the scalability issue in terms of bandwidth and hence storage.

**Possible extension/expansion:** This pilot deployment is in a small scale with only 6 cameras and roughly 50 devices in total. It is not sufficient for an accurate estimation for a full-scale Smart City deployment as well as foreseeing all the possible issues that can emerge. In the next step, *a larger scale deployment* is required in order to gain more insights about the collected data, the possible applications and their impacts on the city residents as well as a more accurate planning estimation.

Moreover, as the *city public WiFi* is already available, it is interesting to investigate the possibility of conducting communications through this network instead of building a separate infrastructure for data transport. This approach may save the device costs, however, as user usage fluctuates during a day, the availability of the Smart City devices may be affected and should be investigated.

**R&D topics:** For connection layer, the main issue is scalability. With the sheer amount of devices in a Smart City, the volume of data is expected to be massive. As illustrated in Chapter 3, the amount of data for a full-scale Montreal Smart City could be of Tbps. The question is how to transport this massive amount of data? One approach to solve the problem is to have *distributed data centers* around the city, each accommodate data from a (geographic) region. However, to design and coordinate such distributed/hierarchical structure is not straight forward and needs to be studied carefully to be cost-effective and scalable. Furthermore, use of broadband millimeter-wave (mmW) and advanced wireless communications is promising to provide the required capacity.

### 7.3 Management Layer

**Work done:** In the MSCPS, a prototype management platform for data collection and storage was implemented to demonstrate the feasibility of a centralized management platform on the data plane. The prototype investigated both the on-premises and cloud-based approaches to show the benefits and drawbacks in each case. Likewise, a prototype management platform for device control and monitoring was also implemented, exploring both on-premise and cloud-based approaches. In the device control and monitoring aspects, two most important functions were implemented: device availability check and remote reboot functions.

**Possible extension/expansion:** The prototype management platform is a primitive platform to demonstrate the concept. Significant improvements need to be involved before the platform can be used in production. On the data plane management, several improvements can be done, including real-time monitoring, alarm system design, more interactive GUI. Likewise, on the control plane management, functions such as real-time log display and filter, device health status report (bandwidth usage, CPU, RAM utilization), system health check procedure, alarm system design, reset procedure for parts of the device (for instance wireless interface)... should be implemented.

Moreover, access control management should be developed to define what application can access to what type of data or control what type of devices. It is important to reckon that data and device access can change flexibly depends on the usage context such as location, time, social events... rather than a set of fixed rules, which needs to be taken into the design process.

**R&D topics:** With so many devices in the field, it is impossible to reconfigure the devices manually. Instead, an *automated reconfiguration process* should be developed to fulfil this task. However, with so many device types from many manufacturers co-exist in the same network, functions such as mass firmware update is not straight forward and research should be conducted to find a systematic and scalable approach.

When fully deployed, the Smart-City infrastructure has to be designed to be used as a public platform or to be shared for various parties such as police, fire fighter departments, hospital, government as well as private companies. On one hand, this design approach maximizes the benefits of this Smart-City infrastructure. On the other hand, deploying separate infrastructures for additional applications is cost-ineffective, redundant and to some extents would limit the operations of the existing network due to interference and bandwidth contention. Moreover, in Smart City, it is not possible to predict all the future applications to be able to provision their support at the time of deployment. As the result, the concept of *virtualization* is brought into the picture to provide the abstraction of the physical resources so that different applications (or service providers) can share the same physical infrastructure for added values. However, the realization of such idea is not easy as it requires complex design and communications both at the physical layer and the application layer for *virtualized* Smart-City infrastructure.

### 7.4 Application Layer

**Work done:** In the MSCPS, a scaled down data center is implemented in the BCRL at McGill to use as a platform for data collection and testing of analytic applications. To this end, a demo application was developed based on an open source library (OpenCV) to monitor the street parking space and to count vehicle/pedestrian at intersections. In addition, another application was implemented for crowd counting. Both the applications are able to achieve very good accuracy, demonstrating the capability of the current hardware in video processing applications.



**Possible extension/expansion:** The studies done in the MSCPS only scratch the surface of the possibilities of using data at the application layer. First, more data analytic functions should be developed in order to extract the meaningful information and useful trends from the raw data. To achieve this goal, processing only one source of data is not enough and a *data correlation* function that grinds data from many different types of sensed data should be implemented. For example, to conclude how good the air quality or the water quality is, measurements from several sensing devices should be incorporated. The data correlation also helps to reduce false alarms due to noisy readings or device malfunctions.

Moreover, automation process needs to be developed so that the system can *react automatically based on the collected data*. Such intelligent service are what makes up the smartness of a Smart City, reducing the human intervention in the city operations and improving the quality of the citizens.

In terms of *computer vision applications*, there is still a lot of information that can be extracted from the camera feeds and hence, the potential of this kind of application is huge. In the next steps, many useful computer vision analytic functions should be explored including fire, injury, accident detection, etc. These application can benefit the public safety as well as improve the lives of Montreal residents in a great way.

**R&D topics:** Despite the progress in data analytics, many open issues remain unexplored. First, due to the vast amount of data generated at the edge of the network, where sensors reside, the traditional approach of transporting all of these data to a centralized data center may not be suitable or even possible due to the bandwidth limitation. Moreover, many applications require stringent latency for reactions in terms of ms which makes it infeasible to realize such applications with the traditional centralized data analysis. To accommodate these new applications and data requirements, *data analysis functions should be placed near the edge of the network* which gives rise to the distributed data analysis mechanisms. As a result, one of the most fundamental questions for storing and processing data is on where, when and how to distribute the computation, communication, control and storage

Moreover, as a Smart City spans across a vast geographic area, providing services or acting on the collected data in the same way may not be the best way. Instead, a *context-aware oriented approach* may be more suitable. In this scheme, the data are actioned upon depending on their location, time of the day, day in the year or even the environmental status surrounding the place. For instance, a street display should show information about interesting places in the vicinity of its placement or show warnings/recommendations based on the local environmental data (air pollution, high traffic, and high level of UV).

*Support for real-time or near real-time analytics* is important in public safety and transportation Smart City applications, especially for those applications that deal with videos. With the currently available algorithms, the trade-off between accuracy and processing time has to be considered before the implementation. As a result, future studies are required to propose new innovative methods of data analytics that can both improve the accuracy and reduce the complexity.

*Sharing of information* enables the collaboration and interconnectivity but it also imposes many challenges. Since each government entities has its own confidential information, sharing the data may not be easy without compromising the privacy conditions. In some cases, preprocessing of data before sharing should be done to prevent access to the raw data. For instance, a count number of traffic passing a cross section may need to be done and return to a request for data instead of allowing direct access to the raw video footage.

## 7.5 Security Issues

**Work done:** In this pilot project, data security is only based on commercially available features that are supported on the deployed devices. In particular, at data acquisition layer, data is protected by encryption

at link layer when transmitting via wireless bridges. At communication layer, data is protected by VPN technology between the gateway at the deployed area and the data center. In addition, security features of the currently deployed devices on the control plane are also surveyed and implemented.

**Possible extension/expansion:** In the MSCPS, as devices deployed in place and the data collected into database, it is important to define security policies to prevent security breaches. These policies should clearly define the privileges for database, device and management platform access. Many levels of access should be defined such as data monitoring only, device status monitoring only, device configuration change, etc. Besides, securing the integrity of device logs is also important to investigate misbehaviors, and track incidents.

Currently, data security is merely based on link layer encryption. Data security can be greatly enhanced by bringing the authentication and encryption features close to the sensors. It is acknowledged that many of the currently available sensors do not support data security; however, secure transport of data can be achieved by implement device gateways which are in charge of collecting the data from the sensors and then secure the data by cryptographic techniques when communicating with the upper layer.

**R&D topics:** Most of the standards in IoT focus on the communication aspects for the data transport in IoT network, leaving an unexplored area of research in *physical security* of IoT devices. In the context of Smart City, being exposed to the environment, IoT devices have a significant disadvantage of lacking of physical security. Having physical access to the device, an adversary can extract the security materials or the stored data on the device as well as gaining insights on the detail implementation for further exploit and penetration. Therefore, solutions for protecting the IoT sensors at the physical level still needs some further developments.

*Secure configuration/re-configuration* of IoT sensors is another important issue. During the life cycle of an IoT device, it is needed to be reconfigured and updated regularly to catch up with the functionality requirements and to apply security patches. Due to the numerous devices in a large scale deployment, automation of the update process is mandatory. An adversary can exploit this batch operation to inject malicious modifications or Trojan into the devices' firmware. As a result, a secure mechanism for reconfiguration of the IoT sensors is of top priority before any large-scale IoT deployment is feasible.

Security is a very dynamic domain, new threats and types of attacks are evolving every day. To cope up with those attacks that are not yet documented, new mechanisms of security should be developed using *data analytics and AI/machine-learning* to quickly identify the new attack and come up with a counter measure to protect the system and its data.

# References

- [1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey", *Computer Networks*, Vol. 54(15), pp. 2787-2805, 2010.
- [2] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities", *IEEE Internet of Things Journal*, Vol. 1 (1), pp. 22-32, 2014.
- [3] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications", *IEEE Internet of Things Journal*, Vol. PP (99), pp. 1-17, 2017.
- [4] C. Yin, Z. Xiong, H. Chen, J. Wang, D. Cooper, "A literature survey on smart cities", *Science China Information Sciences*, Vol. 58 (10), pp. 1-18, 2015.
- [5] S. Pellicer, G. Santa, A. L. Bleda, R. Maestre, A. J. Jara, et al. "A Global Perspective of Smart Cities: A Survey". *IEEE Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 439–444, 2013.
- [6] S. Madakam and R. Ramachandran, "Barcelona Smart City: The Heaven on Earth (Internet of Things: Technological God)", *ZTE Communications*, Vol. 13 (4), pp. 3–9., 2015.
- [7] P. Liu and Z. Peng, "China's Smart City Pilots: A Progress Report", *Computer*, Vol. 47 (10), pp. 72–81, 2014.
- [8] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, "SmartSantander: IoT experimentation over a smart city testbed", *Computer Networks*, Vol. 61, pp. 217–238, 2014.
- [9] J. Gutiérrez Bayo, "International Case Studies of Smart Cities: Santander, Spain", *Inter-American Development Bank*, 2016.
- [10] A. Gharaibeh, M. A. Salahuddin, S. J. Hussini, A. Khreishah, I. Khalil, "Smart Cities: A Survey on Data Management, Security and Enabling Technologies", *IEEE Communications Surveys & Tutorials*, Vol. PP (99), 2017.
- [11] S. Talari, M. Shafie-khah, P. Siano, V. Loia, A. Tommasetti, "A Review of Smart Cities Based on the Internet of Things Concept", *Energies*, Vol. 10 (4), 2017.
- [12] S. Mitchell, N. Villa, M. Stewart-Weeks, and A. Lange. "The Internet of Everything for Cities: Connecting people, Process, Data, and Things to Improve the 'Livability' of Cities and Communities". *Cisco*, pp. 1–21, 2013.
- [13] GSMA – "Guide to Smart Cities: The Opportunity for Mobile Operators, GSMA Smart Cities"
- [14] M. Khan, "Las Vegas Valley Water District using IoT technology for water infrastructure management" – Available online: <https://www.iot-now.com/2015/07/08/34535-las-vegas-valley-water-district-using-iot-technology-for-water-infrastructure-management-and-leak-detection/>
- [15] M. Bouskela, "The Road toward Smart Cities: Migrating from Traditional City Management to the Smart City", *Inter-American Development Bank*, 2016.
- [16] J. Frazier, T. Touchet, "Transforming the City of New York", *Cisco*, 2012.
- [17] J.-S. Hwang and Y. H. Choe, "Smart Cities Seoul: a case study", *ITU-T Technology Watch Report*, pp. 1–20, 2013.
- [18] D. Amar Flo'rez "International Case Studies of Smart Cities: Medellin, Colombia", *Inter-American Development Bank*, 2016.
- [19] Cisco White Paper - "The Internet of Things Reference Model", Cisco Press, 2014.

- 
- [20] F. Gil-Castineira, E. Costa-Montenegro, F. Gonzalez-Castano, C. López-Bravo, T. Ojala, "Experiences inside the Ubiquitous Oulu Smart City", *Computer*, Vol. 44 (6), pp. 48-55, 2011.
  - [21] V. Kostakos, T. Ojala, and T. Juntunen, "Traffic in the Smart City: Exploring City-Wide Sensing for Traffic Control Center Augmentation", *IEEE Internet Computing*, Vol. 17 (6), pp. 22-29, 2013.
  - [22] T. Bakc, E. Almirall, and J. Wareham. "A Smart City Initiative: the Case of Barcelona", *Journal of the Knowledge Economy*, Vol. 4 (2), pp. 135-148, 2013.
  - [23] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, "SmartSantander: IoT experimentation over a smart city testbed", *Computer Networks*, Vol. 61, pp. 217-238, 2014.
  - [24] S. Latre, P. Leroux, T. Coenen, B. Braem, P. Ballon, "City of things: An integrated and multi-technology testbed for IoT smart city experiments", *IEEE International Smart Cities Conference*, pp. 1-8, 2016.
  - [25] Miao Wu, Ting-Jie Lu, Fei-Yang Ling, Jing Sun, and Hui-Ying Du, "Research on the architecture of Internet of Things". *International Conference on Advanced Computer Theory and Engineering (ICACTE)*, pp. 484-487, 2010.
  - [26] M. Presser and A. Gluhak, "The internet of things: Connecting the real world with the digital world". *EURESCOM The Magazine for Telecom Insiders 2*, 2009.
  - [27] I. Andrea, C. Chrysostomou, and G. Hadjichristo, "Internet of Things: Security vulnerabilities and challenges", *IEEE Symposium on Computers and Communication (ISCC)*, 2015.
  - [28] F.-J. Wu, Y.-F. Kao, and Y.-C. Tseng, "From wireless sensor networks towards cyber physical systems", *Pervasive and Mobile Computing*, Vol. 7(4), pp. 397-413, 2011.
  - [29] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications", *IEEE Communications Surveys & Tutorials*, Vol. 17(4), pp. 2347-2376, 2015.
  - [30] H. B. Pandya and T. A. Champaneria, "Internet of things: Survey and case studies", *International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO)*, pp. 1-6, 2015.
  - [31] I. Yaqoob, I. A. T. Hashem, Y. Mehmood, A. Gani, S. Mokhtar, et al. "Enabling Communication Technologies for Smart Cities", *IEEE Communications Magazine*, Vol. 55 (1), pp. 112-120, 2017.
  - [32] A. Kandhalu, A. Rowe, R. Rajkumar, C. Huang, and C.-C. Yeh, "Real-Time Video Surveillance over IEEE 802.11 Mesh Networks", *IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 205-214, 2009.
  - [33] R. Wenge, X. Zhang, C. Dave, L. Chao, and S. Hao, "Smart city architecture: A technology guide for implementation and design challenges", *China Communications*, Vol. 11 (3), pp. 56-69, 2014.
  - [34] M. Cesana and A. E. C. Redondi, "IoT Communication Technologies for Smart Cities", *Designing, Developing, and Facilitating Smart Cities*. Cham: Springer International Publishing, pp. 139-162, 2017.
  - [35] M. Díaz, C. Martín, and B. Rubio, "State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing", *Journal of Network and Computer Applications*, Vol. 67, pp. 99-117, 2016.
  - [36] M. Aazam, I. Khan, A. A. Alsaar, and E.-N. Huh, "Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved", *International Bhurban Conference on Applied Sciences & Technology (IBCAST)*, pp. 414-419, 2014.
  - [37] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "Integration of Cloud computing and Internet of Things: A survey", *Future Generation Computer Systems*, Vol. 56, pp. 684-700, 2016.

- 
- [38] Amazon Web Services. Accessed 2017. 2006. url: <https://aws.amazon.com/>
- [39] IBM Bluemix. Accessed 2017. 2014. url: <https://www.ibm.com/cloud-computing/bluemix/>
- [40] Microsoft Azure. Accessed 2017. 2014. url: <https://azure.microsoft.com/>
- [41] S. Bandyopadhyay, M. Sengupta, S. Maiti, and S. Dutta, "A Survey of Middleware for Internet of Things". *Recent Trends in Wireless and Mobile Networks*, pp. 288-296, 2011.
- [42] M. A. Chaqfeh and N. Mohamed, "Challenges in middleware solutions for the internet of things". *International Conference on Collaboration Technologies and Systems (CTS)*, pp. 21-26, 2012.
- [43] A. H. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and M. Z. Sheng, "IoT Middleware: A Survey on Issues and Enabling technologies", *IEEE Internet of Things Journal*, 2016.
- [44] HYDRA. 2010. url: <http://hydramiddleware.eu>
- [45] Global Sensor Networks. 2004. url: <http://lsir.epfl.ch/research/current/gsn>
- [46] Google Fit. 2015. url: <https://developers.google.com/fit/>
- [47] Xively. Accessed 2017. 2014. url: <http://xively.com>
- [48] P. Persson and O. Angelsmark, "Calvin Merging Cloud and IoT", *Procedia Computer Science*, Vol. 52, pp. 210-217, 2015.
- [49] Node-RED, A Visual Tool for Wiring the Internet of Things. url: <https://github.com/node-red/node-red>.
- [50] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions", *Future Generation Computer Systems*. Vol. 29 (7), pp. 1645-1660, 2013.
- [51] R. Roman, P. Najera, and J. Lopez, "Securing the Internet of Things", *Computer*, Vol. 44 (9), pp. 51-58, 2011.
- [52] Gartner press release – <http://www.gartner.com/newsroom/id/3598917>
- [53] J. Singh, T. Pasquier, J. Bacon, H. Ko, D. Eysers, "Twenty Security Considerations for Cloud-Supported Internet of Things", *IEEE Internet of Things Journal*, Vol. 3 (3), 2016.
- [54] J. Zhou, Z. Cao, X. Dong, A. V. Vasilakos, "Security and Privacy for Cloud-Based IoT: Challenges, Countermeasures, and Future Directions", *IEEE Communications Magazine*, 2017.
- [55] S. Ziegler, C. Crettaz, L. Ladid, S. Keco, B. Proric, A. F. Skarmeta, A. Jara, W. Kaster, M. Jung, "IoT6 – Moving to An IPv6-based Future IoT", in Galis A., Gavras A. (eds) *The Future Internet. FIA 2013. Lecture Notes in Computer Science*, vol 7858. Springer.
- [56] V. Tsiatsis, A. Gluhak, T. Bauge, F. Montagut, J. Bernat, M. Bauer, C. Villalonga, P. M. Barnaghi, S. Krco, "The Sensei Real World Internet Architecture", *Future Internet Assembly*, 247 – 256, 2010.
- [57] S. Sotiradis, E. G. Petrakis, S. Covaci, P. Zampognaro, E. Georga, C. Thuemmler. "An Architecture for Designing Future Internet Applications in Sensitive Domains: Expressing The Software to Data Paradigm by Utilizing Hybrid Cloud Technology", *IEEE International Conference on Bioinformatics and Bioengineering (BIBE)*, 2013.
- [58] J. Wurm, K. Hoang, O. Arias, A.-R. Sadeghi, Y. Jin, "Security analysis on consumer and industrial IoT devices", *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2016.
- [59] Symantec Security Response: Insecurity in the Internet of Things, <https://www.symantec.com/content/dam/symantec/docs/white-papers/insecurity-in-theinternet-of-things-en.pdf>
- [60] E. Bertino, N. Islam, "Botnets and Internet of Things Security", *IEEE Computer*, Vol. 50 (2), pp. 76-79, Feb. 2017.
- [61] 2016 DDoS attack Trends, [https://f5.com/Portals/1/PDF/security/2016\\_DDoS\\_Attack-Trends.pdf](https://f5.com/Portals/1/PDF/security/2016_DDoS_Attack-Trends.pdf)

- 
- [62] DDOS Mirai attack on DYN, <https://www.theguardian.com/technology/2016/oct/26/ddos-attackdyn-mirai-botnet>
  - [63] Hack a smart lock – Available online: <https://www.cnet.com/news/have-a-smart-lock-yeah-it-can-probably-be-hacked/>
  - [64] S. Notra, M. Siddiqi, H. Gharakheili, V. Sivaraman, R. Boreli, “An experimental study of security and privacy risks with emerging household appliances”, *IEEE conference on Communications and Network Security*, 2014.
  - [65] J. Wurm, K. Hoang, O. Arias, A.-R. Sadeghi, Y. Jin, “Security analysis on consumer and industrial IoT devices”, *Asia and South Pacific conference on Design Automation Conference*, 2016.
  - [66] A. Caragliu, C. D. Bo, P. Nijkamp, “Smart Cities in Europe”, *Journal of Urban Technology*, Vol. 18 (2), 2011.
  - [67] Abusing smart cities – Available Online: [http://securingsmartcities.org/wp-content/uploads/2016/09/BECCARO\\_abusing-smart-cities.pdf](http://securingsmartcities.org/wp-content/uploads/2016/09/BECCARO_abusing-smart-cities.pdf)
  - [68] Fooling the “Smart City” – Available Online: [http://securingsmartcities.org/wp-content/uploads/2016/09/Fooling-smart-city\\_in\\_template.pdf](http://securingsmartcities.org/wp-content/uploads/2016/09/Fooling-smart-city_in_template.pdf)
  - [69] Trick traffic sensors – Available Online: <https://securelist.com/how-to-trick-traffic-sensors/74454/>
  - [70] A. Gharaibeh, M. A. Salahuddin, S. J. Hussini, A. Khreishah, I. Khalil, M. Guizani, A. A-Fuqaha, “Smart Cities: A Survey on Data Management, Security and Enabling Technologies”, *IEEE Transactions on Communications Surveys and Tutorials*, Vol. PP (99), 2017.
  - [71] Pen Testing a City – Available Online: <http://securingsmartcities.org/wp-content/uploads/2016/03/Pen-Testing-A-City-wp.pdf>
  - [72] Guidelines for Safe Smart Cities – Available Online: [http://securingsmartcities.org/wp-content/uploads/2016/03/Guidlines\\_for\\_Safe\\_Smart\\_Cities-1.pdf](http://securingsmartcities.org/wp-content/uploads/2016/03/Guidlines_for_Safe_Smart_Cities-1.pdf)
  - [73] M. Dunn, and I. Wigert, “International CIIP Handbook 2004: An Inventory and Analysis of Protection Policies in Fourteen Countries”, *Zurich: Swiss Federal Institute of Technology*, 2004
  - [74] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, M. Dohler, “Standardized Protocol Stack for the Internet of (Important) Things”, *IEEE Transaction of Communications Surveys and Tutorials*, Vol. 15 (3), pp. 1389-1406, 2013.
  - [75] IEEE Standard for Local and Metropolitan Area Networks – Part 15.4: Low-Rate Wireless Personal Area Networks (LP-WPAN), *IEEE Standard 802.15.4-2011*, 2011.
  - [76] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler, “Transmission of IPv6 packets over IEEE 802.15.4 Networks”, *RFC 9419*, 2007.
  - [77] J. Hui, P. Thubert, “Compression Format for IPv6 Datagrams Over IEEE 802.15.4-Based Networks”, *RFC 6282*, 2011.
  - [78] P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur, R. Alexander, “RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks”, *RFC 6550*, 2012.
  - [79] M. Dohler, T. Watteyne, T. Winter, D. Barthel, “Routing Requirements for Urban Low-Power and Lossy Networks”, *RFC 5548*, 2009.
  - [80] K. Pister, P. Thubert, S. Dwars, T. Phinney, “Industrial Routing Requirements in Low-Power and Lossy Networks”, *RFC 4573*, 2009.
  - [81] A. Brandt, J. Buron, G. Porcu, “Home Automation Routing Requirements in Low-Power and Lossy Networks”, *RFC 5826*, 2010.
  - [82] Z. Shelby, K. Hartke, C. Bormann, “The Constrained Application Protocol (CoAP)”, *RFC 7252*, 2014.
  - [83] E. Rescorla, N. Moddugu, “DTLS: Datagram Transport Layer Security”, *RFC 4347*, 2006.



- 
- [84] T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", *RFC 4346*, 2006.
  - [85] "Cisco global cloud index: Forecast and methodology, 2014-2019 white paper", 2014.
  - [86] M. Chiang, T. Zhang, "Fog and IoT: An Overview of Research Opportunities", *IEEE Internet of Things Journal*, Vol. 3 (6), pp. 854 – 864, 2016.
  - [87] F. Bonomi, R. Milito, P. Natarajan, J. Zhu, "Fog Computing: A Platform for Internet of Things and Analytics", *Big Data and Internet of Things: A Roadmap for Smart Environments*, Springer Press, 2014.
  - [88] OpenCV Library - <http://opencv.org/>
  - [89] Matlab - <https://de.mathworks.com/products/matlab.html>
  - [90] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks" - <https://www.nvidia.cn/content/tesla/pdf/machine-learning/imagenet-classification-with-deep-convolutional-nn.pdf>