SEAMLESS INTERCONNECTION OF WSN AND IP NETWORKS

by
Kemal Çağrı Serdaroğlu

Submitted to the Institute of Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science
in
Computer Engineering

Yeditepe University
2013

SEAMLESS INTERCONNECTION OF WSN AND IP NETWORKS

APPROVED BY:

Prof. Dr. Şebnem Baydere ……
(Thesis Supervisor)

Assist. Prof. Dr. Gürhan Küçük ……

Assist. Prof. Dr. Özlem Durmaz İncel ……
(Galatasaray University)

DATE OF APPROVAL: ..../..../2013

ACKNOWLEDGEMENTS

It is with immense gratitude that I acknowledge the support and help of my Professor Şebnem Baydere. Her support and infinite tolerance were a big motivation for me.

I thank members of Wireless Network Laboratory, Summer 2013 Wireless Network Laboratory Internship Trainers: Erman Gonul, Baturay Ozcan, Orhan Can, İlbey Kurt, also Uğur Arpacı and Erdem Çakır for helping in development of this project. I thank Anıl Avcı for his kindly support.

Finally, I would like to thank my family for their endless love and support, which makes everything more beautiful.

ABSTRACT

SEAMLESS INTERCONNECTION OF WSN AND IP NETWORKS

In this thesis, an interconnection approach, named as WiSEGATE (**Wi**reless **Se**nsor **Gate**way), with addressing the end-to-end reliable interconnection problem between multiple internet entities and sensor nodes is proposed. For the derivation of proposed approach, firstly, we analyze the recent interconnection approaches and highlight their advantages and disadvantages, secondly, we combine their advantages . A prototype of a new web server which supports three tier service scheme with a data acquisition mechanism of WSN to access the physical data in a particular location by remote entities is developed. In the proposed model, an interconnection gateway handles operations required for the interoperability. Since, this gateway maintains reliable TCP/IP connections of the interconnected entities, the resource constrained sensor nodes do not require a TCP/IP stack for handling end-to-end connections. A lightweight service layer is implemented on a sensor node for operations required by the interconnection. The strength and novelty of the proposed model lies in the fact that this lightweight service layer relieves extra memory usage for end-to-end connection management. For determining the limits of the proposed model, firstly, we examined steps for request/response mechanism and formulize the queuing system. By doing this, we derived a definition of the request traffic. As a proof of concept, we have performed comprehensive tests in simulation and real environments. WiSEGATE can achieve reasonable response times up to 80 simultaneous connections from remote entities to WSN when WiFi PER is less than 0.2.

ÖZET

KDA VE IP AĞLARININ KALICI OLARAK BİRBİRİNE BAĞLANMASI

Bu tezde, uçtan uca güvenilir bağlanma sorunu ele alınarak çoklu ve aynı zamanlı internet varlıkları ile duyarga ağ düğümleri arasında kurulacak bir karşılıklı bağlantı modeli önerilmiştir. Önerilen modelin ortaya çıkarılması için, birinci olarak, şimdiye kadar ortaya atılmış karşılıklı bağlantı modelleri incelenmiş ve onların avantajları ve dezavantajları ortaya çıkarılmıştır, ikinci olarak ise, avantajları birleştirilmesi suretiyle model ortaya konulmuştur. İnternet varlıkları tarafından dünyanın herhangi bir noktasında fiziksel veriye ulaşılabilmesi için KDAdan veri alımı mekanizmasına sahip olan üç aşamalı servis modelini destekleyen yeni bir web sunucu prototipi geliştirilmiştir. Önerilen modelde, bir ağ geçidi birlikte işlerlikle ilgili işlemleri uygulamaktadır. Bu ağ geçidi güvenilir TCP/IP bağlantılarının sağlanmasından sorumlu olduğu için, kaynak kısıtlı duyarga ağ düğümlerinde TCP/IP katmansal modeline ihtiyaç olmamaktadır. Ağların birbiriyle karşılıklı olarak bağlanması gereken operasyonlar için duyarga ağ düğümlerinde kaynakları fazla tüketmeyen bir katman kullanılmaktadır. Önerilen modeli güçlü ve diğerlerine göre yeni kılan tarafı, bu katmanın duyarga ağ düğümlerini uç uca bağlantıların sağlanması için gerekli olan fazla bellek kullanımından kurtarmasıdır. Önerilen modelin limitlerinin görülebilmesi için, ilk olarak, sistemin istem ve cevap mekanizmasının adımları incelenmiş ve bir kuyruk sistemi formülize edilmiştir. Böylece, kuyruk sisteminin trafik tanımı ortaya konulmuştur. Modelin gerçeklenmesi amacıyla, bir dizi simülasyon ve gerçek testler yapılmıştır. Önerilen model, WiFi paket kayıp oranı 0.2nin altında iken 80 tane eş zamanlı olarak çalışan internet istemcisine kadar düzgün çalışmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

3G	Third Generation
6LoWPAN	Internet Protocol Version 6 over Low power Wireless Personal Area Networks
DAD	Duplicate Address Detection
GSM	Global System for Mobile Communications
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
ICMPv6	Internet Connection Management Protocol Version 6
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv6	Internet Protocol Version 6
kbps	Kilobits per second
Kbytes	Kilobytes
LoWPAN	Low power Wireless Personal Area Network
MAC	Medium Access Control
MTU	Maximum Transmission Unit
ND	Neighbourhood Discovery
OS	Operating System
PER	Packet Error Rate
PMF	Probability Mass Function
PHY	Physical Layer
RAM	Random Access Memory
RFC	Request for Comment
ROM	Read Only Memory
SIP	Session Initiation Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
WSN	Wireless sensor network
UDP	User Datagram Protocol
WiFi	Wireless Fidelty

WMSN	Wireless Media Sensor Networks
XML	Extended Markup Language

1. INTRODUCTION

A new era of ubiquity is coming[1]. In this era, the computing view based on human generated data is losing the ground to the physical data centric view of computing [2]. This change bears the Internet of things paradigm. This paradigm is used for building the technological backbone required to bring the traditional Internet concept to anything, anytime and anywhere[3]. Internet of the real world requires the information autonomously obtained from the observations, actions and events occur over time at particular locations in the real world. Therefore, this concept highly demands with technologies with a tight connection to the physical world [4-6].

Internet of the real world data requires highly deployed WSNs which are able to interoperate with IP networks. The main reason for this that WSNs are usually deployed for data acquisition and actuation mechanism for a physical phenomenon [2,7]. This network augments the traditional IP with sensing services at a global scale [8]. In addition, IP is the traditional communication standard used on internet hosts.

The interconnection with WSN and IP networks is necessary in several application scenarios such as remote health-care systems [9], in-home healthcare monitoring [10], activity monitoring [11] , agricultural yield control systems [12], industrial control systems [13], safety monitoring systems [14], smart home environments [15,16], traffic control systems [17], data center monitoring systems [18] and among others [4,8].

1.1. PROBLEM DEFINITION

Interoperability between WSN and IP networks is important for remote data acquisition about a physical phenomenon. This can be achieved by the seamless interconnection of WSN and IP networks. This is a challenging research problem since WSNs do not support IP networks directly. So an indirect interconnection mechanism, in which an intermediate gateway operates, is required between these networks. In addition, a data transformation and adaptation scheme is necessary in such interconnection mechanism.

In this thesis, the end-to-end interconnection problem between multiple IP hosts and a WSN node is addressed. To lay a ground for the proposed solution to this problem, the factors which affect the interconnection performance are examined and the recent approaches for the solution are elaborated in the following subsections.

1.1.1. Factors Affecting Interconnection Performance

Several factors affect end-to-end interconnection performance, its usability and interoperability. These factors are summarized below:

- *Packet Length:* Today's WSN technologies use small size frames due to low bandwidth and unreliable physical channels between nodes. Since a WSN frame size is much smaller than the minimum MTU of IP, a data fragmentation and reassembly mechanism should be added in the interconnection mechanism.
- *Bandwidth and Delay:* A physical radio channel of WSN has about 250 kbps transfer capacity due to low bandwidth. Thus, data transmission over a WSN channel introduces higher delays.
- *Loss Rates:* Since a WSN node has low transmit power and because of the harsh radio conditions of WSN, the transmission channels in WSN are more prone to noise and interference. So, packet loss rates in WSNs are higher than in a typical wireless IP network. Higher packet loss rates affect the end-to-end transmission performance negatively. If a typical reliable data delivery protocol such as TCP is required in connections between a WSN node and an IP host, this protocol will misinterpret the packet loss as a congestion and packets will be resent due to false timeouts which is a known problem for TCP over unreliable wireless links [19].
- *Multi Hop Transmission of WSN:* Low transmit power for WSN channels leads the packets to travel in shorter distances. So, for acquisition of data in remote environments, a multi hop data transmission scheme is used in a typical WSN. As the number of hop increases in WSN, end-to-end delay between sink node and any one of the remote sensor node increases. Thus, end-to-end transmission is affected negatively in term of response time.

1.1.2. Interconnection Approaches

There are two basic categories of approaches for solving the interoperability problem[8]. First is the proxy-based approach [20] which simply separates the WSNs and IP networks and lets WSN to operate its own dedicated protocols and a proxy server is used for the interconnection. The second approach, named as gateway-approach [8,21], considers the sensor nodes as the individual IP hosts and aims to interconnect these nodes to the IP hosts with an end-to-end interconnection mechanism.

1.1.2.1. Proxy Based Approach

It is simple to resolve the interconnection problem through the setup of WSN-IP proxies [20]. A WSN-IP proxy periodically acquire sensor data from WSN connected to its back-end with using protocols dedicated to WSNs. Then it records the data obtained from WSN to a relational database. For its front-end, it acts like a server and it responds clients with the data in its relational database. In addition, data inquiry mechanisms may help the proxies for an intelligent decision system. Figure 1.1 illustrates the connection scheme for the proxy approach.

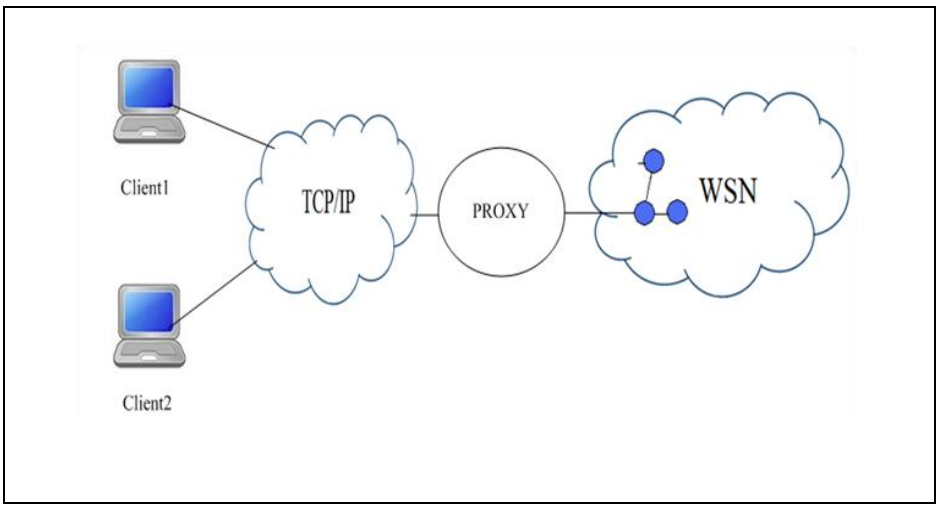


Figure 1.1. Proxy approach

1.1.2.2. Gateway Based Approach

Gateway based approach [21] aims to solve the interconnection problem with an end-to-end communication scheme as illustrated in Figure 1.2. This approach considers the sensor nodes as individual IP hosts. Thus, this approach aims to integrate the IP concept into these nodes. For achieving this, sensor nodes require an IP communication stack on themselves. With this approach, a data communication standard, named as 6LoWPAN, is defined for the integration of IP to the sensor nodes [22]. In addition, this approach requires a gateway node for the interconnection mechanism. An adaptation layer should be added between layer-2 and layer-3 of TCP/IP stack for necessary packet transformation [22] on the gateway.

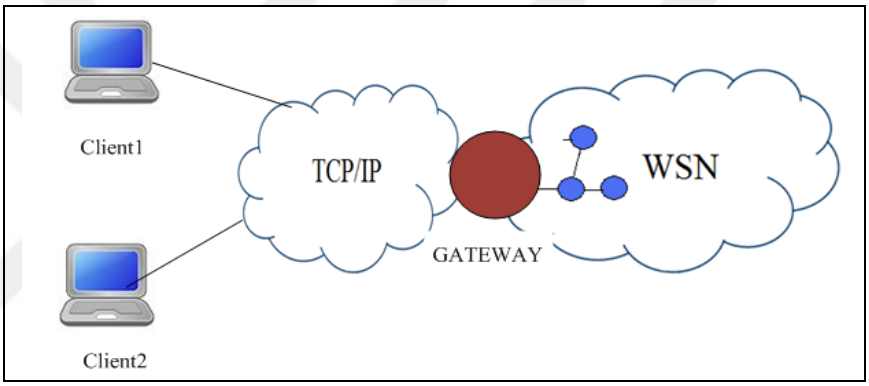


Figure 1.2. Gateway Approach

1.1.2.3. Discussion

These approaches are basic definitive models for the solution of the interconnection problem. These approaches have some advantages and disadvantages in their nature. The former approach relieves sensor nodes from IP stack activities and it does not require extra computational and storage resources of sensor nodes. Hence, the overall WSN only operates protocols tailored to its constraints. In addition, a typical web server operation can be applied on the proxies and a proxy can be configured with security and authorization schemes for the multiple users [20]. However, this approach lacks a standardized interoperability model so everyone can have their proxies and significant semantic translations are required between those proxies [23].

For the latter approach, a direct end-to-end intercommunication model between IP and sensor nodes is offered. In this model, a data adaptation standard [22] for the interoperability is used. This standard requires an adaptation layer both in gateway and sensor nodes. Whenever overall adaptation and interconnection mechanism is setup on sensor nodes and the gateway, this approach can be easily used for the interoperability. However, this approach requires IPv6 because it assumes the sensor nodes as individual IP hosts and these nodes should have unique IP addresses [22]. It is difficult to realize this with IPv4 because of its addressing space [23]. In addition, sensor nodes require extra operations for the adaptation of the packets and an adapted TCP/IP stack in it. Moreover, for a fully interactive sensor service, the sensor nodes require additional functionalities for end-to-end connection management, application message coding and client session management. For example, if any one of the sensor node is used as a multi client web server, end-to-end TCP/IP connections should be maintained by the sensor node and HTTP message coding mechanism should be added to these nodes. The overall operation for a web server in a sensor node is a memory intensive and complex way to solve the problem. Thus, this might lead unfeasible solutions for the sensor nodes because of their resource constraint nature. In addition, gateway machine requires additional operations to extract link layer frames and by-pass the unnecessary parts in it. This is an extra overhead for the system.

1.2. MOTIVATION AND AIM

Considering the factors affecting interconnection design and approaches mentioned in the previous section, a new approach combining these two approaches is proposed in this thesis as an alternative solution for the interconnection problem. The proposed model has the following design considerations.

- *Small memory footprint for sensor nodes:* Sensor nodes are resource constrained devices and operations for IP interoperability and end-to-end connection management require intensive resources. We can easily deploy a resource intensive interconnection gateway as the intermediate node and since this interconnection gateway handles such operations required for end-to-end connection management

between the client and a sensor node, we can relieve sensor nodes from these resource intensive operations.

- *Standardized Interconnection:* Between interconnection entities, several packet transmission and messaging standards should be operated. So, such network entities which are in the interconnected heterogenous networks can interoperate with each other.
- *Application Gateway:* A multi-client sensor service requires simultaneous session management and message coding mechanism for these sessions. A web application over TCP/IP is generally used for these operations. So an application gateway is designed to handle simultaneous clients' requests and for necessary semantic translations between client messages and sensor node messages.
- *Transparency of Underlying Network:* Application gateway scheme mentioned above is helpful to implement the server logic with a transparency of underlying network. So this scheme can be easily operated over some network protocols such as IP, GSM and 3G.
- *Modular Gateway Design:* The application gateway is responsible to handle various application layer messages. A generic modular design for the application components is helpful to extend the application gateway for operations with various application layer protocols.

Considering the properties mentioned above, this dissertation aims to produce a new interconnection mechanism at which multiple simultaneous remote clients can reach to the sensor service seamlessly.

1.3. OVERWIEW OF THE PROPOSED MODEL

In this thesis, a hybrid interconnection mechanism named as WiSEGATE (Wireless Sensor Network Gateway) which combines proxy and gateway approaches mentioned in the previous section is proposed. It addresses the seamless interconnection problem highlighted in Section 1.1. WiSEGATE design is composed of two functioning components; an interconnection gateway and a lightweight service layer on WSN node.

WiSEGATE uses a new server scheme in which a WSN and the interconnection gateway operate together as a web server. In this scheme, when a WSN is used to gather sensor data from the environment for incoming requests from clients, an application gateway is used for the interconnection. This gateway establishes connections between the sensor server and multiple clients via TCP/IP socket interface. Then, it accepts incoming requests in the form of application messages from the clients, exchanges the data between the interconnected networks and make sensor data ready to service to the client. Additionally, the interconnection gateway handles TCP connections between WSN and the clients because of high loss rates in WSNs. If a sensor node handled the reliability of the connections in itself, more false TCP timeouts would have occured due to high wireless loss rates in WSN side. In addition, for this approach, the overall WSN is considered as a local IPv6 network and the 6LoWPAN datagrams are used to carry sensor data between WSN nodes and the gateway.

On the WSN side, the sensor nodes do not implement a TCP/IP stack. There are two reasons for this with considering the resource constrained nature of the sensor nodes. Firstly, TCP handles its connections independently and for each connection, it requires computational and storage resources for this mechanism. Secondly, for each session of a client, every server requires extra computational and storage resources. So, as the number of simultaneous clients increases, extremely large number of resources will be needed in a sensor node. Therefore, some studies [2,24] show that operating the TCP/IP over WSN is not suitable for sensor nodes even for the small number of simultaneous clients. This situation would get worse if we want to bring the sensor service to multiple simultaneous clients. Since the sensor nodes are resource constrained devices, in our model, a lightweight service layer is employed on sensor node for the interconnection.

Table 1.1 gives the comparison of WiSEGATE with proxy and gateway approaches. The strength of WiSEGATE is that it handles reliability of the interconnection channels between clients and WSN with the gateway and it relieves operations for TCP/IP and application layer messaging in a sensor node for multiple simultaneous connections of clients.

Table 1.1. Comparison of interconnection approaches

Proxy Based Approach	**Gateway Based Approach**	**WiSEGATE**
no TCP/IP activity on sensor node	TCP/IP activity on sensor node	IP activity on sensor node
no TCP/IP stack	TCP/IP Stack	lightweight service layer
WSN: protocols for its constraints	WSN: Protocol for IP and WSN constraints	WSN: protocols for its constraints
Proxy: Multiclient web server	Gateway: data adaptation. Sensor Node: Server	Gateway: Multiclient web server, data adaptation
End-to-end interconnection to proxy	End-to-end interconnection to sensor node	End-to-end interconnection to sensor node
Web server logic on proxy	Web server logic on sensor node	Web server logic on gateway
Sessions for semantic translation.	No semantic translation, direct messaging to clients	Sessions for semantic translation
Reliability: between proxy and client by proxy	Reliability: between sensor node and client by sensor node	Reliability: between sensor node and client by gateway

1.4. CONTRIBUTIONS

Considering the problem highlighted in Section 1.1 and the specification of the model defined above, the following contributions are made: An interconnection model is designed combining the traditional three-tier server approach with a data acquisition mechanism of WSN. A gateway operation model is designed for the interoperability and adaptation of interconnected networks. In this approach, since TCP/IP connections are maintained by the gateway node, TCP/IP adaptation is not required in sensor nodes. In this model, while sensor data requests from multiple number of clients are served from this gateway with application message protocols, a lightweight data acquisition and adaptation mechanism is used in WSN side. The data can be obtained from WSN and viewed easily using a web browser from the client side.

For the proof of concept, a series of comprehensive tests are performed in a simulation environment to investigate the scalability of the WiSEGATE with increasing number simultaneous clients in a wireless transmission environment. A Poisson-based traffic model is defined and used to generate a constant bit rate (CBR) request traffic. With stable and bursty request traffic, the performance of the multiple TCP connections over WiSEGATE is observed in a WLAN.

Moreover, the performance and implementation costs of the proposed interoperability model is evaluated on a real implementation environment. For this, a real WSN testbed with Telos equivalent TmoteSky nodes is constructed. This testbed is connected with multiple web clients via a gateway machine which implements the proposed gateway model. Several performance tests have been done and early performance results for the proposed approach in the real environment have been obtained. This preliminary study is accepted as a publication [6].

1.5. ORGANIZATION OF THESIS

The rest of the thesis is organized as follows: Chapter 2 presents the related work, categorizing the studies in terms of interconnection models, data communication standards and gateway performance aspects. In Chapter 3, the operational units of the proposed

interconnection model are introduced. Chapter 4 explains the traffic model which is derived for the performance analysis of WiSEGATE in a simulation environment. In Chapter 5, the performance analysis of the WiSEGATE is presented. Lastly, Chapter 6 presents the conclusion and the future work for the thesis.

2. RELATED WORK

In this chapter, background information and the state of art are introduced. In Section 2.1, the interconnection models used by several studies are provided. In Section 2.2, several data communication standards which specify transmission of data technique in the interconnected networks and way of interpretation for data in network entities are presented.

2.1. INTERCONNECTION MODELS

Proxy and gateway approaches are implemented with several interconnection models. These models have specific design considerations such as data collection scheme, interconnection method, implementation area of the models, data adaptation and the implementation method in those models.

2.1.1. Proxy Based Models

For the proxy approach, there are two common patterns used in the models for collection of data [20]. First pattern is the forward-server and second is the front-end server. For the former, the data is forwarded directly and a database in the proxy is not necessary. For the latter, it is necessary to use a database for the collected sensor data and these data are used to provide services, in analysis, management and inquiry. The physical data can be obtained using SQL queries or a web browser from the client.

Ting et al. [20] use a proxy approach for the interconnection in their work. As presented in Figure 2.1, this work uses user agents (UA) for the connection between clients and the proxy. When a client requests data from the WSN, a UA is established for this request. This UA sends the request to the Resource Management (RM) module. This module is responsible for the data inquiry from the WSN. This module gets recent data from the database (DB) at the proxy or with the help of a transceiver (TR) module, it gets the fresh data from the WSN. The data inquiry flow which expresses their implementation for proxy is depicted in Figure 2.2.

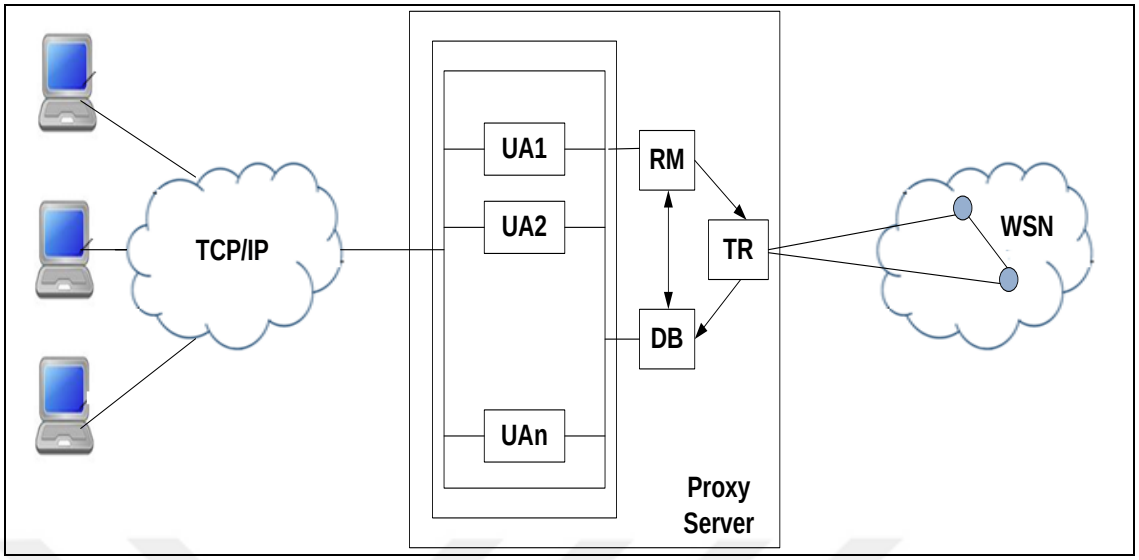


Figure 2.1. Interoperability model by Ting et al. [20]

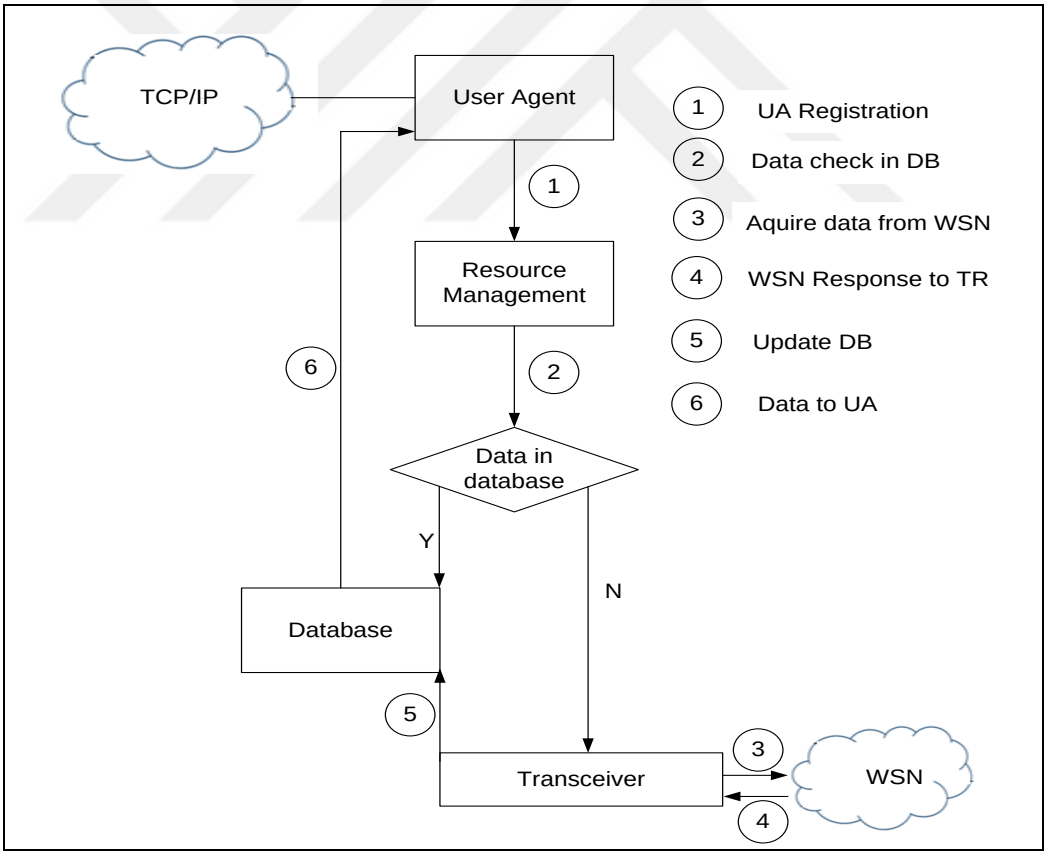


Figure 2.2. Data inquiry flow chart by Ting et al. [20]

Chen et al. [25] present a prototype of a smart gateway which operates as a proxy for the WSN. This smart gateway is built for indoor health care systems and provides interconnection and services management platform. It is compatible with an on-board data decision system and a database which enable to make the patient's health state decision in the proxy. The proxy model is divided into two models: simple model and intelligent model. In so called simple model, the proxy acts as a forward-server and the proxy only forwards health state data to the health-care centers. In the second model, proxy has a decision making mechanism and acts as a front-end server. This decision making mechanism is used to detect real time health state of the elderly people for remote health-care.

Narmada et al. [26] present an intelligent parking guidance and management system which uses a proxy for the interconnection. This system is based on gathering data from sensor nodes which determines the free slot of the parking area and forwarding them to the some operational units such as a motorized barrier control, a ticket machine, a help console, a sign boards, a GSM modem and the internet. There is no requirement for any external data storage since this proxy uses forward-server approach.

2.1.1.1. Discussion

Proxy based models are important to examine the interoperability behavior of the proposed model. A User Agent or a session management module in the proxy is important for semantic analysis for the requests. WiSEGATE can be considered as a proxy because it does a semantic analysis of the request messages coming from simultaneous clients in its session management modules (i.e., user agents). It can interpret the request messages and create a byte-coded representation of these messages. In addition, for a response, it interprets the byte coded data coming from the WSN and creates application messages with this byte coded data. However, WiSEGATE cannot also be considered as a proxy since, a client application establishes end-to-end connection to the sensor service on the sensor node rather than to a user agent in the proxy. User agent in the proposed model forwards the request message with its byte coded representation to the WSN.

2.1.2. Gateway Based Models

Gateway based approach provides an end to end interconnection mechanism between clients and sensor nodes. This mechanism needs two operational units for the interconnection. The former is the adaptation layer in a gateway machine [22]. The latter is an adapted communication stack on sensor nodes. This stack is used to handle various communication protocols for the end to end interconnection of sensor nodes and clients and the gateway.

The studies in [27-33] aim communication stack design for sensor nodes. The main property of these stacks is the compatibility of 6LoWPAN [22]. These so-called 6LoWPAN stacks aim to adapt sensor nodes to IPv6.

Harvan et al. [27] introduced a 6LoWPAN stack specifically designed for TinyOS 2.0 operating system [34] and tested on TelosB [35] and MicaZ [36] sensor nodes. This stack supports UDP transport protocol on sensor nodes and necessary header compression schemes of 6LoWPAN [11]. For the design of the gateway, as seen in Figure 2.3., an adaptation mechanism requires a tunneling daemon and Tun device [37] to bypass link layer frames. This mechanism is required to extract necessary data from the frame and do necessary data transformation.

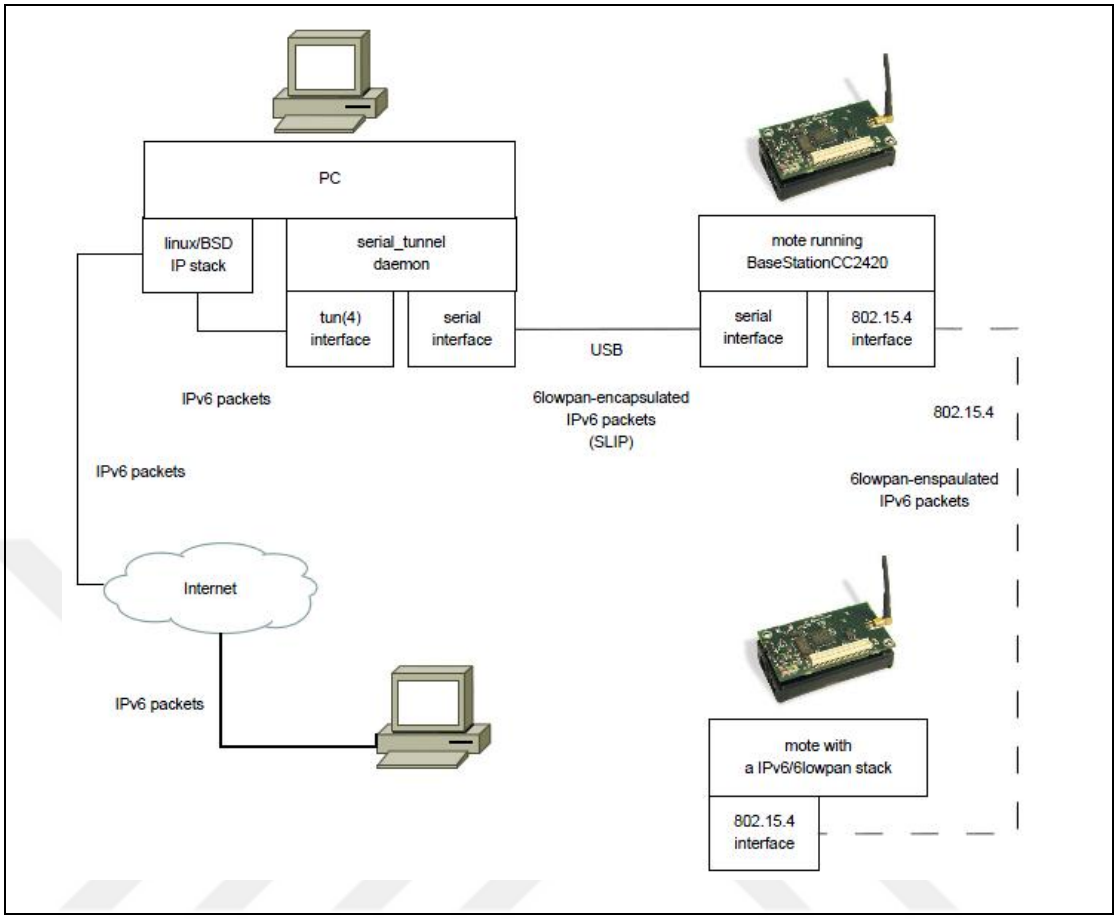


Figure 2.3. Motes, tunelling daemon and Internet by Harvan et al. [27]

Blip (Berkeley IP Information) [38] is another 6LoWPAN stack which is an interconnection model coded for TinyOS 2.1.1 [39]. Blip uses UDP as the transport protocol and includes IPv6 neighbour discovery (ND) [40], default route selection, point-to-point routing and network programming support.

Dunkels et al. [28-31] introduced uIPv6 (microIPv6) and lwIP (lightweight IP) stacks for IP based sensor nodes. Their work is among the first to be recognized for integration TCP to the sensor nodes for end-to-end reliable communication. When uIPv6 were designed for small devices, lwIP supports extra IPv6 operations such as ICMPv6, ND and DAD. These stacks were integrated in Contiki OS [41]. A memory usage analysis have been done for these stacks. The results are given in Tables 2.1 and 2.2. The values in these tables exclude the memory usage by the hardware drivers, 802.14.5 PHY and MAC, 6LoWPAN with

fragmentation and header compression, Contiki OS implementation. If those values are added, for the uIPv6, total memory usage reaches to 39K.

Table 2.1. Memory footprint for uIPv6 stack (Dunkels et. al [28])

Function	**Code Size(bytes)**
Check Summing	464
IP, ICMP and TCP	4452
Total	5188

Table 2.2. Memory footprint for lwIP stack (Dunkels et. al [28])

Function	**Code Size(bytes)**
Memory Management	2512
Checksumming	504
Network Interfaces	364
IP	1624
ICMP	392
TCP	9192
Total	14588

High packet drop rates are common in wireless networks because of high error rates in the transmission medium. Since TCP always interprets packet drops as a sign of congestion, it misinterprets a packet drop which is a result of bit error as a sign of congestion. So in a wireless network which has a harsh radio conditions (e.g., WSN), TCP will decrease the sending rate, even though congestion does not occur in the network [19]. Therefore, end-to-end data recovery performance for interconnection is affected with this paradigm, especially in the gateway models. Dunkels et al. [42] extends the lwIP stack with a data recovery method named as Distributed TCP Caching (DTC). DTC was mainly designed for error recovery on the communication stack of the sensor nodes. In DTC, each data segment is cached in an intermediate node. When a data segment is lost, the cached segment is sent with local retransmissions rather than an end-to-end retransmission of the segment. For higher performance, in their model, they devise a node selection algorithm

for determining cacher node for a segment. They did a packet load analysis of nodes with detecting number of packets sent in each node. This analysis show that DTC reduces the load of the node.

BACNet [32] is an extension of uIPv6 stack which is the interoperability model for remote building automation and control. It is used to collect information from other devices or objects (read property), command a device to execute a particular operation (write property) and notify the occured event to the device. The memory footprint analysis of BACNet is given in Table 2.3.

Table 2.3. Memory footprint for BACNet of Zhou et. al [32]

Read Property	**Write Property**	**Memory Consumption**	
		ROM (Bytes)	RAM (Bytes)
No	No	30356	2564
Yes	No	35765	3042
Yes	Yes	39743	3341

Han et al. [33] introduces a subset of TCP/IP stack for sensor nodes, which they call Tiny TCP/IP. In this interconnection model, full functionality of the TCP/IP protocol suite is not implemented. Figure 2.4 illustrates the architecture of their design. As seen in this figure, in a WSN cloud, there could be a sink node, mobile nodes and fixed nodes. The application layer is only implemented in the sink node for handling request of a client. UDP is used as the transport protocol between the sink node and other nodes in the WSN. Their model supports mobility for the sensor nodes and SIP is implemented in those nodes.

They excluded some propoerties of Network Layer, Transport Layer and Application Protocols in TCP/IP protocol suite for their design. The most important exclusion had been done in the TCP. The Figure 2.5 illustrates the reduced TCP state diagram for a connection established between their sink node model and a client. Tiny TCP/IP uses 11.6 Kbytes of code to accomplish its tasks.

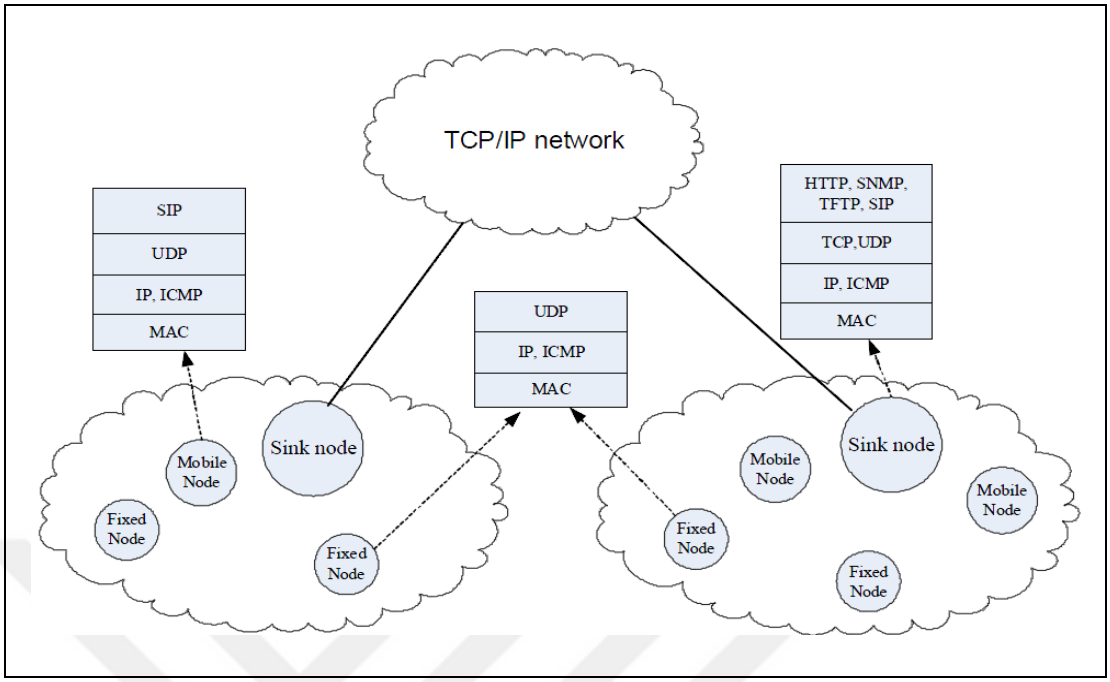


Figure 2.4. Interconnection scheme for Tiny TCP/IP by Han et al. [33]

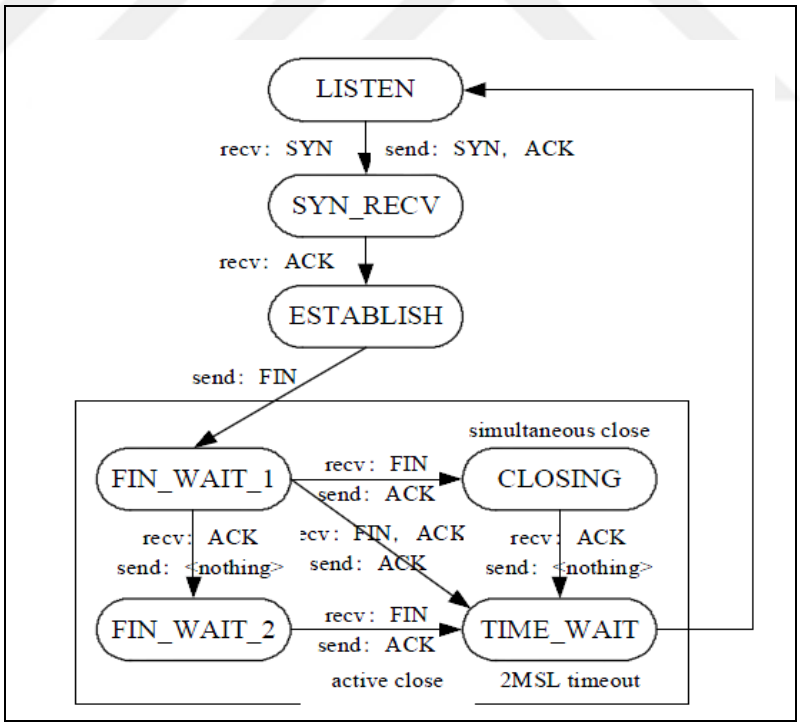


Figure 2.5. TCP state diagram used in sink node for Tiny TCP/IP by Han et al. [33]

2.1.2.1. Discussion

These models use end-to-end interconnection scheme and the gateway is used for data adaptation and packet forwarding toward the destination node. A sensor node should handle the incoming packets in its communication stack using specifically designed protocols and it requires extra operations for the interconnection. The memory usage results given in tables show that gateway models need require resources on a sensor node. In the presented analysis of the models given above, any analysis for evaluation of these models with a simultaneous multi-client request traffic was not found. So we cannot comment about their memory usage on a sensor node when simultaneous multi-clients request sensor service from it. On the other hand, our interconnection model eliminates this uncertainity with handling simultaneous end-to-end connections on the interconnection gateway.

Since WiSEGATE aims to solve the interconnection problem with a end-to-end connection scheme, the proposed model can be considered as a gateway model. As mentioned earlier, instead of using a protocol stack in a sensor node, a lightweight service layer is used for the interconnection operations in WiSEGATE. The memory usage analysis of this layer is given in memory comparison section in Chapter 5.

2.2. DATA COMMUNICATION STANDARDS

Several studies focus on data communication standards for the interoperability. These standards specify the packet formats, the data adaptation scheme of the interconnection, the messaging scheme for the interconnection entities.

2.2.1. 6LoWPAN Standard

In RFC 4644 and RFC 6282 [22,43], the IETF working group had introduced an interconnection standard for IPv6 networks and LoWPANs. This is called 6LoWPAN standard and used in a broad research community for solution of the interconnection problem. As mentioned earlier, this standard is generally used in the gateway-based solutions.

The 6LoWPAN standard mainly focuses on the frame formats for transmission of IPv6 datagrams on LoWPANs and IPv6 addressing scheme [44] for the nodes in a LoWPAN [22]. Since minimum MTU for IP datagrams is 1280 octets and the remaining size for a IP datagram in an IEEE 802.15.4 frame is 81 octets, a data fragmentation and reassembly mechanism are introduced for the interconnection. So this standard requires an adaptation layer below IPv6 in the TCP/IP protocol suite for this operation. In addition to this, to carry large amount of data for an application message which is going to be inserted in a LoWPAN frame, header compression mechanisms are introduced for eliminating the unnecessary and redundant infromation in a IPv6 datagram. In addition, 6LoWPAN standard supports multihop transmission of a LoWPAN frame, so this standard defines a multihop data transmission mechanism for 6LoWPANs. Figure 2.6 illustrates the frame formats defined in 6LoWPAN standard. As seen in this figure, four different frame formats are defined to support fragmentation and multi hop frame handling mechanism of the adaptation layer. Every type of frame has its own 6LoWPAN datagram header to specify the operation which should be carried out for it. Creation mechanism of such datagrams and the overall adressing scheme for the IPv6 sensor nodes which is included in the standard are introduced in detail in RFC 4644 [22].

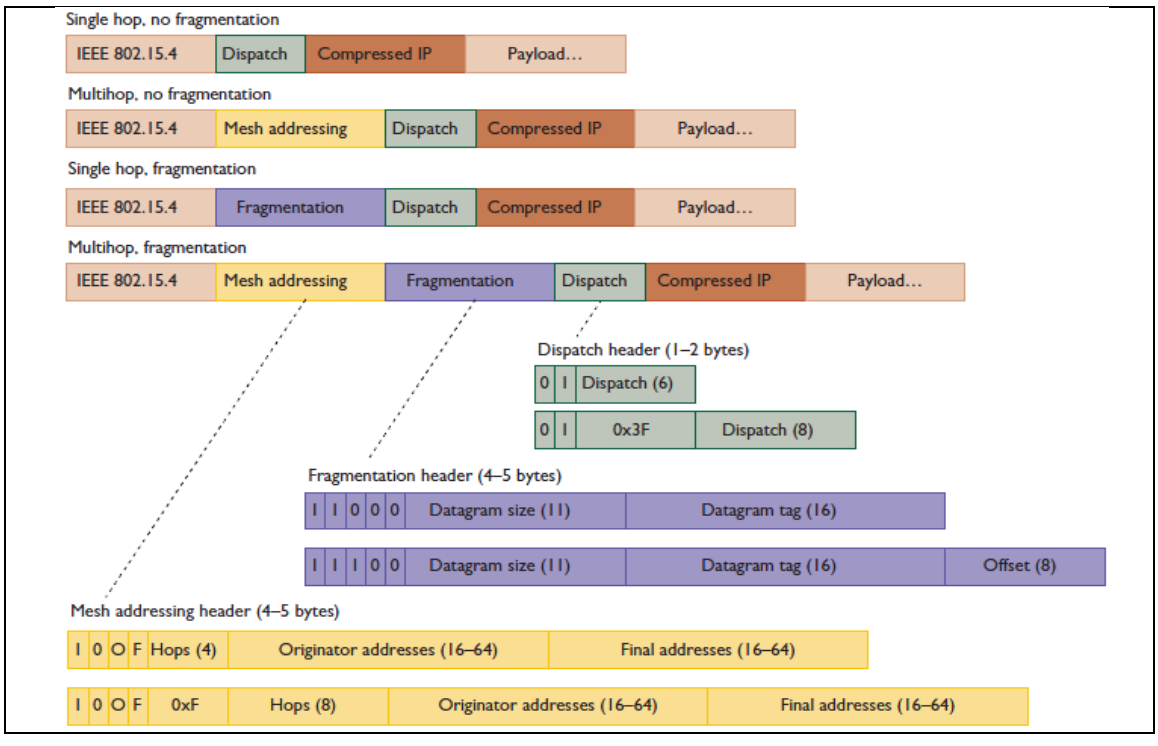
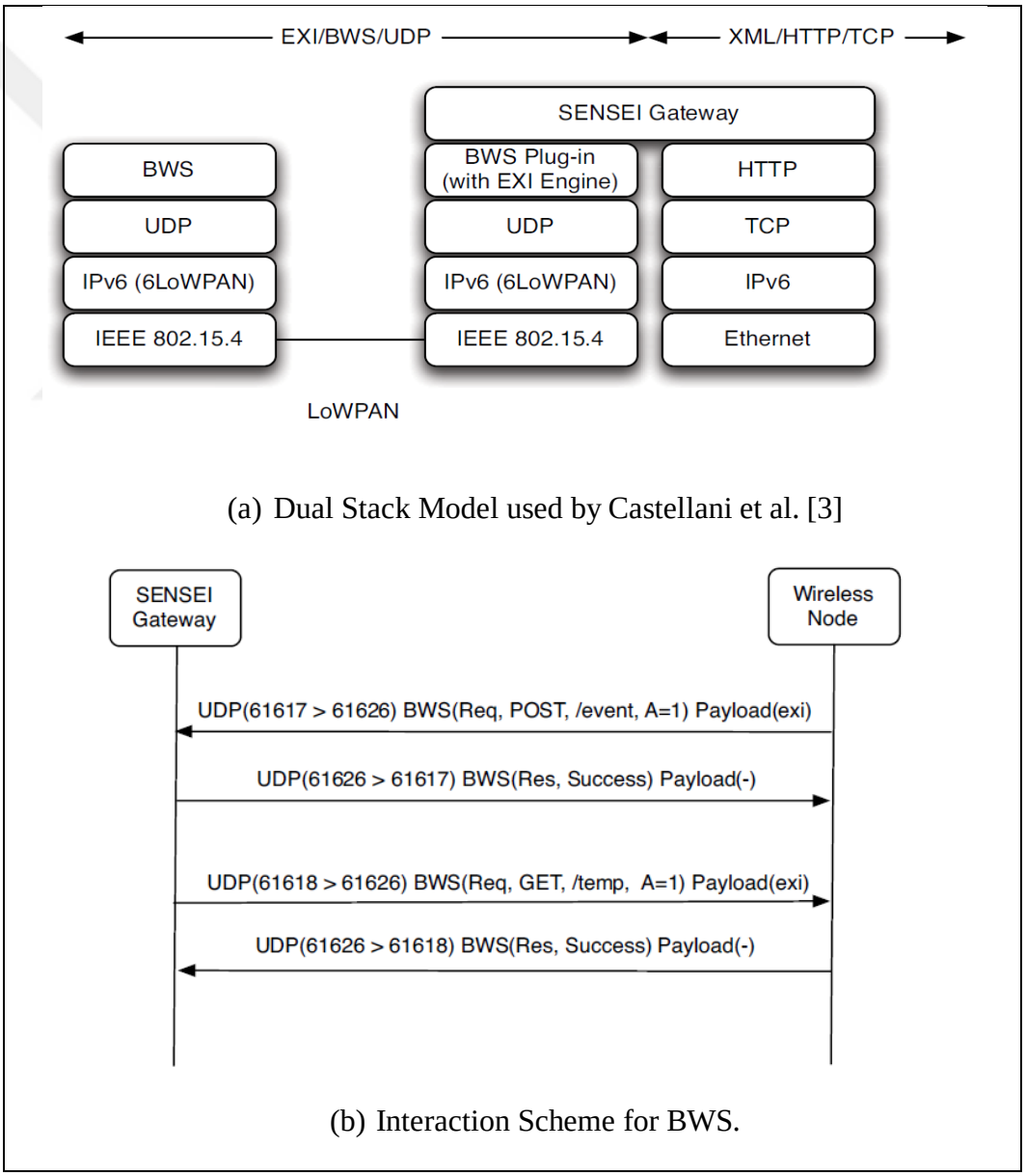


Figure 2.6. 6LoWPAN datagram types (figure from [5])

2.2.2. Application Layer Standards

2.2.2.1. BinaryWS (Binary Web Services)

Castellani et. al. [3] introduce a REST (Representational State Transfer) standard, named as BinaryWS (BWS), which is used to deliver byte coded data in an application message. This byte coded data represents the compact form of a Web Service message in which the sensor node response is carried. This work was the part of the SENSEI project [45] which is a pan-european testbed for IoTs in the global scale network.



(a) Dual Stack Model used by Castellani et al. [3]

(b) Interaction Scheme for BWS.

Figure 2.7. Stack model and interaction scheme of BWS by Castellani et al. [3]

As illustrated in Figure 2.7-a, BWS uses a dual stack in the gateway machine. The stack of the back end operates for the LoWPAN. BWS Plug-in of this stack converts XML messages into BWS messages and sends them to wireless node on LoWPAN. An example of interaction of BWS between a wireless node and the gateway is given in Figure 2.7-b. It is clear that the front-end of the gateway is a traditional web service interface and XML and HTTP [46] are used to carry web service messages between the gateway and the clients. BWS messages are converted to XML messages by the BWS Plug-in.

BinaryWS had been implemented with several 6LoWPAN stacks. The table 2.4 gives the ROM/RAM utilization of BWS on those stacks. BWS costs 19K in a TinyOS [34] compatible sensor node which operates blip [38] as 6LoWPAN Stack.

Table 2.4. Memory Footprint for BWS (Castellani et al. [3])

Component	**Memory Consumption(bytes)**			
	TinyOS		**ContikiOS**	
	ROM	**RAM**	**ROM**	**RAM**
TinyOS + 802.15.4	10816	332		
UDP/6LoWPAN (blip)	5182	1936		
Contiki + IPv6/802.15.4 (uIPv6)[17]			40960	3024
libBWS (BWS logic)[3]	2950	326	1454	0
Total	18948	2594	42414	2858

2.2.2.2. CoAP(Constrained Application Protocol)

CoRE (IETF Constrained RESTful Environments working group) defines CoAP [47] which is a REST standard for resource constraint devices. With the help of CoRE, these devices can communicate with network nodes which use HTTP messaging.

Figure 2.8-a illustrates the communication scheme for CoAP. CoAP uses a proxy between constrained environments (e.g. a WSN) and internet. With the help of the CoAP proxy, HTTP server in the internet cloud can communicate with a resource constraint node. A CoAP proxy operates a dual stack for interconnection. The elements of this dual stack is

illustrated in Figure 2.8-b. In addition, HTTP server can use CoAP to communicate directly with a resource constrained device by using an external application in it.

Within UDP headers, CoAP uses a four byte binary-header followed by a sequence of options. In its payload, it uses familiar four HTTP request methods: GET, PUT, POST and DELETE. The resources (i.e., the constrained device) are identified with URIs (Universal Resource Identifiers) which is a well-known resource identifier method for RESTful services.

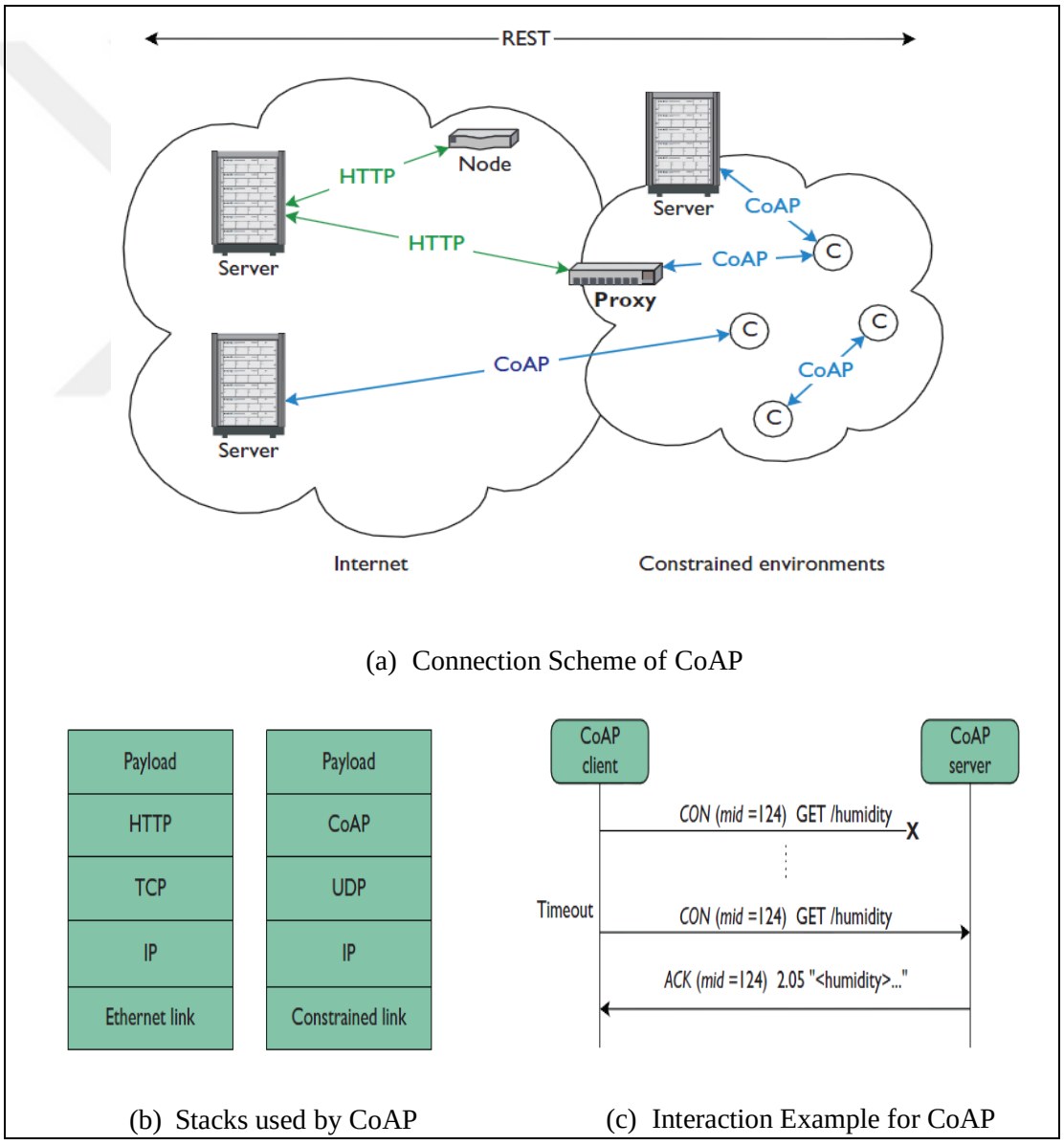


Figure 2.8. Connection scheme, dual stacks and interaction scheme for CoAP

On the resource constrained device, CoAP operates UDP for transmission of application data. Since the data transmission is unreliable, CoAP provides an timeout mechanism for retransmission of lost packets, as illustrated with the interaction diagram in Figure 2.8-c.

CoAP had been integrated for several constrained environments. Colliti et al. [48] had studied the integration of CoAP to WSNs. In [49], Chander et al. had given the specification of CoAP based integration model for Web of Things (WoT) [50]. Laum et al. [51] had specified an architecture for augmenting CoAP in wireless cellular systems. Rajesekaran et al. [52] had introduced a remote toll gate system which uses CoAP for messaging. Chatzigiannakis et al. [53] had studied for a CoAP based network self-configuration tool integrated for IoT of consrained environments. Park et al. [54] had presented a CoAP based method for monitoring 6LoWPAN testsbeds.

2.2.2.3. Discussion

These application layer standards determine the interoperability between the gateway node and the destination node which gathers sensor data from the environment. Gateway node uses a dual stack in which the first stack accepts incoming connections and requests in the form of HTTP messages from clients and second stack interprets the messages in well known XML and HTTP formats and creates compact representations of them. In addition, the second stack operates UDP for connection between the gateway node and the destination sensor node.

WiSEGATE has an interoperability scheme as in the connection models developed for these application layer standards. However, when it is configured as a web server, it eliminates all HTTP headers in an HTTP message and with the help of a pre-determined data structure, it creates a byte coded representation of the request in that message. Since connection channels between a sensor node and the gateway is unreliable, the proposed model has a timeout mechanism in the gateway like in CoAP to maintain a reliable communication over WSN.

3. WISEGATE : WIRELESS SENSOR NETWORK GATEWAY

This chapter describes the components of WiSEGATE. As shown in Figure 3.1, WiSEGATE is based on the design of the service scheme in which the whole WSN and the gateway operate as a web server together. WiSEGATE uses an application gateway which operates over TCP/IP for multiple clients. This application uses an adaptation layer for transformation of data packets to be understood in each networks. In addition, for the adaptation of sensor nodes to this application gateway, a lightweight middleware is designed. With WiSEGATE, multiple remote clients are able to reach any sensor data over WSN without the need of an external storage.

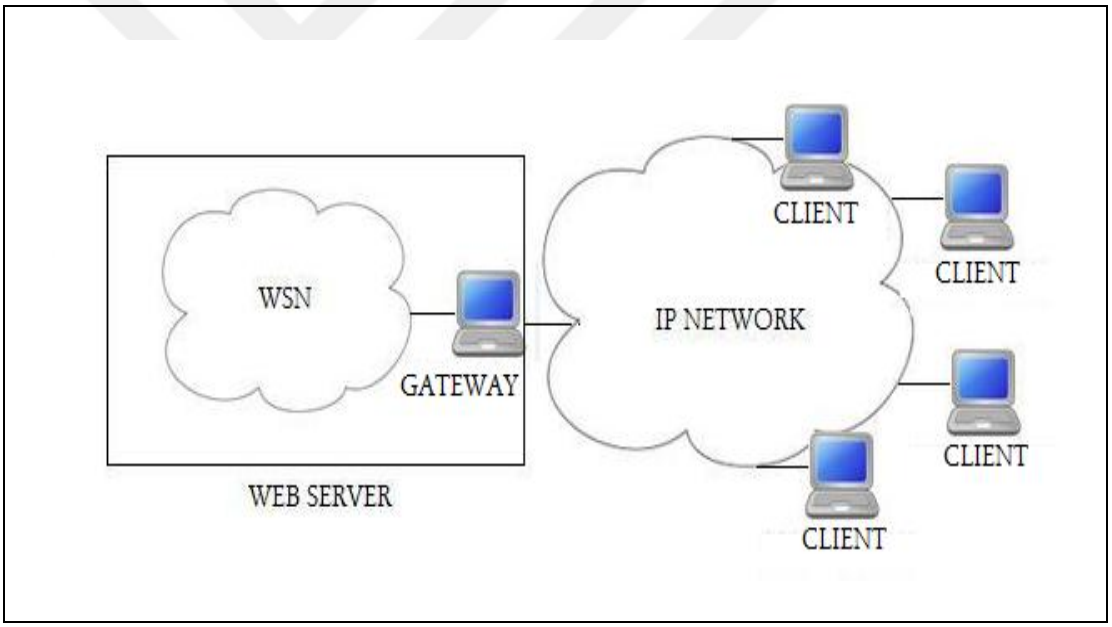


Figure 3.1. Service scheme for the proposed solution

Figure 3.2 illustrates the whole scheme of the request/response system. As seen, the system has three tiers which are the presentation, service and the WSN tiers. In this three tier model, a WSN node is the physical data acquisition device for the server and the whole WSN is used to gather physical data in a particular area. This data gathering mechanism is a data tier for the server and it can be specifically called as the WSN tier.

The gateway node is responsible for replying the sensor data requests coming from the clients and request the sensor data from the WSN tier. Thus, gateway provides the service

tier functionality. The application under the client serves the function of the presentation of the data and this tier is called as the presentation tier similar to traditional server-client model.

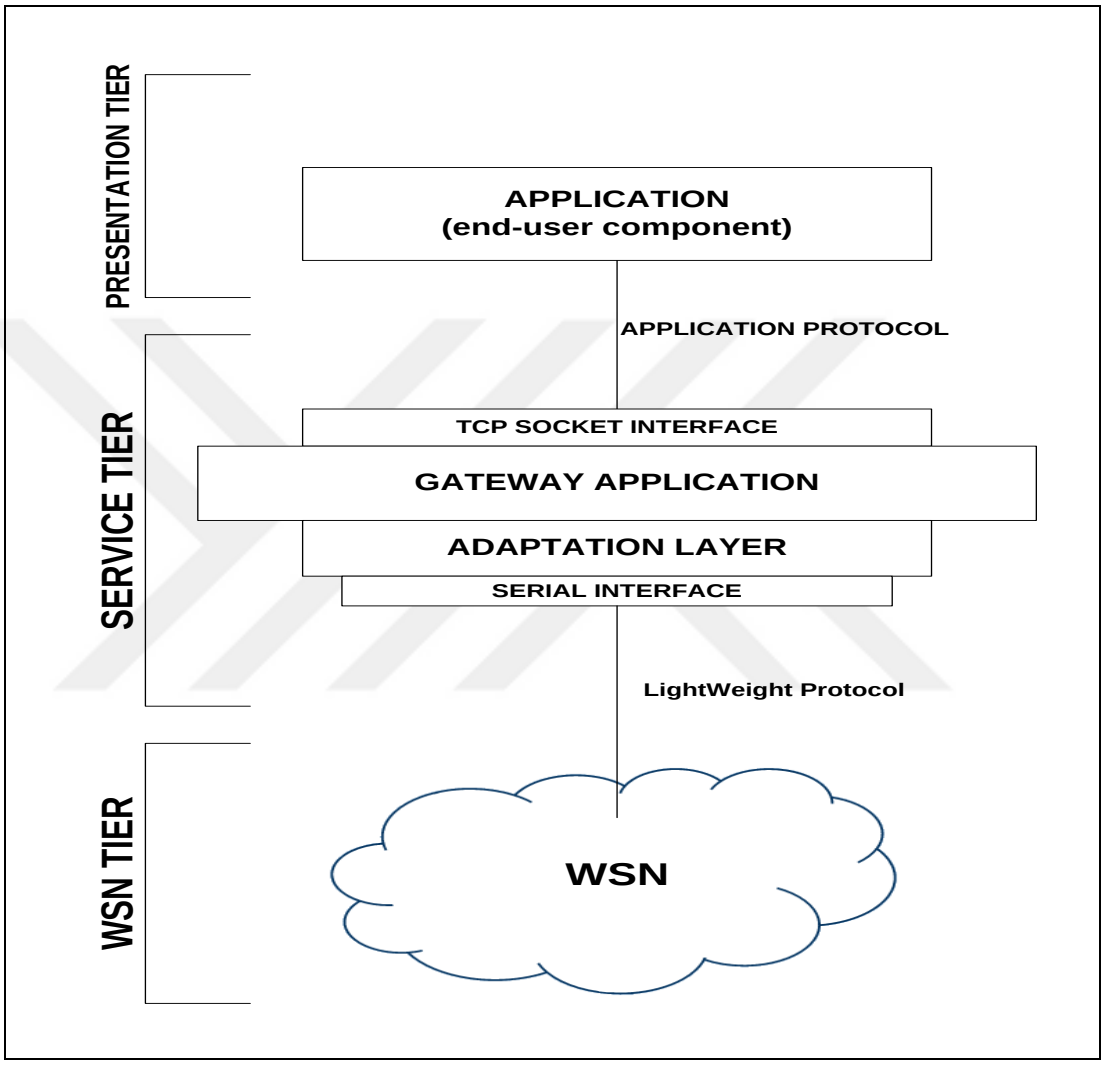


Figure 3.2. Interconnection scheme of WiSEGATE

3.1. GATEWAY NODE

Gateway node is the network unit which is used to exchange packets between WSN and IP networks. It has an adaptation layer which is responsible for transforming packets to be understandable in each interconnected network. In addition, the gateway node maintains the established TCP/IP connections.

In our model, gateway handles application messages arriving from clients via IP network. Thus, it is responsible for handling various application protocols such as HTTP so it uses various network applications. Each application operates on a specific port for the protocol and handles incoming messages under it. For example, if the gateway operates as a web server, it uses HTTP messaging protocol, port 80 over the socket interface and TCP as the transport layer protocol.

For its operation, the gateway should communicate with both WSN and IP networks. So, communication interfaces are required by the gateway node. These communication interfaces are used to send/receive data packets to/from the corresponding network. For the communication with IP, TCP/IP socket interface is used and for the communication with the WSN, serial communication interface is used.

The gateway has the following components. It has the gateway application and the adaptation layer. These components are described in the following sections.

3.1.1. Gateway Application

WiSEGATE communicates with clients via various application protocols. It exploits the request messages in the form of these application protocols to understand the service expected from the WSN server. Hence, various gateway applications are designed to handle these application layer protocols. The common form for gateway application design in WiSEGATE is illustrated in Figure 3.3. These gateway applications are the combination of user agents that are handling the requests coming from the clients. Each user agent represents one remote client which connects to the server and waiting for the sensor service.

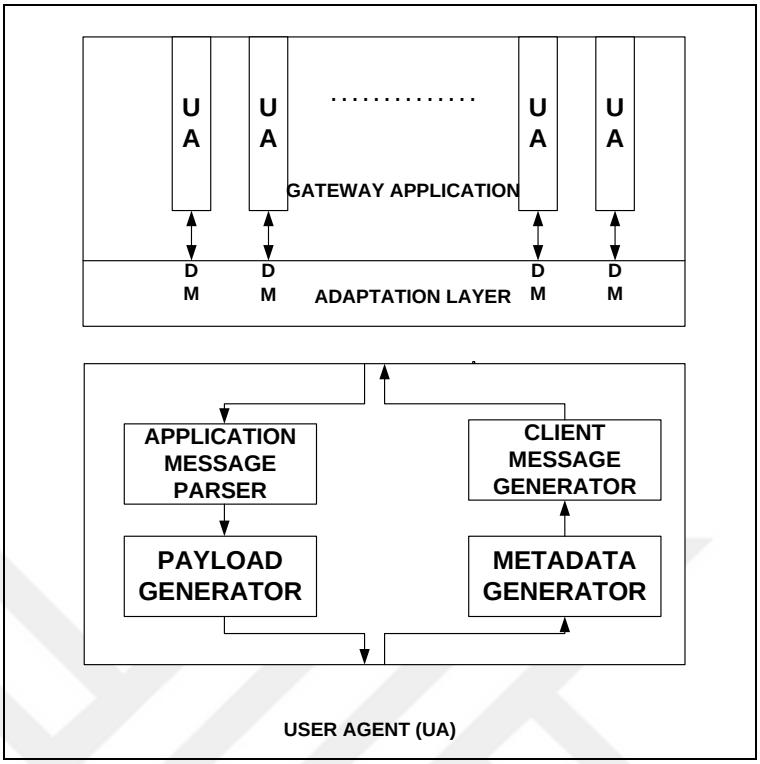


Figure 3.3. Gateway Application and User Agent

A common form of a user agent is illustrated in Figure 3.3. A user agent operates with two steps for handling incoming messages from the IP network. In the former, Application Message Parser Module in User Agent decomposes incoming application messages into the understandable parts. These parts are used for determining the service and its options. In this step, the User Agent specifies if this service requires any response from WSN or not. If so, in the latter, Message Generator Module in User Agent creates the byte coded query into a payload. In addition, this module adds a byte coded label to this payload. This byte-coded label is the *query id* obtained from the gateway application and the right user agent is specified with this label in a response. So, this field cannot be changed in a transmission of a packet. After creation of the payload, the user agent specifies the destination node. Then, the destination node information and the payload are sent to the adaptation layer.

User Agent operates with two steps for outgoing messages to the IP network. In the former, user agent extracts the byte coded payload from the datagram coming from the adaptation layer. Then, with using the byte-coded data in this payload, metadata generator

module specifies the data parts of the response and prepare a metadata table for the response message. In the latter, client message generator module generates a response message with this metadata table.

3.1.2. Gateway Application for Sensor Web

In this section, the gateway application specified for a sensor service which uses HTTP for presentation is discussed. This gateway application is composed of user agents which accept the client requests in the form of HTTP messages and present the sensor data with HTTP messages.

As illustrated in Figure 3.4, for the service and its options, HTTP message received from the web client (i.e. browser) is parsed and its GET query part is extracted. Then, from the GET query, some operational parts are extracted. Firstly, these operational parts are used to determine the service type. Then, with help of the service definition table specified for this requested service, the meaning of other parts are determined. These parts are then sent to the payload generator module for further operations.

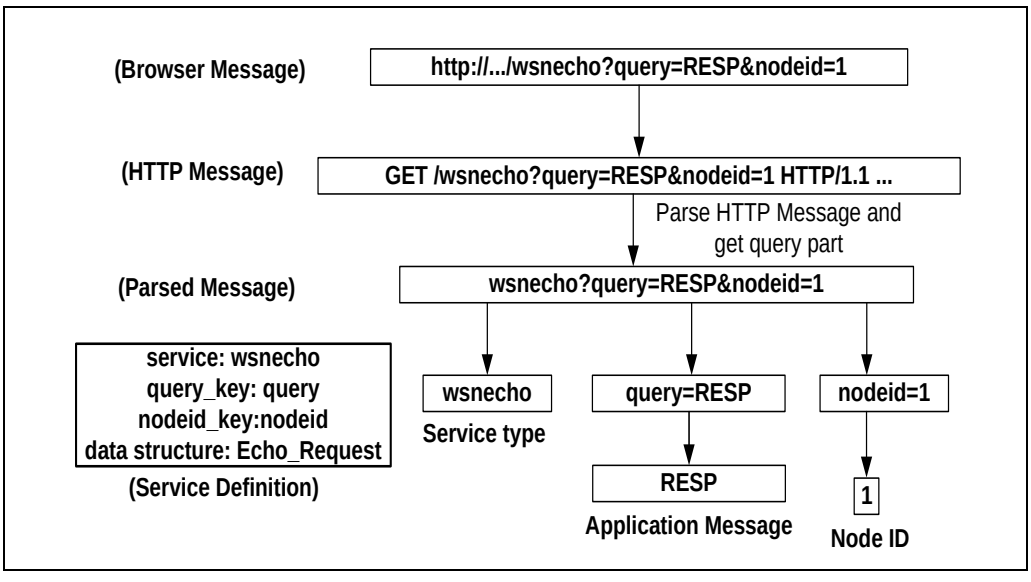


Figure 3.4. HTTP Message Parser in action. (Echo example)

When the necessary data parts are parsed in an HTTP message, the payload generator module creates the payload for the adaptation layer. As seen in Figure 3.5., this module

creates byte-coded payload in the sequence of data structure pre-determined for the request of the service. In this payload, it adds a byte coded query id to let the system to determine the right user agent in a response. After the payload is generated, a datagram manipulator is allocated at the adaptation layer and the payload is sent to this datagram manipulator.

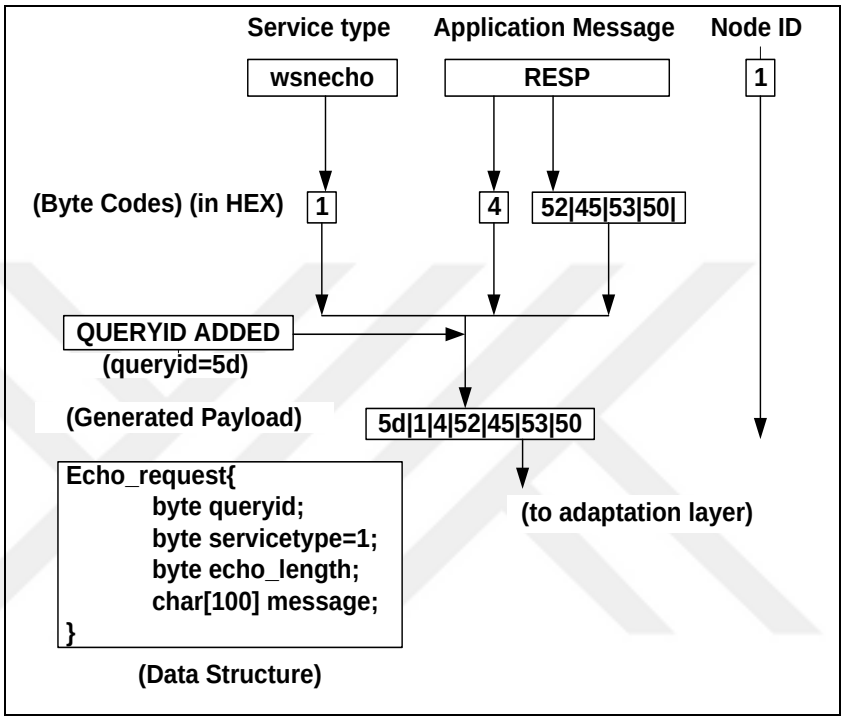


Figure 3.5. Payload Generator in action. (Echo example)

The response mechanism of the gateway application to HTTP requests is as follows: When the payload containing the response from the WSN node reaches to one of the user agents from the adaptation layer, for the first step, the data parts in the byte coded data are extracted by the metadata generator module. Then, these data parts are used to generate the metadata table for the further step in the data presenter module. This mechanism is illustrated in Figure 3.6.

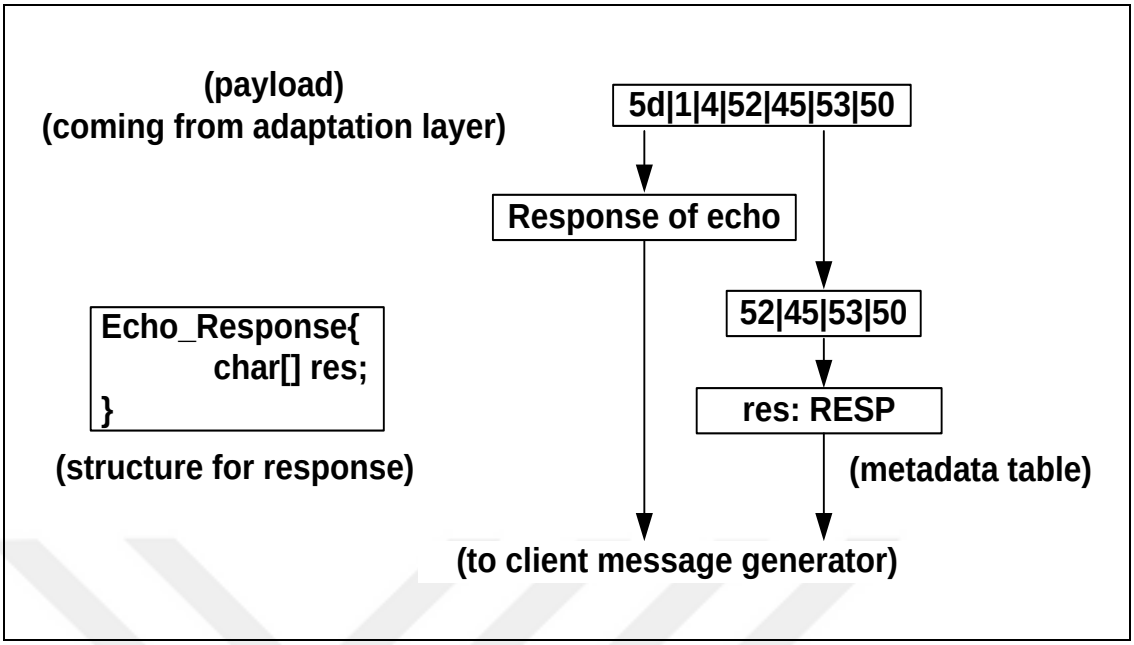


Figure 3.6. Metadata Generator in action. (Echo example)

After the metadata table and the response type are prepared, the data presenter module generates the HTTP message with using metadata table and response type as seen in Figure 3.7. The response type helps the data presenter module to generate right response HTML message. After the response message is generated in an HTTP message, it is sent to the remote client which is connected to this user agent.

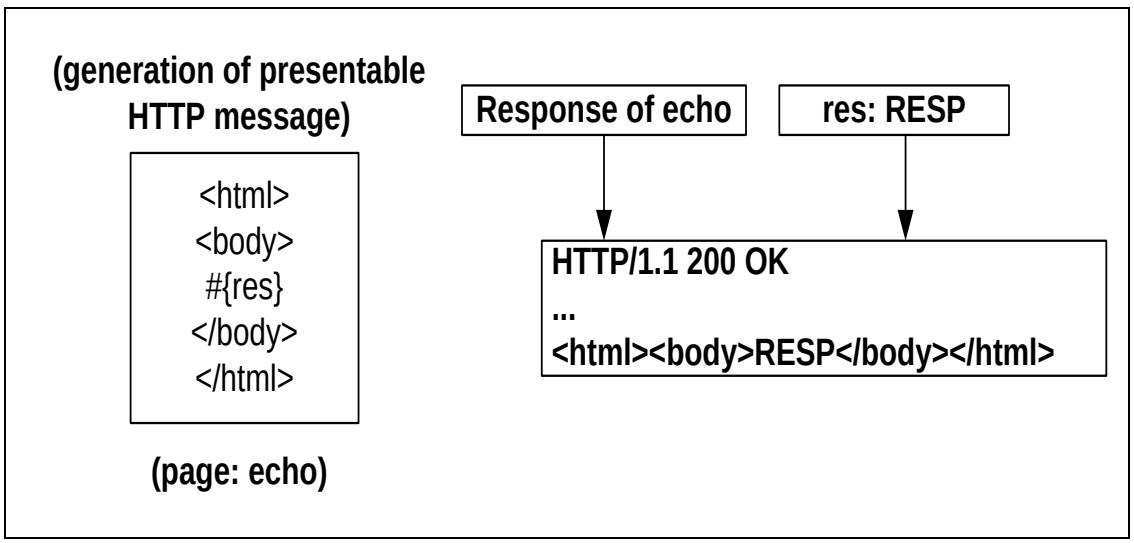


Figure 3.7. Data Presenter in action. (Echo example)

3.1.3. Adaptation Layer

The adaptation layer in WiSEGATE is used to transform packets to be understandable by sensor nodes and the gateway application. This layer has two functionalities to exchange data packets. The former is to encapsulate payload coming from the gateway application into a data packet understandable by the sensor nodes. The latter is to extract the payload in the packet coming from a sensor node. The packet format for these operations is determined by the interconnection protocol used between the sensor nodes and the adaptation layer itself. For example if the interconnection protocol defined in RFC 4644 [22] is used, then 6LoWPAN datagrams are used to carry the payload between the gateway and the destination sensor node.

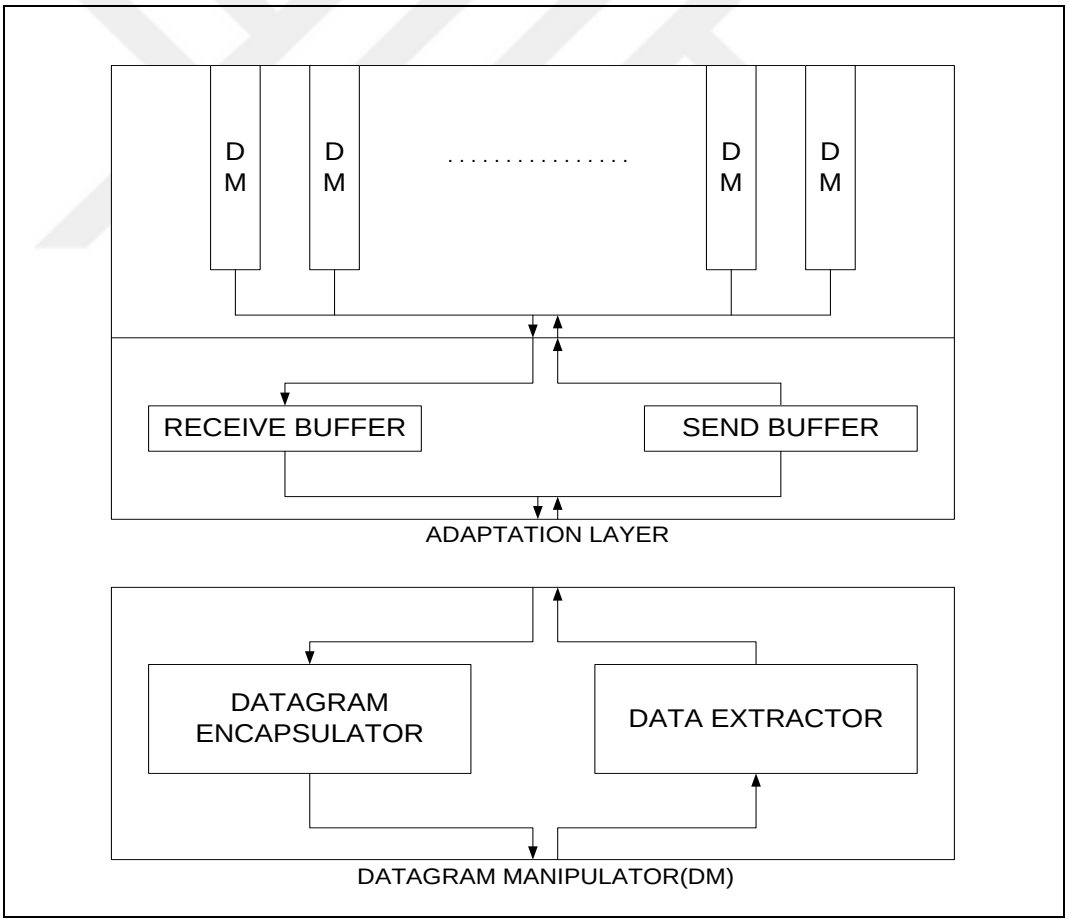


Figure 3.8. Adaptation Layer and Datagram Manipulator

As seen in Figure 3.8, the adaptation layer consists of datagram manipulators to create the datagram and extract the payload in the datagram. The datagram encapsulator module is used to create datagrams. This module determines the requirement for the routing of the payload over WSN using destination address. In addition, this module determines the necessity of the fragmentation of the payload. For this, it calculates the size of the payload. These criteria are used to create the appropriate datagram in this module. After the datagram is created, it is sent to a common send buffer by the datagram encapsulator module. Since the datagram manipulators operate concurrently with threads, the synchronization between these threads is required when they acquire the common send buffer. So a mutual exclusion mechanism is used for synchronization.

Whenever a datagram is taken from the send buffer at the adaptation layer, the payload of the datagram is extracted using datagram manipulator's data extractor module. Then it is sent to the right waiting user agent.

3.1.4. Adaptation Layer for Sensor Web

In this section, an adaptation layer design for the sensor web is presented. For the WSN tier of the web server, a 6LoWPAN based interconnection mechanism is used. Thus, the modules under the datagram manipulator handles 6LoWPAN standard.

As illustrated in Figure 3.9, firstly, the address of the destination node is looked up to determine if it is directly reachable or not. After that, the requirement of the fragmentation is determined. A suitable datagram format is chosen after these operations and the payload is encapsulated in a 6LoWPAN datagram. Then, this datagram is sent to a common sending buffer.

Whenever the datagram is taken from the common sending buffer, it is encapsulated in an Active Message packet and an 802.15.4 frame to be understandable by sensor nodes. Then it is sent to the WSN.

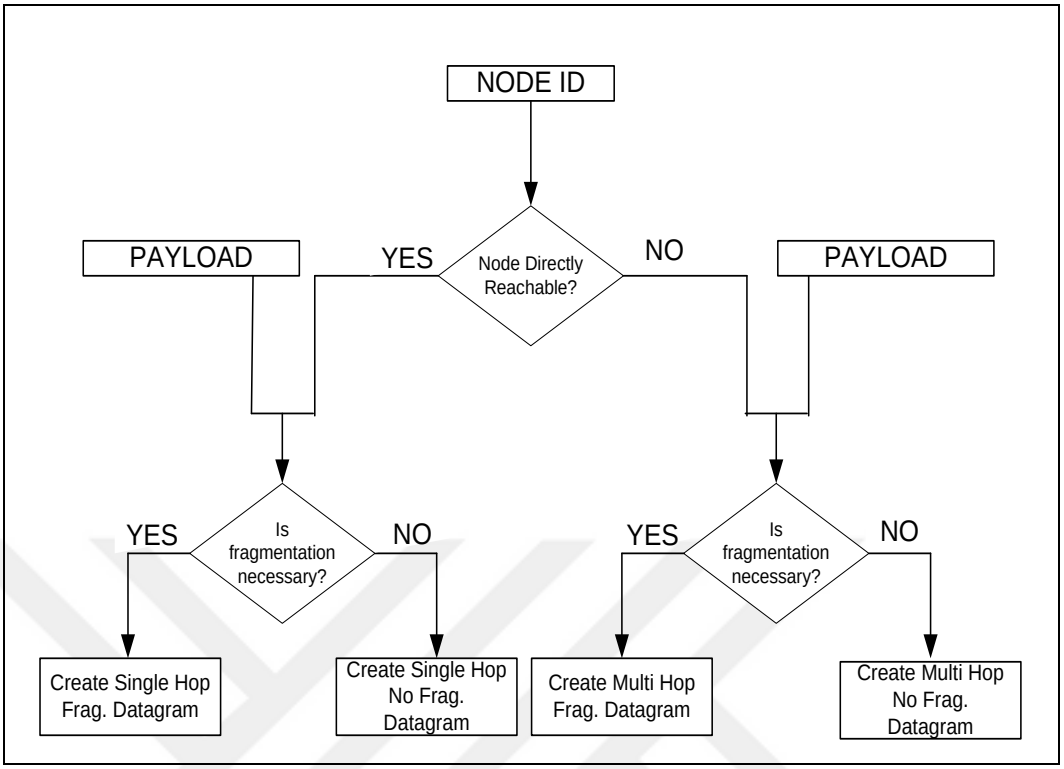


Figure 3.9. Flow diagram for choosing datagram type

For the messages coming from the WSN, the data extractor module firstly determines if the destination is the gateway or not. If so, it determines if the reasseambly of the data is necessary. Then it extracts the payload from the datagram and it creates the understandable data by the gateway application and forwards these data to the gateway application.

3.2. SENSOR NODES

In this thesis, sensor nodes are the sensor data acquisition devices for WiSEGATE. As mentioned earlier, operation of these devices are specifically called as the WSN tier. This tier has two operations. In the former, one sensor node in WSN tier only determine the requirement of the routing of a packet and they do not extract the sensor request in that packet. For this operation, the sensor nodes use the destination address information for routing. In the latter, a sensor node requires the request data in the payload for creating the response.

For the operations in WSN tier, two types of sensor nodes are defined. The former is named as the intermediate node and used for the first operation. The latter is named as the

destination node and used for the second operation. Any one of the sensor node in a WSN can be in one of the intermediate node or a destination node both. The destination node is determined with the destination address information in the datagram.

In the following sections, the design issues for the overall WSN tier is discussed. Firstly, the interconnection service is presented and secondly, the response mechanism of the sensor nodes is presented. Then overall service under the sensor nodes is discussed.

3.2.1. Interconnection Service

The overall interconnection scheme between the sensor nodes and the gateway node is illustrated in Figure 3.10. As seen in this figure, the sensor service and gateway application is abstracted from the interconnection service and the interconnection service is operated as a middleware to manipulate the datagrams. This middleware determines the requirement of the routing of the packets. If a datagram needs routing, payload is not extracted from the datagram and it is routed towards the destination. If the destination is reached, the payload is extracted from the datagram and sent to the sensor service or the gateway application.

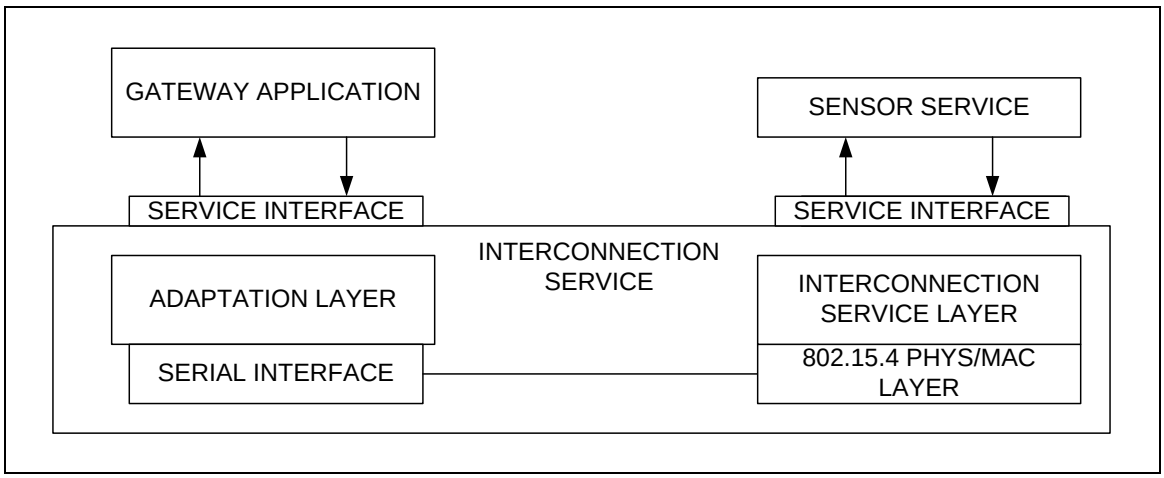


Figure 3.10. Interconnection scheme between Gateway and a WSN node

The interconnection service is operated on a lightweight interconnection service layer in sensor nodes. This layer is used for determining if any of the datagram is reached to the destination node or need of forwarding. As depicted in Figure 3.11, if the datagram reaches

the destination node, the data is extracted and sent to the sensor service application, else this datagram is routed to any one of the sensor nodes using some WSN routing policies.

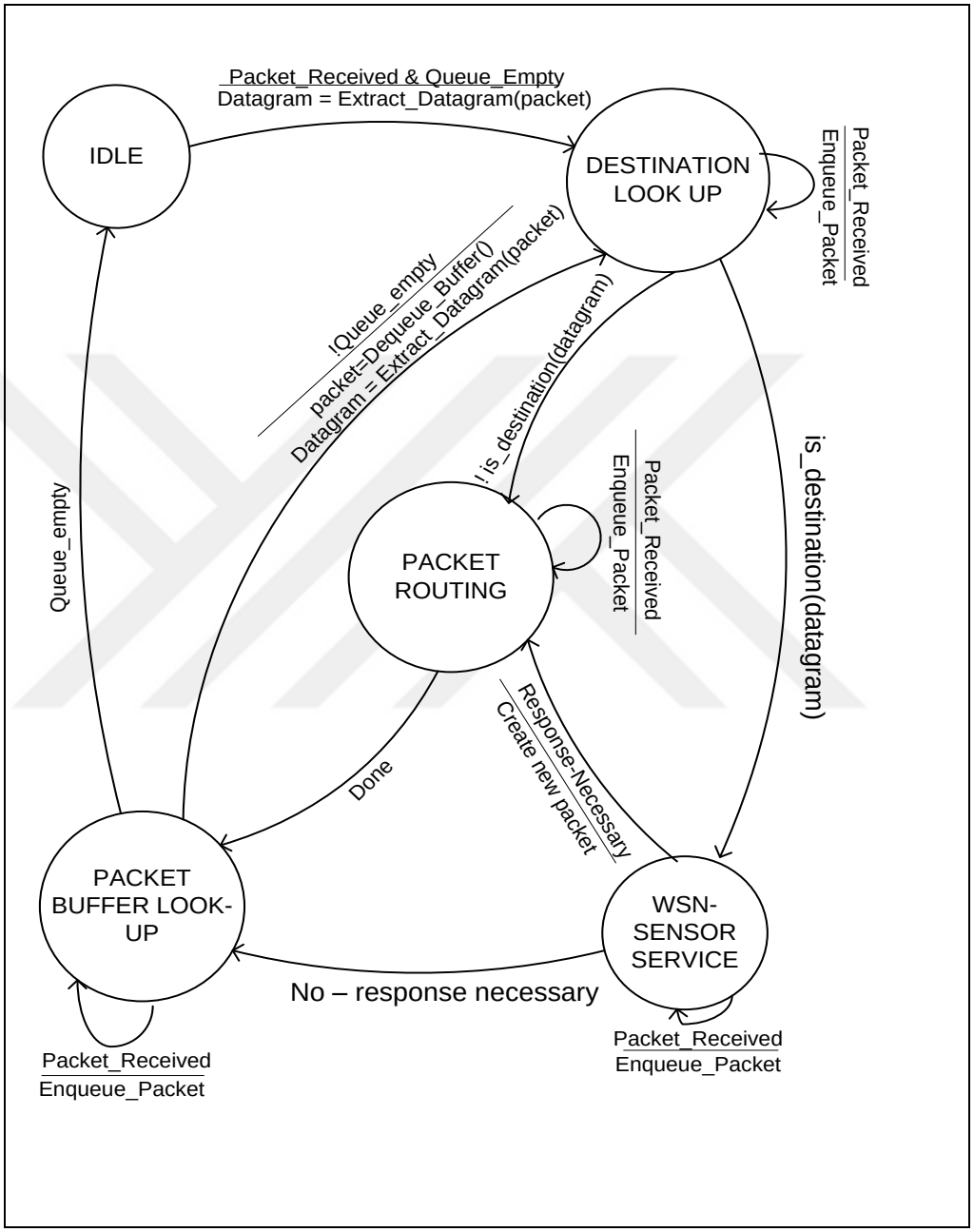


Figure 3.11. State diagram of the service model of a sensor node

3.2.2. Sensor Service

The sensor service is the application-based data acquisition mechanism on the sensor node. The byte-coded payload in the datagram is used to determine the service type required by

the remote client. With using a data structure pre-determined for the service, the service type is determined and then the sensor data is extracted from the sensor node. Then, a byte-coded response is created with this sensor data.

3.2.3. WSN Service

In the previous sections, we define two service types for the overall operation of WSN tier. With these criteria, two types of service stacks can be defined. The structure of these stacks is illustrated in the Figure 3.12. An intermediate node only operates the lightweight service layer to forward the datagram towards the destination. The sensor service layer is operated only on the destination node.

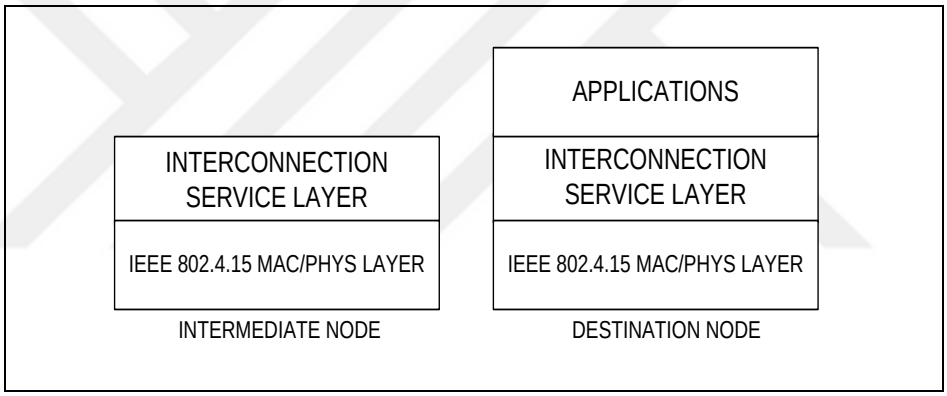


Figure 3.12. Sensor node stack models

4. TRAFFIC MODEL

This chapter provides the traffic model used for the evaluation of WiSEGATE. Firstly, the operational flow for the system and time costs affecting the performance of the system are given. Secondly, the derivation of the queue model for the system is provided. A request traffic model is derived to evaluate the stable behavior of the gateway. Then, an extension in the gateway model is provided to evaluate it in bursty traffic conditions. Finally, a service scheme is given for the back end network.

4.1. OPERATIONAL FLOW

WiSEGATE is used to connect multiple clients to the WSN. So the interconnection gateway is responsible for handling multiple requests coming from these clients. As described in Chapter 3, three tiers exists in the request/response mechanism for handling these requests. Presentation tier represents the client and in this tier, client creates the request for the presentation of the sensor data. After that, client sends this request to the gateway. The time cost for transmission of the request from client to the gateway is represented with t_{11}. In the service tier, gateway handles this request in two steps. In the former, the user agent handles the application message and creates the payload which has the request data. In the latter, adaptation layer creates the datagram which carries the request to a sensor node. The adaptation layer may enqueue this datagram in its send buffer (i.e., the waiting queue). t_2 represents the waiting time of a packet in this queue. When a datagram is ready, the gateway sends it to the destination sensor node. The time for transmission of this datagram from gateway to the destination sensor node is represented with t_{31}. In WSN tier, sensor node takes the request and prepares a response for it. After creation of the datagram which has the sensor response, the sensor node sends it to the gateway node. The time cost for transmission of this datagram from the destination node to the gateway is represented with t_{32}.

For the incoming messages from WSN, the gateway operates in two steps. In the first step, the response data is taken from the datagram by the adaptation layer. In the second step, it is sent to the gateway application and the presentable message is generated. This message

is sent to the client by the gateway. Transmission time cost of the response from gateway to the client is represented with t_{12}. Then this message is interpreted by the client application for the presentation of the sensor data.

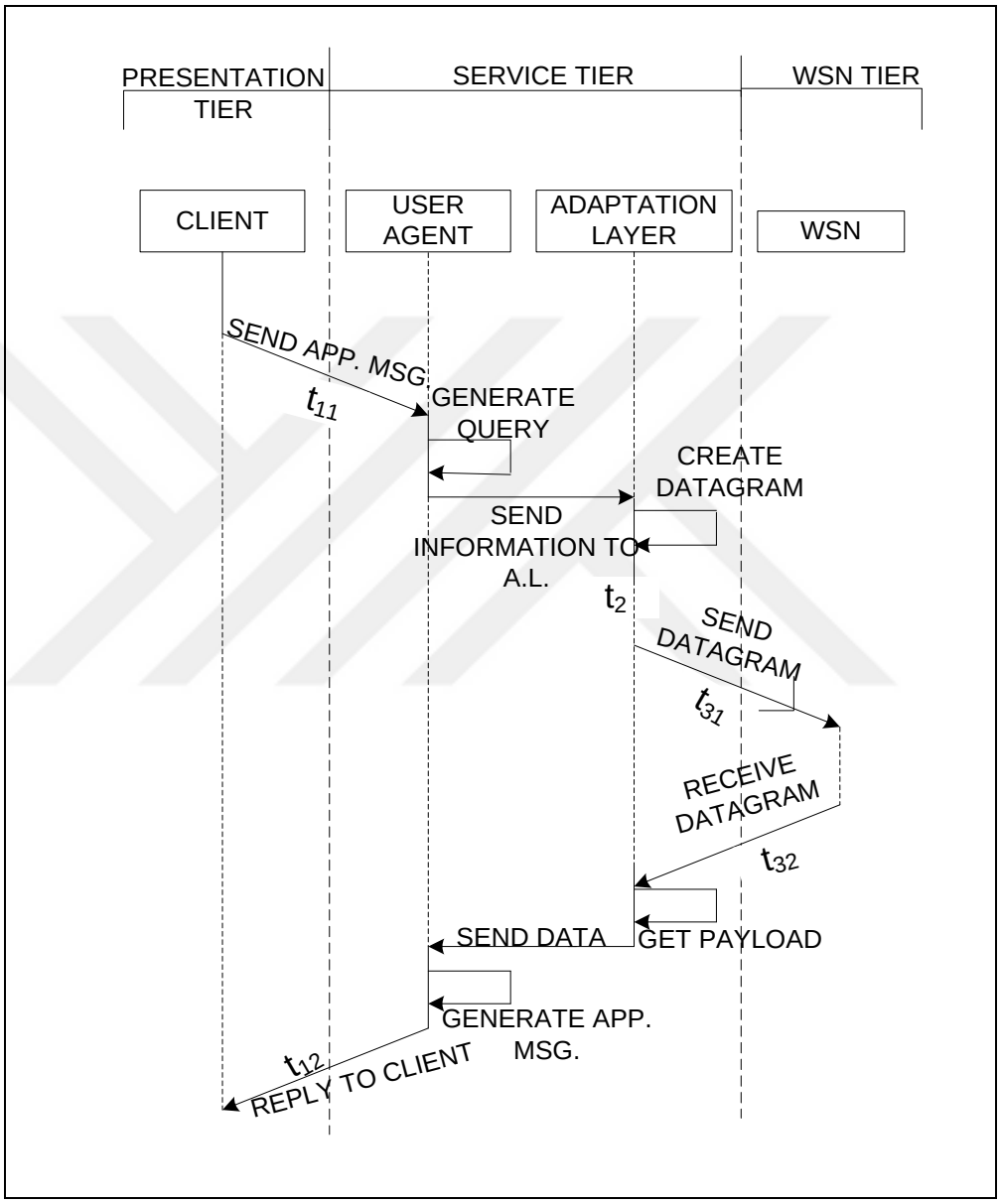


Figure 4.1. Timing sequence of request/response system

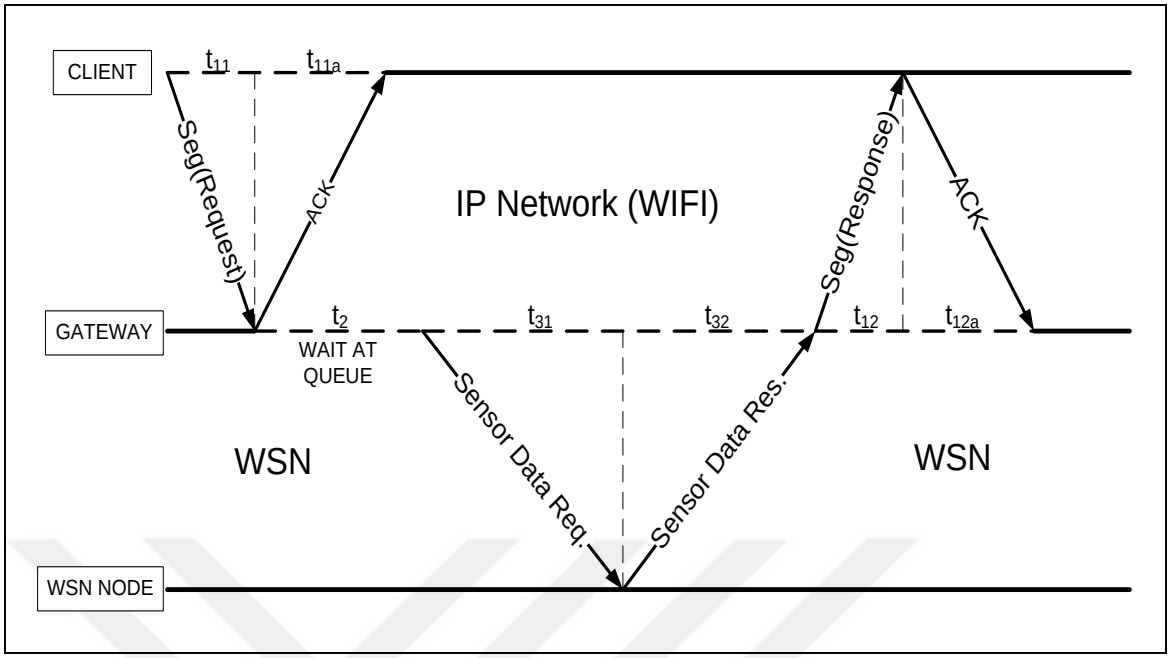


Figure 4.2. Timing diagram for a request

Figure 4.1 and 4.2 illustrates the sequence of the steps and the time costs of the request/response mechanism. With the help of the steps of the operational flow, we can derive a connection scheme for the interconnection system as illustrated in Figure 4.3.

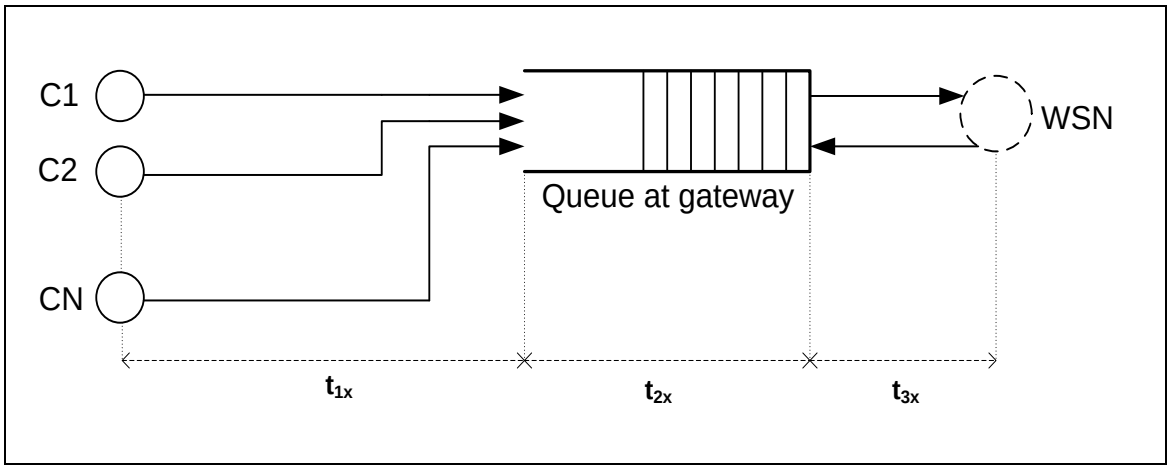


Figure 4.3. Overall system model and timing costs

In the system, whenever a request from a client arrives the gateway, if another request is on transmit within the WSN, the request packet is enqueued to the waiting queue. Since a stop and wait service is operated for handling the packets in the queue, the dequeue of the

request packet in the front is done when a reply packet is received from the WSN. As a server, the gateway operate with a very slow network at its backend. So the service rate might be very low and request packets are enqueued to wait for long times. In addition, the gateway should handle requests coming simultaneously from clients, so the requests shall compete for the same WSN node and may wait in the resource queue, if the WSN is busy.

Considering that t_{11}, t_{12}, t_2, t_{31} and t_{32} are independent random variables as given above, the mean response time for a request can be represented as follows:

$$\mathrm{E}[t_r] = \mathrm{E}[t_{11}] + \mathrm{E}[t_2] + \mathrm{E}[t_{31}] + \mathrm{E}[t_{32}] + \mathrm{E}[t_{12}] \tag{4.1}$$

4.2. TCP OPERATION

From TCP perspective of transmission, the segments for the request and response travel in a manner illustrated in Figure 4.4.

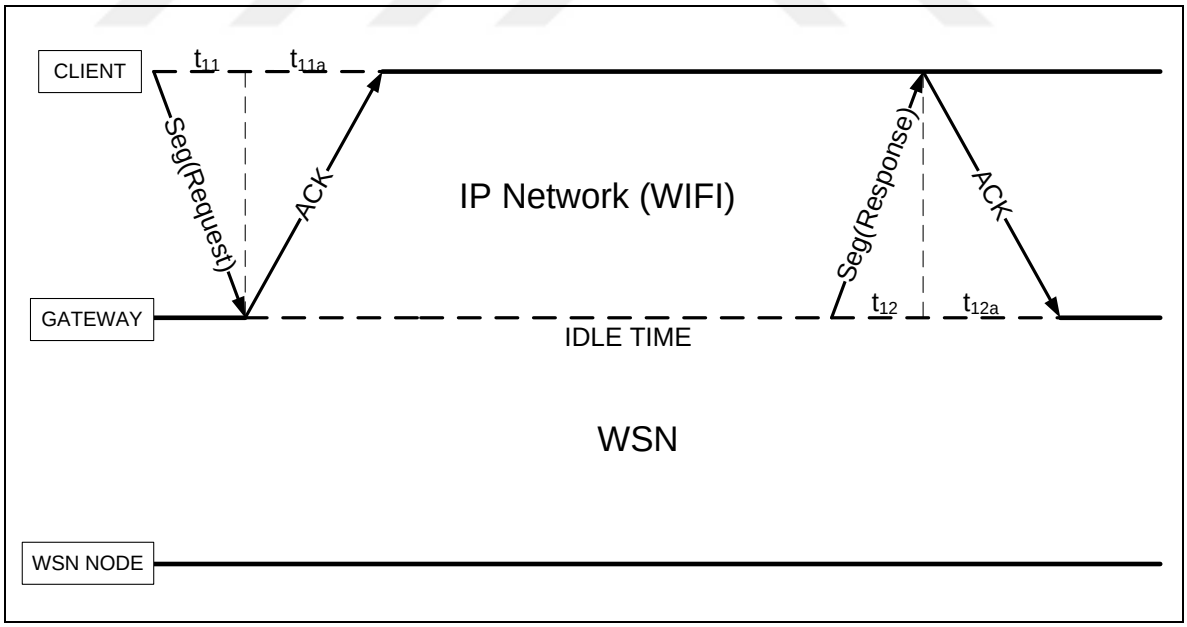


Figure 4.4. TCP operation for segments

The timing sequence for the TCP is defined by the steps below:

- When a client wants to send a request to the WSN, it uses TCP/IP; this TCP connection is handled by the gateway machine. In client's TCP layer, the request is encapsulated in a TCP segment and passed to the MAC layer for transmission.
- When the segment which has the request for the sensor data reaches to the gateway's TCP layer, an acknowledgment for this segment is sent immediately to the client. This segment is deleted and the application data is sent to the gateway application.
- There is no operation of TCP for this connection at the gateway machine till any one of the response for the client reaches to the gateway. Gateway determines the right outgoing client which the response should send to with the label id given by the user agent. Then the gateway's TCP encapsulates the response in a segment and this segment is sent to the client. An acknowledgement for this segment is sent immediately to the gateway.

We can elaborate on the steps of TCP operation given above with some arguments. The first is that each TCP connection between the gateway and the clients is handled independently. So in the perspective of TCP, request/response mechanism maintained for a client does not affect the request/response mechanism maintained for other clients. The second is that, for a periodic request/response mechanism of a TCP connection, idle times occur as seen in Figure 4.4. Considering the operation of independent TCP connections with immediate ACKs by the gateway, these idle times do not affect the timeout mechanism of TCP connections.

4.3. QUEUEING ANALYSIS

We can model the gateway as M/M/1 queue since all incoming requests of simultaneous clients join a single waiting queue. So, this section gives analysis of this queue in our approach.

Since a stop and wait mechanism is used for the service of requests in WSN, a request packet is enqueud to the queue in the gateway when the WSN is busy. So, the best case for a request is that, it is in front of the queue and immediately gets a service from WSN. The worst case is that it waits all the packets in the queue to be served from the system.

Therefore, the average wait time in the queue can be found by the followifng in which *n* represents the average number of packets in the queue and t_2 is the waiting time in queue:

$$\mathrm{E}[t_2] = \frac{\left(\mathrm{E}[t_{31}] + \mathrm{E}[t_{32}] + \cdots + n(\mathrm{E}[t_{31}] + \mathrm{E}[t_{32}])\right)}{n} = \frac{(n+1)(\mathrm{E}[t_{31}] + \mathrm{E}[t_{32}])}{2} \tag{4.2}$$

As illustrated in this formula, E[t_2] is dependent to E[t_{31}] and E[t_{32}] because dequeue of the packets requires transmission of a packet over WSN and reach back to the gateway. The service rate for the queue is dependent to these timing costs so it can be found by the following formula:

$$\mu = \frac{1}{\mathrm{E}[t_{31}] + \mathrm{E}[t_{32}]} \tag{4.3}$$

The second element for the queueing analysis is the arrival rate λ. It is clear that this value is dependent to the request traffic rate of the network. However, finding an arrival rate for a stable queueing system is critical for the performance analysis of the interconnection model because high arrival rate causes packets to pile up in the queue and so increase of *n*. Higher *n* means higher average response time of the system.

An ideal λ for the gateway queue can be obtained using Little's law. The average waiting time in the queue and service rate for the system has already been obtained above. Thus, the ideal λ can be found using the following formula:

$$\mathrm{E}[t_2] = \frac{1}{\mu\text{-}\lambda'} = \frac{(n+1)(\mathrm{E}[t_{31}]+\mathrm{E}[t_{32}])}{2} = \frac{1}{\frac{1}{(\mathrm{E}[t_{31}]+\mathrm{E}[t_{32}])}\text{-}\lambda'}$$

$$\lambda' = \frac{(n-1)}{(\mathrm{E}[t_{31}]+\mathrm{E}[t_{32}])(n+1)} \cong \frac{1}{(\mathrm{E}[t_{31}]+\mathrm{E}[t_{32}])} = \mu \tag{4.4}$$

$$\lambda' = \mu$$

With this formula, an arrival rate value for the stable behavior of the system can be obtained. If any arrival rate is smaller than λ', then the gateway is stable, else packets pile up in the queue.

4.4. REQUEST MODEL

The stable condition for the interconnection system is defined with a queueing analysis in the previous section. In this subsection, a traffic model for the requests is defined for the analysis of our system. In addition, a model for a bursty traffic condition is given using the traffic rate of the system.

A typical web traffic can be modeled as a Poisson-based CBR request traffic. Thus, for the multi client web traffic model, the PMF values obtained from the Poisson process are assigned to each client in the system. This value represents the request rate distributed to each one of these clients. Thus, if PMF value for a client is f_i, the arrival rate of client i for a stable traffic is represented with:

$$\lambda_i = f_i \lambda' \tag{4.5}$$

With using λ_i value obtained from this formula, the sending period of a client can be found as follows:

$$T_i = \frac{1}{\lambda_i} \tag{4.6}$$

An expansion for λ_i can be done for traffic rates. Using a traffic rate value *r*, the request sending rate of a client can be represented with:

$$\lambda_i = \lambda' f_i \, r \tag{4.7}$$

When *r*<1, the gateway can handle packets without a pile up in the queue. Thus, the system is stable. However when *r*>1, the packets starts to pile up in the queue as time flows. This means a bursty request traffic for the queueing system.

In Formula 4.4, we assume that *n* is ignorable and λ' is dependent on only μ. There are two cases for this. Firstly, when *r*>1, *n* is very great, thus, λ' will be very close to μ. Secondly, when *r*<1, the birth-death period of the queue is more reasonable and all the packets in the queue can be served with a short periods of time. So, with a small error, we can predict the average response time of the system.

4.5. QUEUE FOR BURSTY TRAFFIC

Traffic rate value *r* is used to determine the traffic condition for the gateway. If *r*>1, the request traffic is bursty for gateway and packets pile up in its waiting queue. Therefore, average response time of the request/response system increases as time goes by due to waiting packets in the queue. In this section, we examine packet handling mechanisms of the proposed approach in such bursty traffic conditions.

When a bursty traffic exists in the system, *n* increases with the time. Since, the average response time depends on this value as expressed in the Formula 4.2, average response time increses with the time. If we use an infinite queue to handle simultaneous packets in the gateway, *n* reaches to the infinity with a continous bursty traffic in the system which results higher response times of the system. Thus, a bounded size queue should be used to limit *n* and average response time.

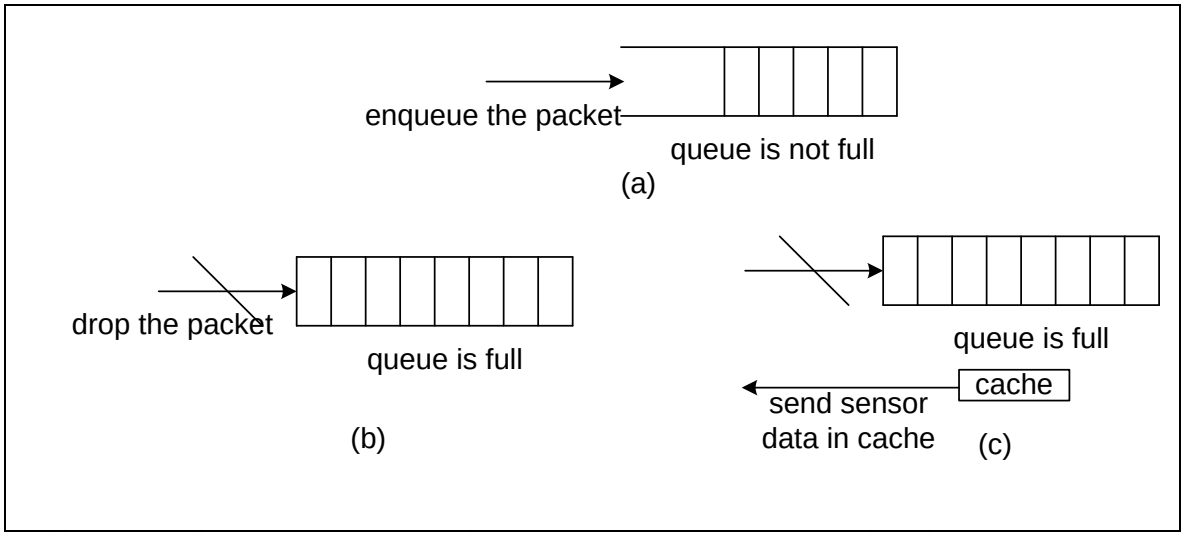


Figure 4.5. Queue handling methods for bursty traffic

Figure 4.5. illustrates three bounded queue handling mechanisms used in this thesis. In the first *(a)*, queue is not full and the gateway can enqueue a new request packet. In the second *(b)*, when the queue is full, gateway drops new incoming request packets. This means loss of any one of requests of a client. In the third *(c)*, to avoid loss of requests, a caching mechanism is introduced. In this mechanism, the recent sensor data acquired from WSN is recorded temporarily in the gateway (i.e., in a temporary cache). So when the queue is full, rather than dropping new incoming request packet, the gateway sends an immediate response with the sensor data in the cache to the corresponding client.

4.6. SERVICE MODEL OF WSN

In this thesis, WSN service is modelled as the single server of the queuing system. Hence, it operates with service rate μ where inter service time is exponentially distibuted. This model gives us a backend network that returns a response packet to the gateway in every $1/\mu$ time period in average. It can be argued that this assumption is over simplistic considering the dynamics of a real WSN. For a more realistic service rate of the queue, service time traces could be obtained with a WSN simulation scenario.

5. PERFORMANCE ANALYSIS

For the performance analysis of WiSEGATE, firstly we performed comprehensive simulation tests using OMNET++ discrete event simulator [55] to evaluate the performance of the gateway with the traffic model defined in Chapter 4. Consequently, we conducted experiments to evaluate the proposed model in a real test environment. This chapter presents the analysis of these performance tests.

5.1. PERFORMANCE METRICS

In performance analysis of WiSEGATE, the following performance metrics are used:

- *Response Time:* Response time is the time slice lagged between one of clients request and the corresponding sensor data response to this request. In Chapter 4, the response time is represented with t_2.
- *Throughput of WSN:* Average rate of successful packet delivery over a WSN channel in bits.
- *Goodput:* Average rate of successful application data delivery (i.e., the sensor data response) over clients in bits.
- *RTT (Round Trip Time):* Round Trip Time for a TCP connection is the difference between the time at which a segment is sent and the time at which an acknowledgement (ACK) arrives for the data in that segment [56]. In a send/response system, this value cannot be higher than theoretical request sending period of a client for reasonable response times. For example, the maximum expected RTT value of Service-2 Model is 480ms for N=40. If a RTT value for a request/response is greater than this value, we can consider that in front-end wireless network, the packets are dropped due to high collisions or high packet errors.

5.2. SIMULATION ENVIRONMENT

For analysis for the scalability of the gateway model with increasing number of simultaneous clients and wireless transmission errors, we conducted comprehensive tests

in a simulation environment. For these simulation tests, we used OMNET++ and its inet-manet simulation package.

In these tests, the gateway node and clients are on the same WLAN and communicate with each other via WiFi. A sensor node is connected at the back-end of the gateway. Table 5.1-5.3 shows the simulation parameters for these tests. In analysis of these tests, *N* represents the number of simultaneously requesting clients in the system.

Table 5.1. WIFI parameters used in simulations

Parameter Name	**Value**
WIFI Type	802.11g
WIFI Bandwidth	2 Mbps
WIFI Carrier Frequency	2.4 GHz
SNR Threshold	4 dB
Sensitivity	-85 dBm
Base Noise Level	-110 dB
Channel Model	Rayleigh
WIFI PER	0.05 -0.6
MTU for Linklayer Frames	1500B

Table 5.2. TCP parameters used in simulations

Parameter Name	**Value**
Maximum Segment Size (MSS)	536B
Congestion Control Protocol	TCPReno
Connection Model	Asynchronous

Table 5.3. Application parameters used in simulations

Parameter Name	Value
Request Size	30Bytes
Send Queue Size	Infinite – (1 2 4 8 16 32 64) KBytes

For the service model of the WSN, the first group of tests used the simplistic generic service model defined in Section 4.6 and we call this as Service-1 Model. In the second group of tests, for a more realistic analysis, we performed external simulations for the transfer of a single echo packet to the sensor node using TOSSIM simulator [57] and obtained a sequence of service time traces. The histogram of these service time values is given in Figure 5.1. We then fed these service time values to the gateway simulator as WSN response times for TCP requests. Table 5.4 shows the parameters of these external simulations. We call this service model as Service-2 model.

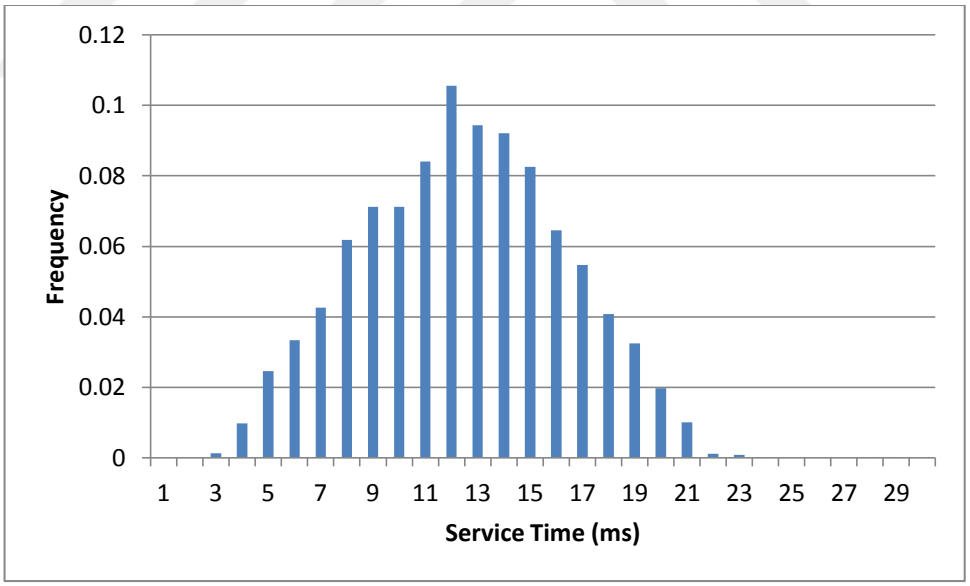


Figure 5.1. Histogram for service time traces obtained from a WSN Simulation

Table 5.4. WSN parameters

Parameter Name	Value
Bandwidth	250kbps
Radio Attenuation	30 dB
Scheduling Policy	Stop and Wait
MTU for Link Layer Frames	104 Bytes
Base Noise Level	-98 dB

Following subsections describe the analysis of the simulation tests done for evaluation of WiSEGATE. Section 5.1.1 presents the analysis of the results of the tests performed with Service-1 Model. In Section 5.1.2, we describe the analysis of results of the scalability tests performed with Service-2 Model.

5.2.1. Service Model-1 Tests

This section presents the analysis of the results obtained from tests with generic service model in stable traffic conditions. These tests have been done to observe the performance of the gateway in different service rates and different WiFi PERs.

5.2.1.1. Effect of Service Rate

This analysis focuses on the effect of the increasing service rate of the backend network. So, we have conducted tests with 64 kbps, 128 kbps and 250 kbps as the data rate of the back end network.

As depicted in Figure 5.2 and Figure 5.3, the average goodput and throughput of WSN decreases as N increases and these values are relatively lower as increasing of the service time of the back end network.

Figure 5.2. Average Goodput results of Service-1 Model Tests

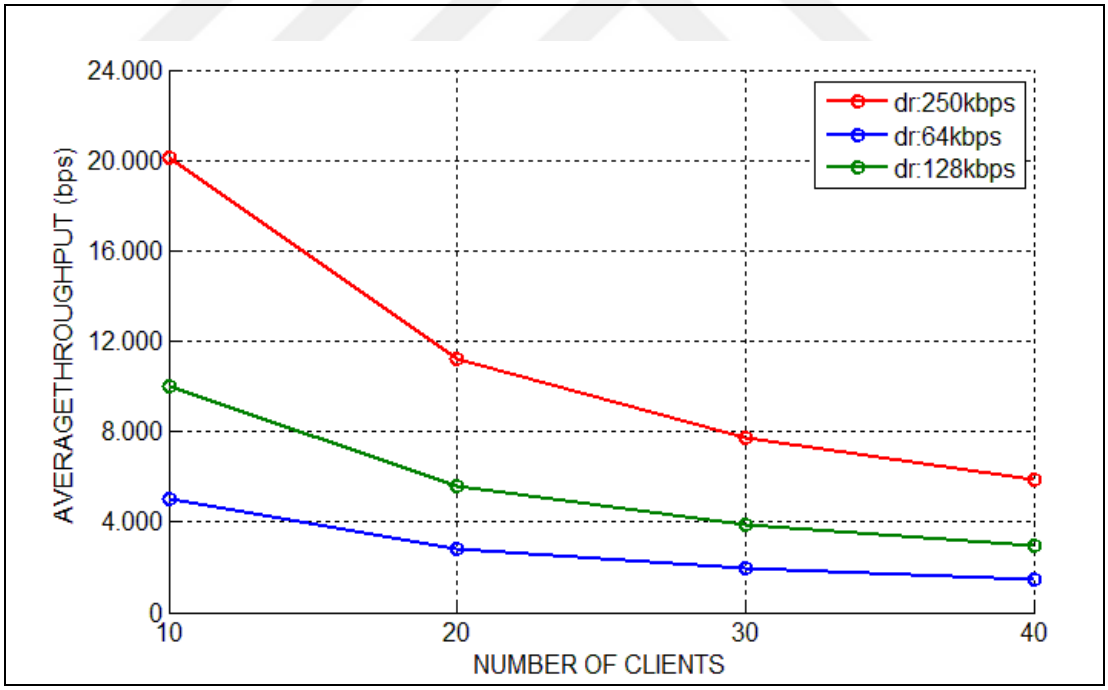


Figure 5.3. Average Throughput Results of Service-1 Model Tests

5.2.1.2. Effect of WiFi PER

To analyze the effect of WiFi PER in different data rate of the backend network for a stable system, several tests have been performed. For these tests, WiFi PER is increased

from 0.1 to 0.6. The first group of tests have been done with 64 kbps service rate and the second group of tests have been done with 250 kbps service rate.

When the service rate is 64 kbps, for a stable traffic, clients send their request in relatively long periods of time. Hence, WiFi MAC of the gateway and clients can handle this request traffic entirely even in higher PERs. So the system has constant average goodput with WIFI PERs up to 0.6 as depicted in Figure 5.4 even it decreases with increasing *N*.

As illustrated in Figure 5.5, when the service rate is 64kbps, a dramatic increase in average response time is not observed as increasing WIFI PER and *N* up to 40. So the average response time of the system is not higher than 480ms which is the theoretical average request sending time period of clients when N=40 and service rate is 64kbps. Hence, a stable request traffic for a slow service rate leads reasonable average response times.

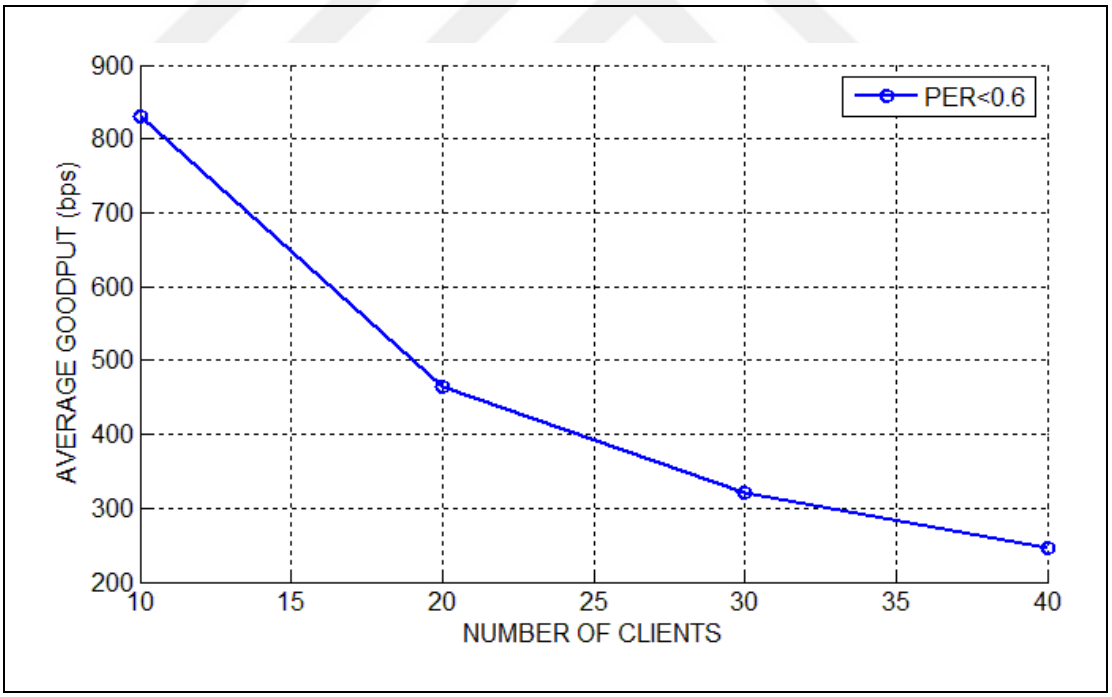


Figure 5.4. Average Goodput results of Service-1 Model Tests with different WiFi PERs and service rate is 64 kbps

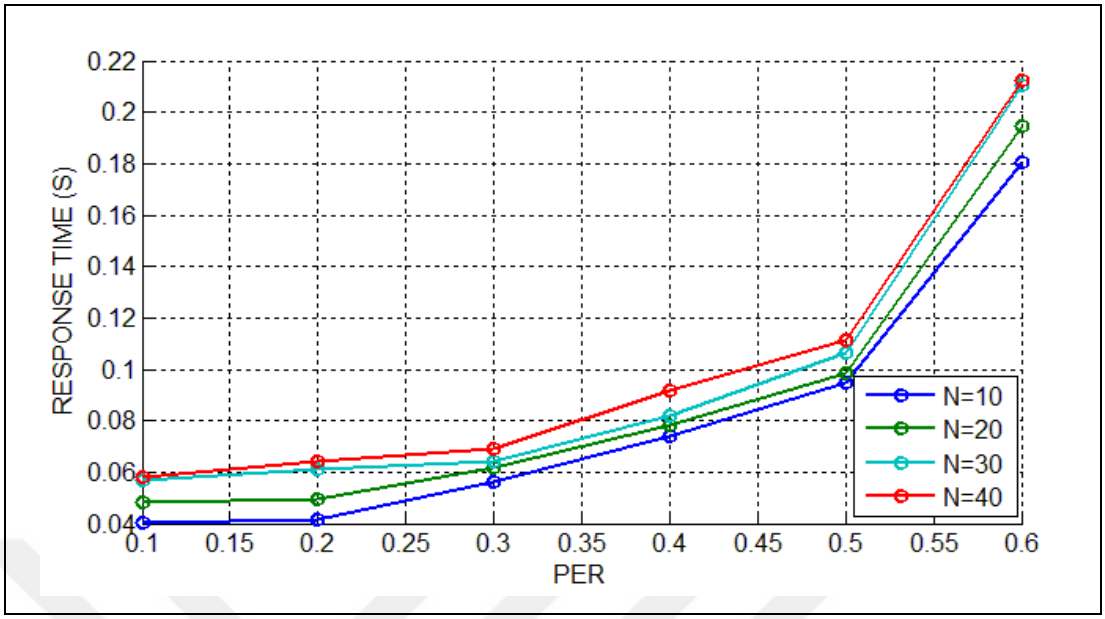


Figure 5.5. Average Response Time results of Service-1 Model Tests with different WiFi PERs and service rate is 64kbps

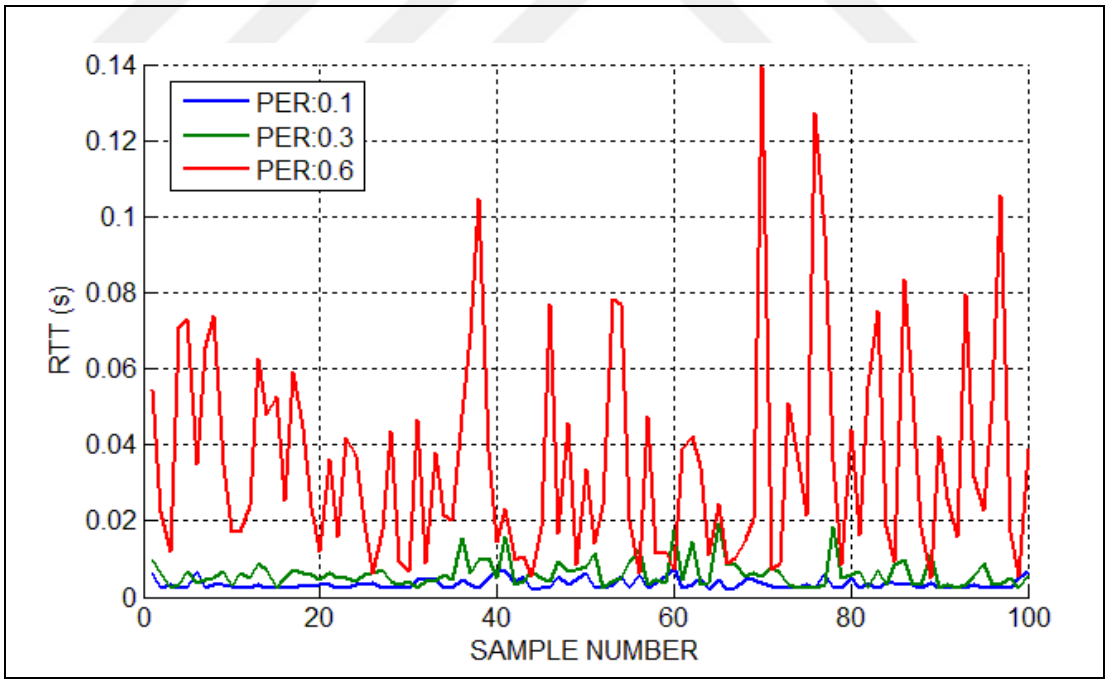


Figure 5.6. Consecutive RTT samples obtained from a client when N=40 and data rate is 64 kbps

As depicted in Figure 5.6, although consecutive RTT values obtained from a client increase and start to deviate in a long interval with increasing of WIFI PER, the maximum

value of RTT is not higher than 480 ms which is the maximum expected RTT value when N=40 and service rate is 64kbps. This is due that underlying service of TCP can handle the low request traffic of the system without higher RTTs.

When the service rate of the backend network is 250kpbs, even for a stable system, the clients send their requests in relatively short periods of time. This leads a denser traffic condition on the network. Within this traffic condition and increasing WIFI PER, more TCP segments are lost in wireless channels and they are sent again by TCP layer. Thus, as seen in the Figure 5.7, the RTT on the connections are extremely high and sometimes they are greater than 120ms which is the maximum expected RTT value for N=40 and service rate is 250kbps.

Although the service rate is high and this situation leads relatively high goodput, as illustrated in the Figure 5.8, the goodput of the system decreases while increasing of WIFI PER. High RTT values lead in decrease of the goodput of the system.

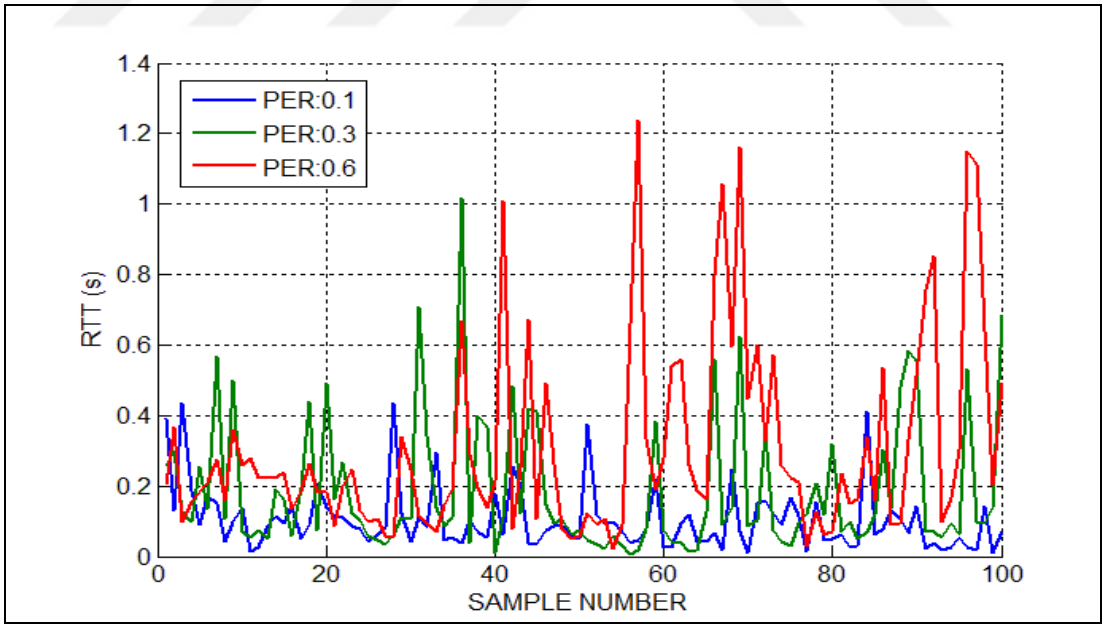


Figure 5.7. Consecutive RTT samples obtained from a client when N=40 and data rate is 250 kbps

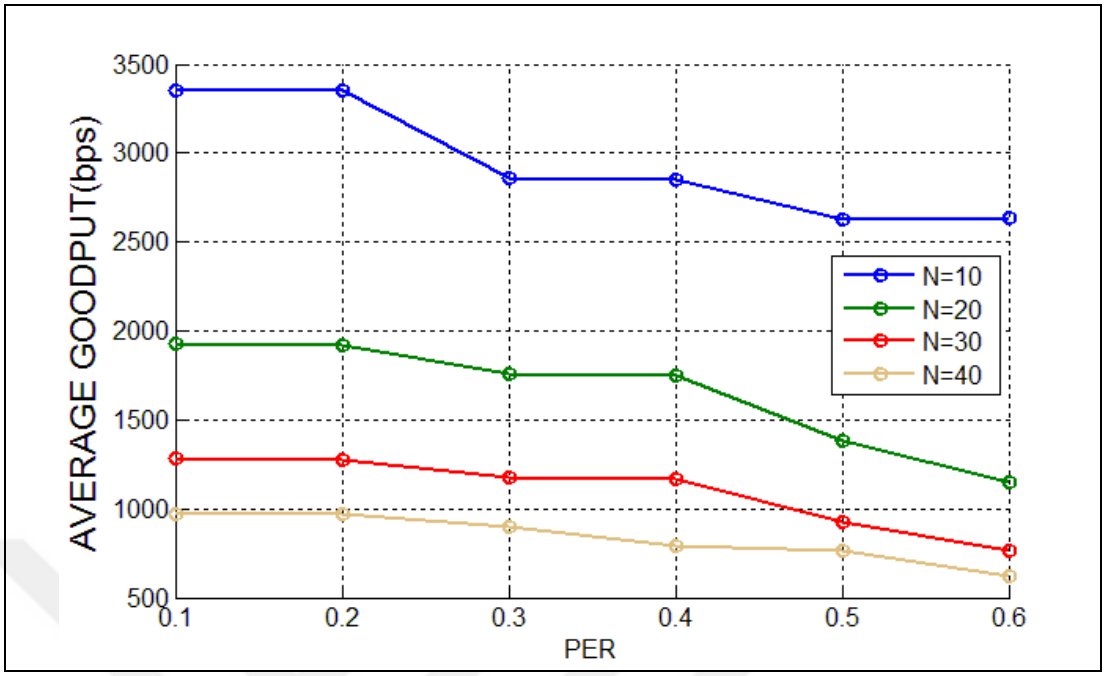


Figure 5.8. Average Goodput results of Service-1 Model Tests with different WiFi PERs and service rate is 250 kbps

5.2.2. Service Model-2 Tests

5.2.2.1. Scalability Tests With Stable Traffic Conditions

We have conducted two sets of scalability tests for the performance analysis of the WiSEGATE in stable traffic conditions. In former set of test, we have focused on the scalability of the gateway in higher WiFi PERs. So these tests involve the experiments with increasing WiFi PER up to 0.6 and N limited to 40. In latter set of tests, we observe the scalability of WiSEGATE with increasing N in reasonable WiFi PERs. So, these tests have been performed with WiFI PER up to 0.2.

Figure 5.9 illustrates the average goodput results of clients obtained from first set of tests. As depicted in this figure, the goodput of the system is constant when the WiFi PER is less than 0.4. This means that the gateway can handle entire request traffic generated by the clients soundly within WIFI PER up to 0.4.

Figure 5.10 and Figure 5.11 illustrate the average response time results obtained from first set of tests. The average response time increases nearly linearly as N increases up to 40

when WiFi PER is less than 0.4. This is due to the periodic and stable request traffic in the system and gateway can handle this traffic soundly. Within these PERs, the average response times are not higher than the expected maximum value for response time as illustrated in Figure 5.10. When WiFi PER is 0.5, a dramatic increase in average response time is observed as depicted in Figure 5.11. This means that the request traffic of the network cannot be entirely handled by the WiFi MAC of the gateway and TCP layer of the gateway resends the segments again and again for false timeouts because of high packet drop rate and high collusion in the wireless transmission media.

Figure 5.12 illustrates the average RTT results obtained from first set of tests. When WiFi PER is 0.5, average RTT values are extremely higher than values when WiFi PER is less than 0.5. In addition, consecutive RTT values obtained from a client is larger and deviates in a longer interval when WiFi PER is 0.5 as depicted in Figure 5.13. The overall scalability analysis for the first group of tests reveal that when the number of simultaneous clients is limited to 40, the gateway model can be applied with WiFi PER which is less than 0.5.

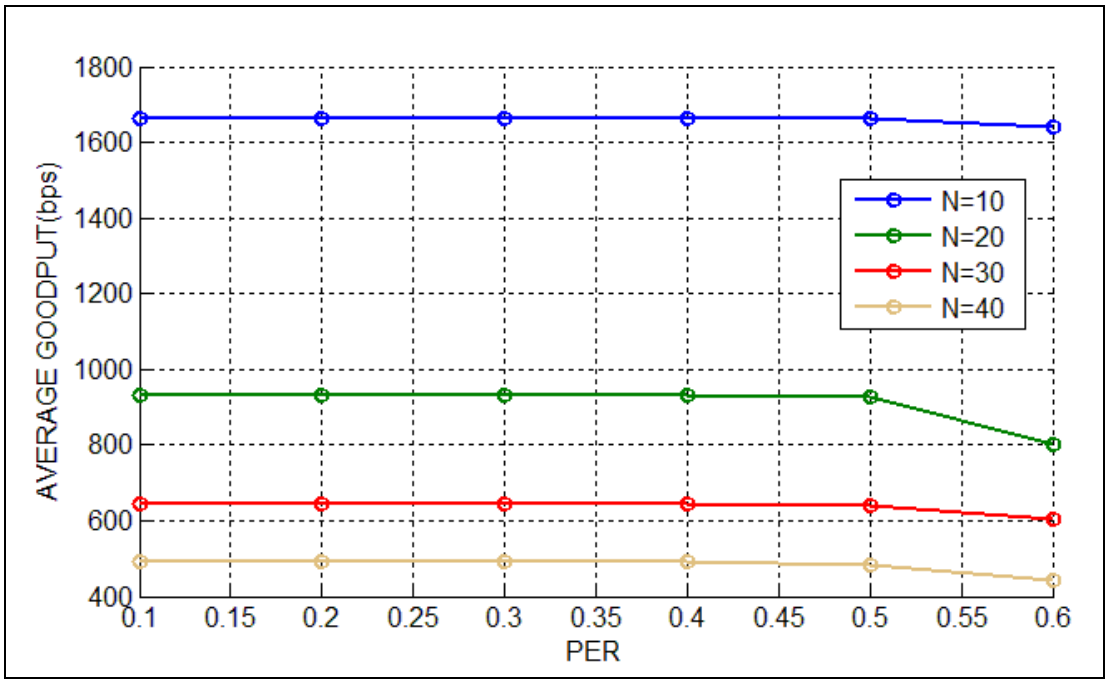


Figure 5.9. Average Goodput results of Service-2 Model Tests with different WiFi PERs

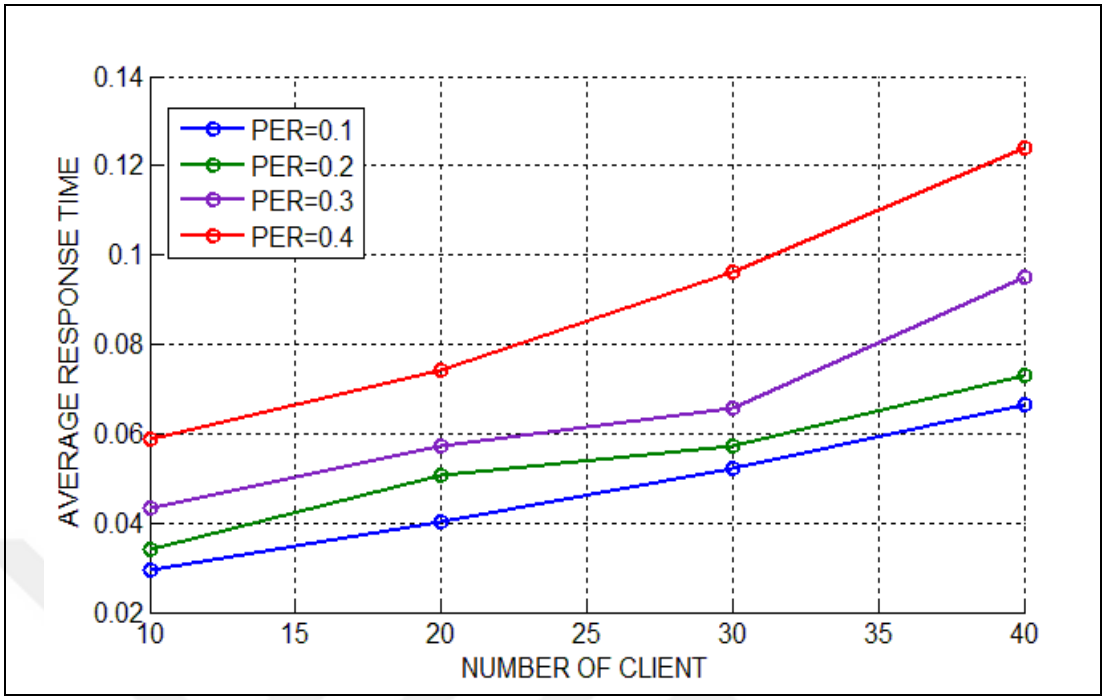


Figure 5.10. Average Response Time results of Service-2 Model Tests when WiFi PER is up to 0.4

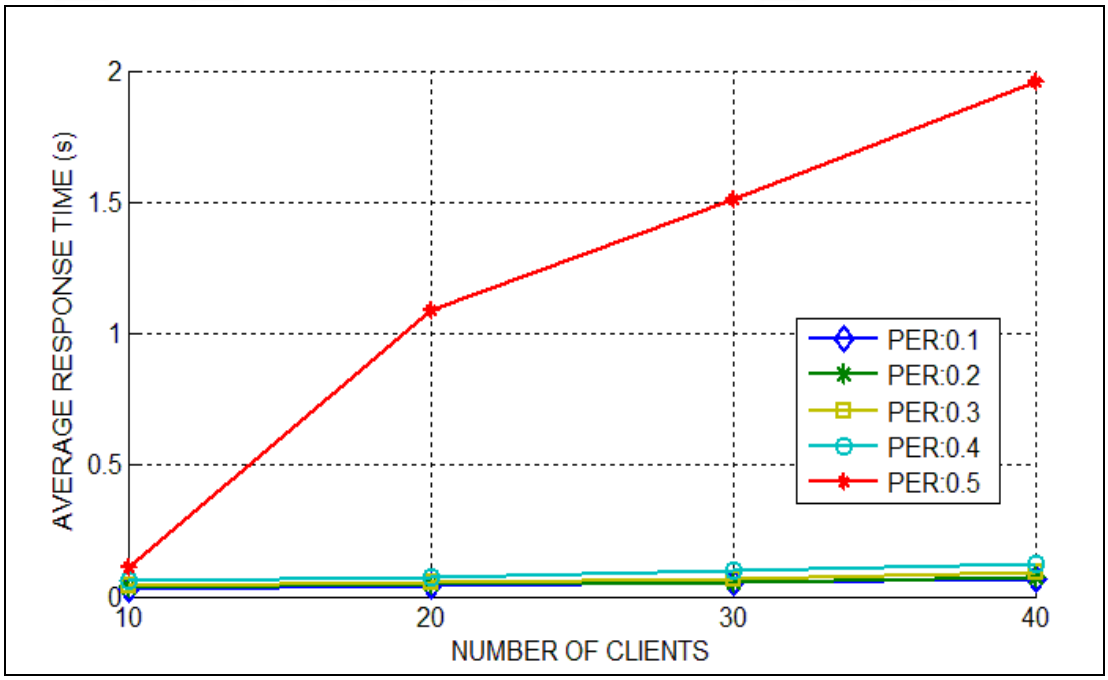


Figure 5.11. Average Response Time results of Service-2 Model Tests with different WiFi PERs

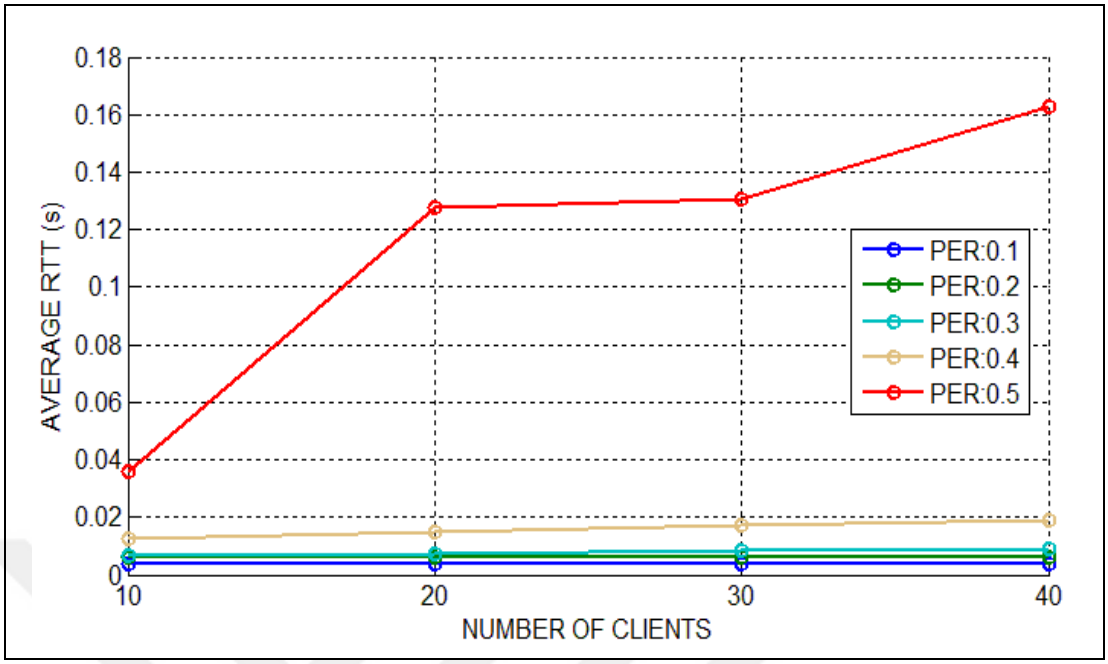


Figure 5.12. Average RTT results of Service-2 Model Tests with different WiFi PERs

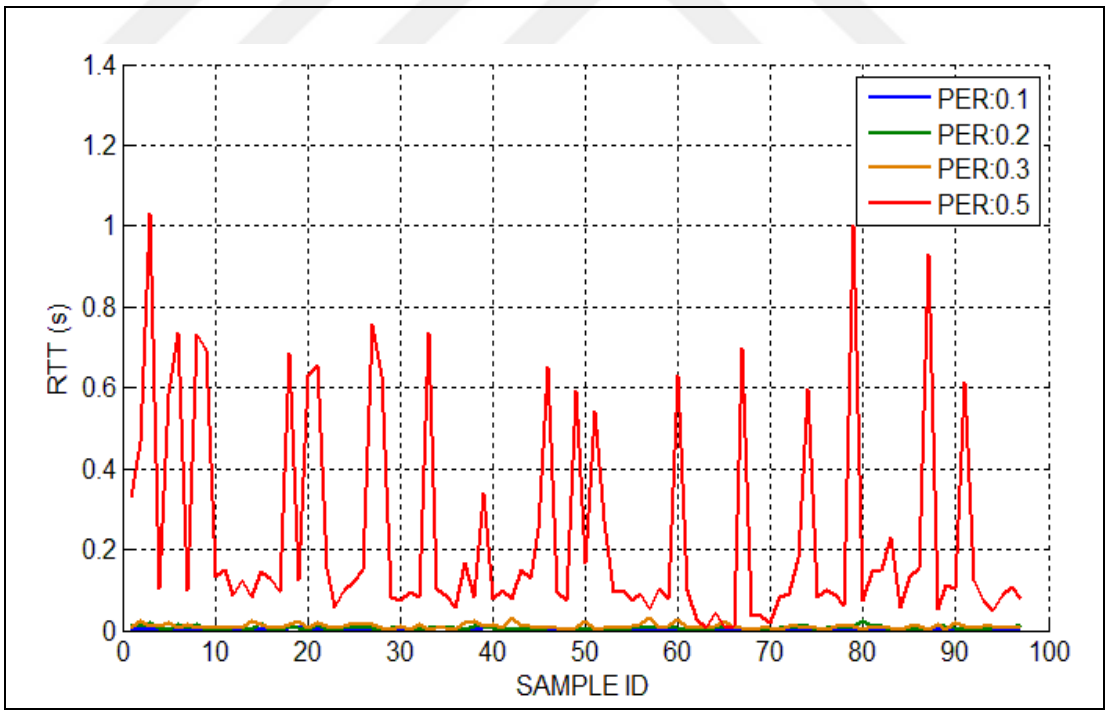


Figure 5.13. Consecutive RTT samples obtained from a client when N=40 for Service-2 Model

Figure 5.14 illustrates average response time results when WiFi PER is less than 0.2. When N increases up to 80, the average response time results obtained from the tests of the same N are close to each other. However, when N is 160, a dramatic increase in average response time is observed. In addition, when N is 160, the average response time increases with increasing WiFi PER. If we examine the results of RTT, as illustrated in Figures 5.15 and 5.16, we do not observe an significant change in RTT so this is not the reason for this dramatic change.

However, if we examine results of the average number of packets collected in the queue (i.e. *n*), we observe a dramatic increase as illustrated in Figure 5.17. In addition, when N is 160 and WiFi PER is 0.2, the birth-death period for the queue is extremely longer as illustrated in Figure 5.18. Since Formula 4.2 expresses that the average response time is dependent to *n*, dramatic increase in *n* means dramatic increase in the average response time. So the dramatic increase in average response time when N is 160 is due to this performance metric. As illustrated in Figure 5.18, although the birth-death period for the queue increases when N is up to 80, the whole enqueued request packets are served from server as time goes by. Hence, in overall, this analysis reveals that when WiFi PER is less than 0.2, this gateway model can handle 80 simultaneous clients for a stable operation.

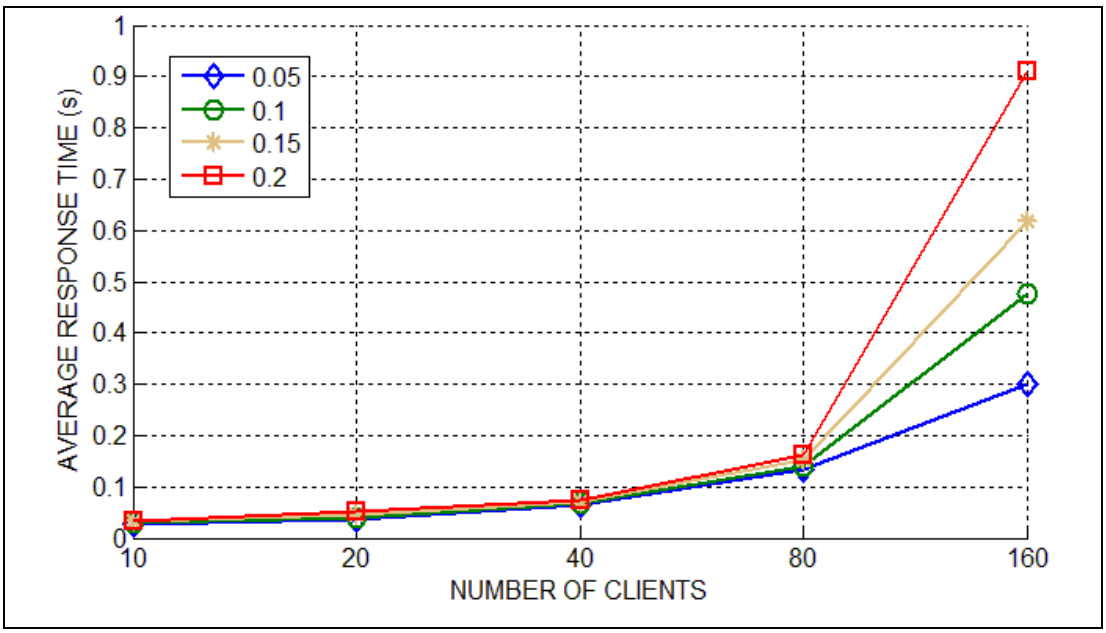


Figure 5.14. Average Response Time results of Service-2 Model when WiFi PER is up to 0.2

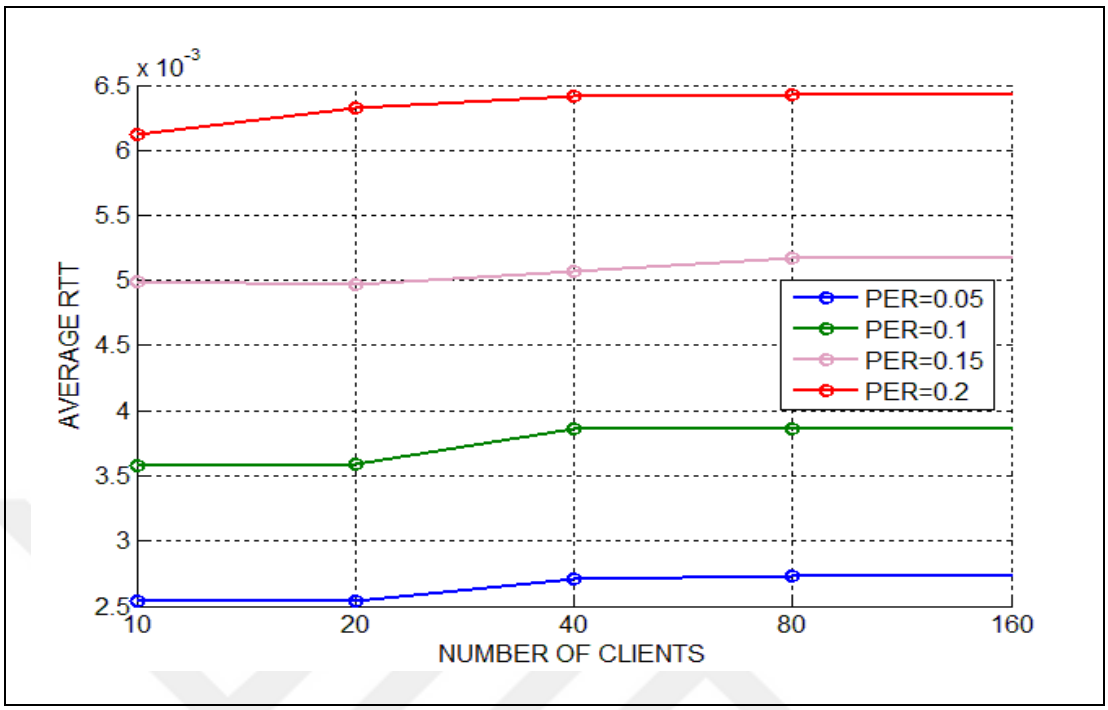


Figure 5.15. Average RTT results of Service-2 Model when WiFi PER is up to 0.2

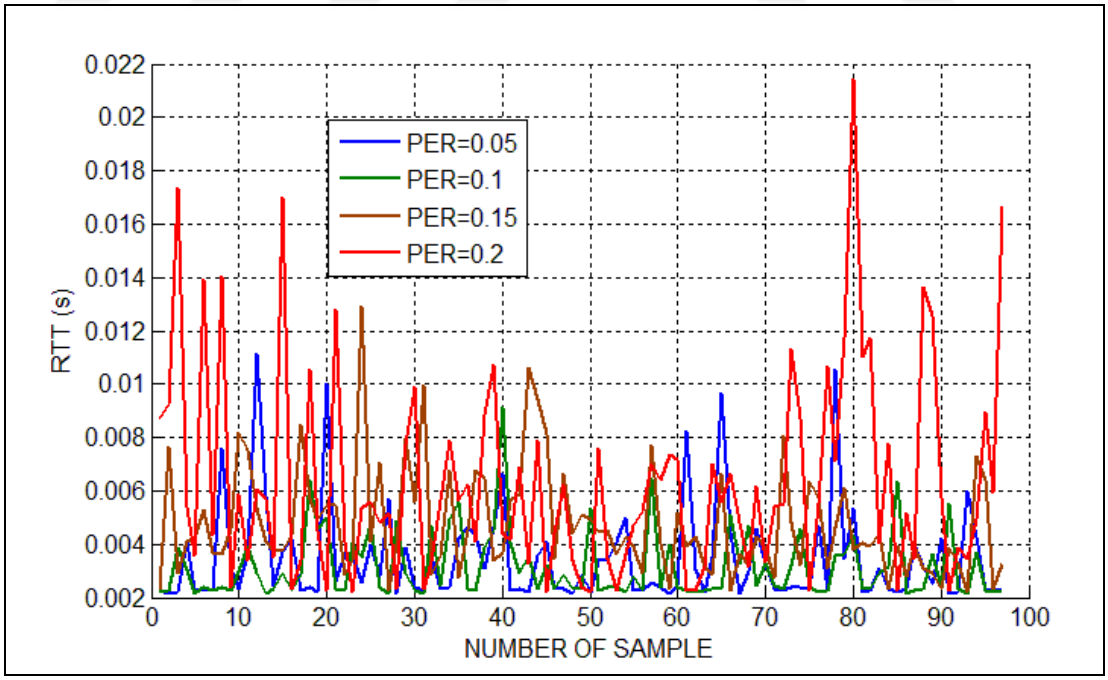


Figure 5.16. Consecutive RTT results obtained from a client with Service-2 Model when WiFi PER is up to 0.2

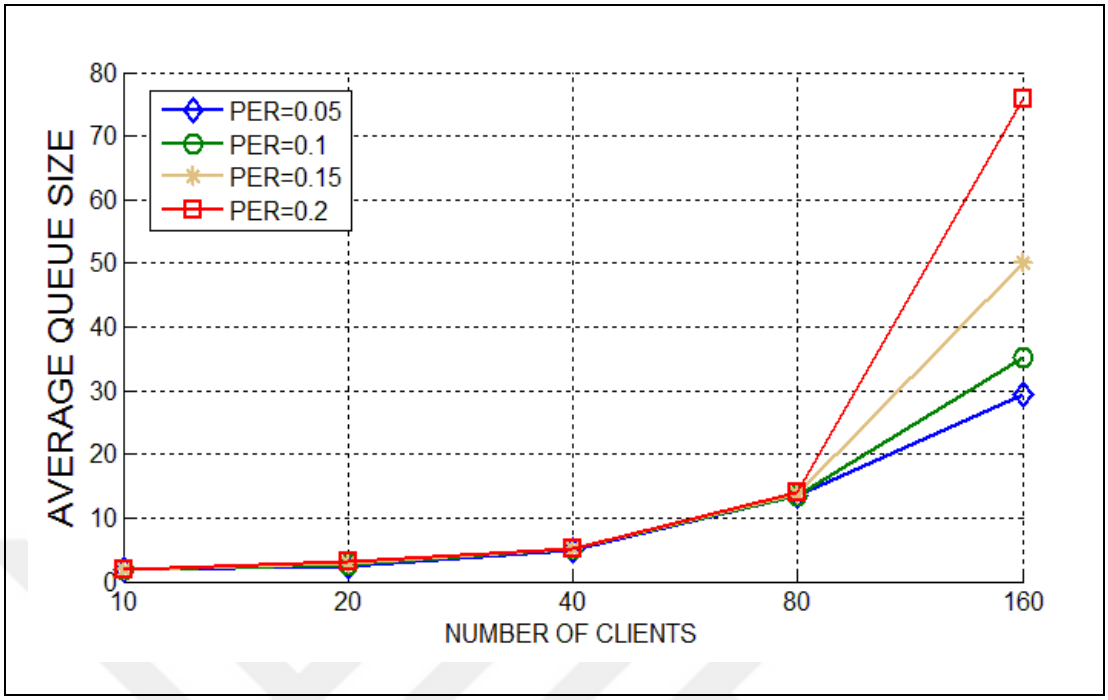


Figure 5.17. Average Queue Size (i.e, *n*) of Service-2 Model when WiFi PER is up to 0.2

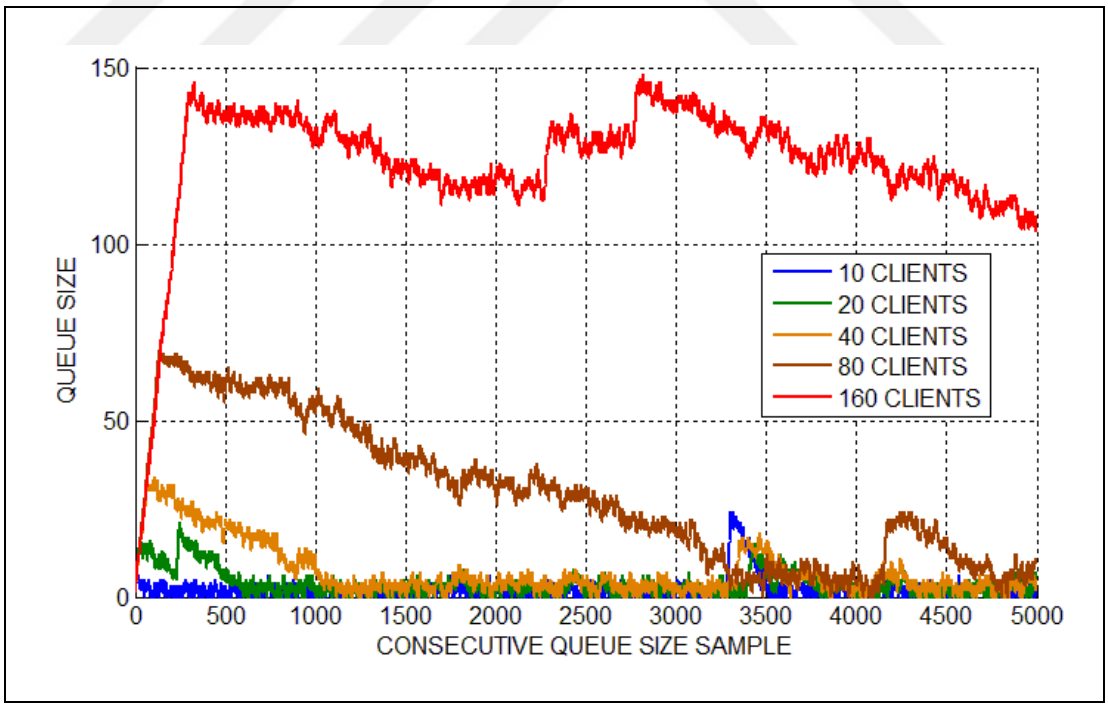


Figure 5.18. Packet number changing in the queue of Service-2 Model when WiFi PER is 0.2

5.2.2.2. Tests with Bursty Traffic Conditions

In first group of tests with bursty traffic conditions, we evaluate WiSEGATE with different arrival rates when we are using an infinite queue in the gateway. Since the queue size is not limited, any one of the request packets is not dropped by the gateway.

The Figure 5.19 illustrates the number of the request packets in the queue with different arrival rates when N is up to 40. It is clear that when r=1, the gateway has a stable behavior and the packets do not pile up in the queue. So, the average response times are not extremely high as depicted in Figure 5.20.

When r>1, request packets start to pile up in the queue and this results with extremely high average response times as depicted in Figure 5.20. Hence, a bounded size queue could be used to limit the average response time.

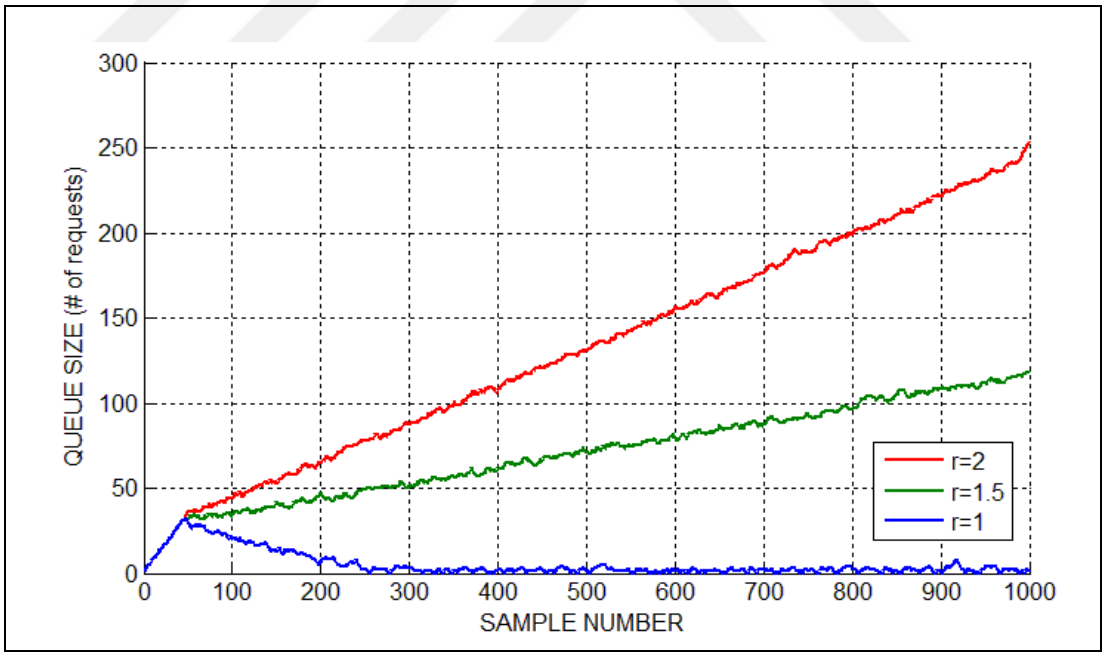


Figure 5.19. Packet number changing in the queue with different request traffic rate

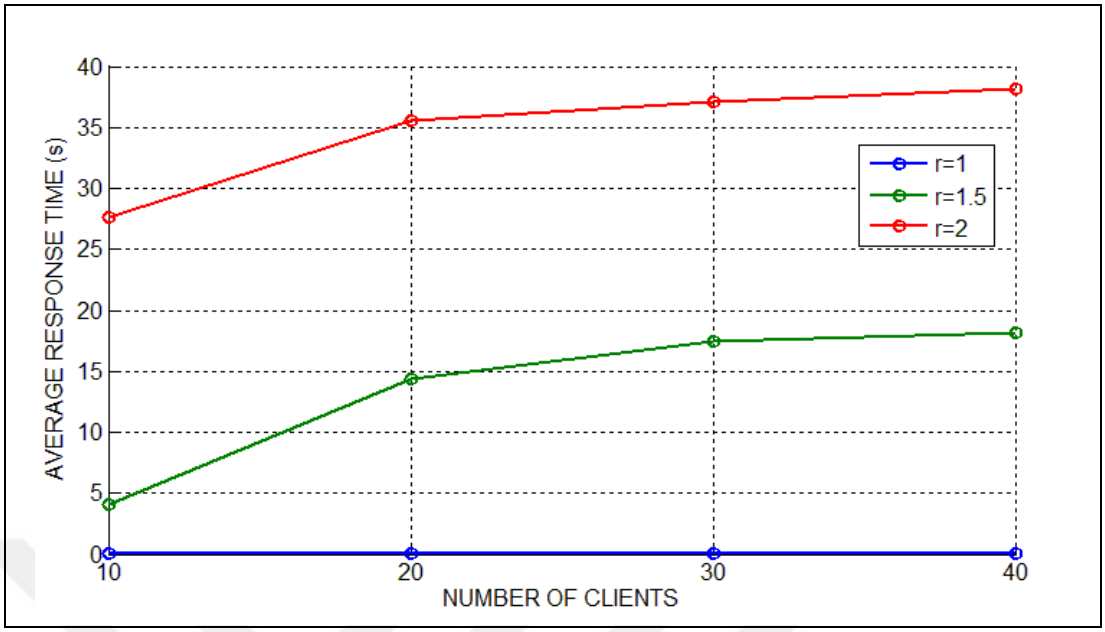


Figure 5.20. Average Response Time results with different request traffic rate

In second group of bursty traffic tests, we evaluate WiSEGATE with a bounded size queue. Figure 5.8 illustrates the results when the sensor data caching is not implemented in the gateway. Figure 5.21 gives the response time results and Figure 5.22 gives the drop rate of packets. As seen in Figure 5.21, response time increases as increasing size of the queue. Traffic rate is an important factor in increase of average response time. In addition, as traffic rate increases packet drop ratio increases. Thus, denser traffic results bounded size queue to reach its limits quickly and so more request packets are dropped by the gateway.

The Fig. 5.23 illustrates comparative results of two queue handling mechanisms; in the first, sensor data is not cached in the gateway and in the second, the sensor data is cached in the gateway. It is clear that the increase in bound of the queue results the response time to increase. When $r \leq 3$, using a caching mechanism slightly improves average response time. Note that using the caching mechanism on the gateway and dropping the packet have almost same effect when r=3. However, when r=4 caching worsens the average response time of the gateway by a factor 2. We can argue that, in bursty traffic conditions, WiFi cannot handle the multiclient request traffic because of high collision rates in the network and TCP resends the segments for false timeouts.

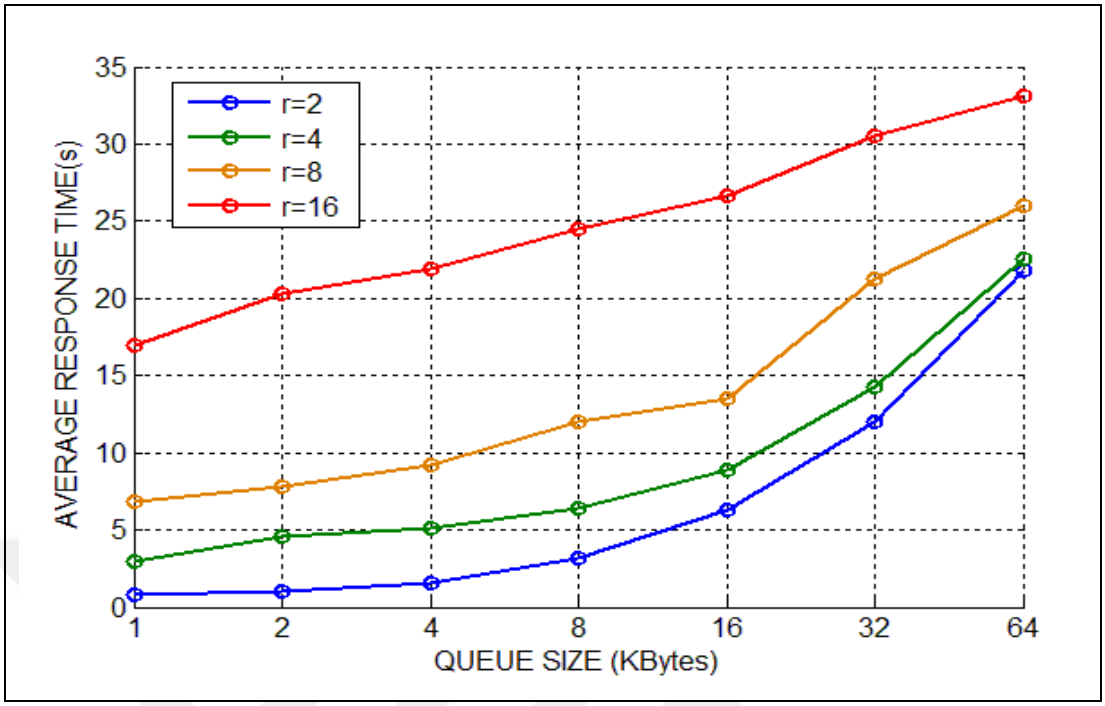


Figure 5.21. Average Response Time results in the queue with different request traffic rate when N=10

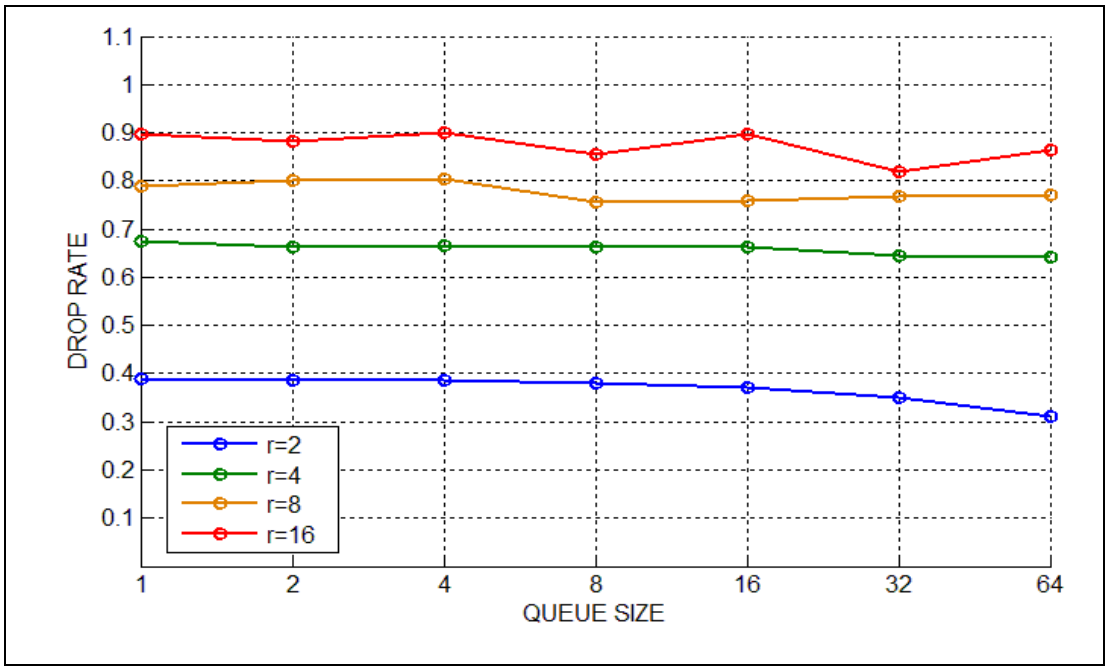


Figure 5.22. Drop rate in the queue with different request traffic rate when N=10

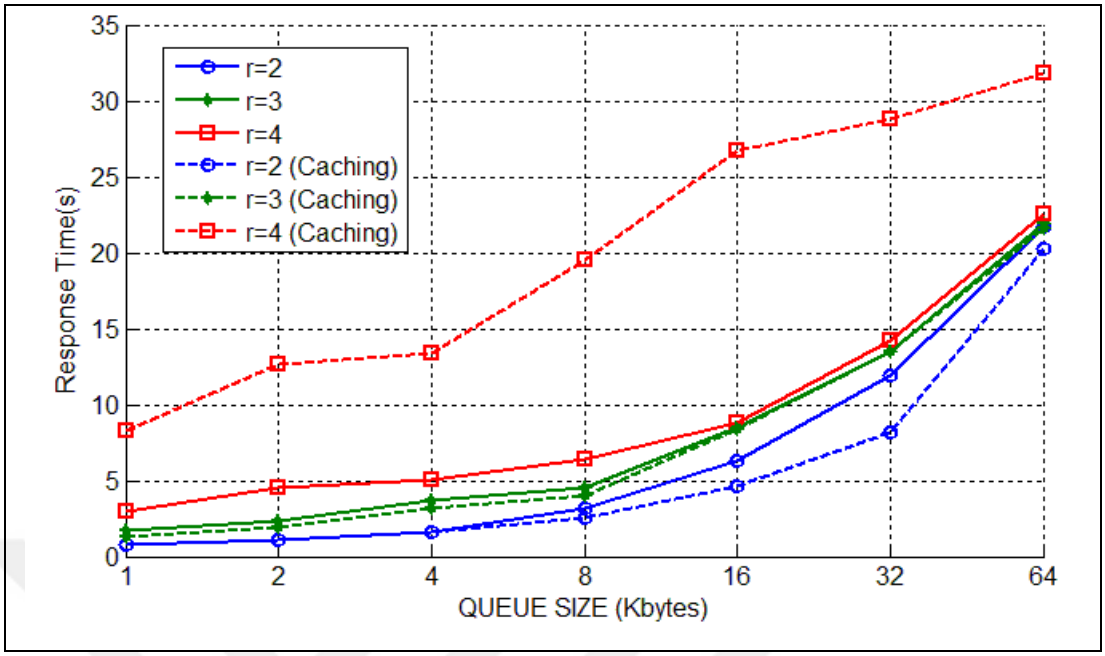


Figure 5.23. Comparison of caching mechanism and non-caching mechanism with Response Time results

5.3. REAL TESTBED

For the experiments with a real test environment, the interconnection service layer logic is implemented in TelosB WSN nodes. These nodes use CC2420 radio transceiver [58] which is compatible with IEEE 802.15.4 radio channels for wireless communication. In addition, these nodes are compatible with TinyOS 2.0 [34] and NesC [59] language. A real testbed with a 4 hops chain topology is setup with these sensor nodes. This WSN is considered as a local IPv6 network and the local network addressing scheme of IPv6 is used to determine the sensor nodes. So the sensor nodes have unique IPv6 addresses with local IPv6 prefix FE80::/64 for the network identifier and 16-bit short address for the interface identifier. The 6LoWPAN datagrams are encapsulated in the default Active Messages of TinyOS [34].

The gateway logic of WiSEGATE is implemented in an Intel Core 2 Duo Machine with a 3.00 GHz processor. This machine can communicate with the interconnected networks using the communication interfaces on itself. For the experiments, a test program is implemented on a remote machine which is on the same LAN with the gateway. Between

the remote machine and the gateway, Ethernet is used as the MAC protocol. The overall experiment scheme is illustrated in Figure 5.10.

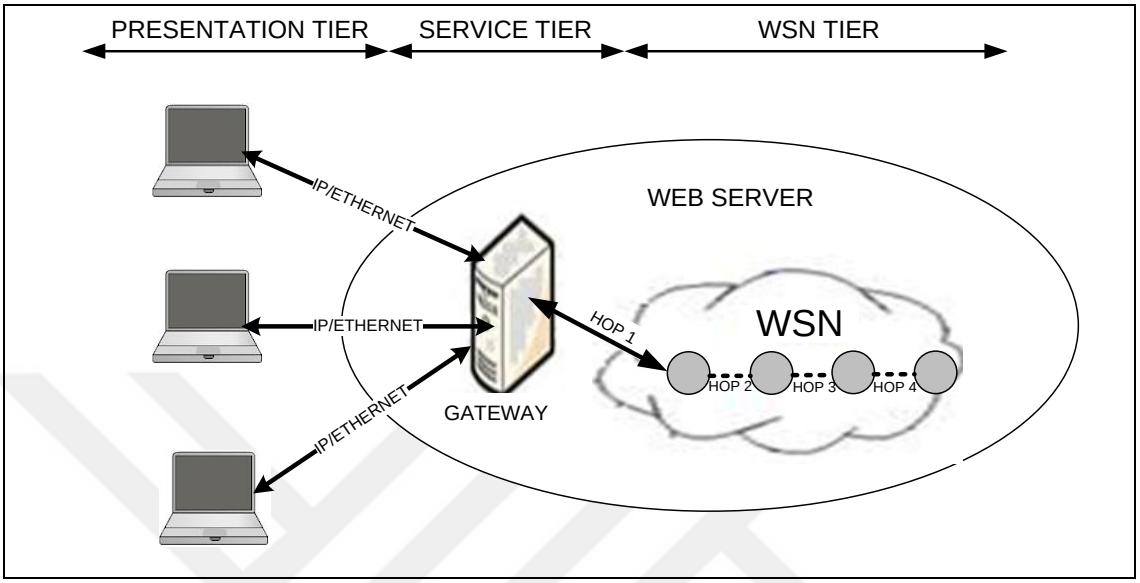


Figure 5.24. Real testbed interconnection scheme

5.3.1. Response Time Tests

The gateway logic of WiSEGATE is implemented in a Intel Core 2 Duo Machine with a 3 GHz processor. This machine can communicate with the interconnected networks using the communication interfaces on itself. For the experiments, a test program is implemented on a remote machine which is on the same LAN of the gateway. Between the remote machine and the gateway, Ethernet is used as the MAC protocol. The overall experiment scheme is illustrated in Figure 5.24.

Figure 5.25 illustrates the average, maximum and minimum response times obtained from each tests. The results show that the response time increases linearly as the number of simultaneous requests increase. In addition, end-to-end response time scales linearly with the increasing number of radio hops at WSN.

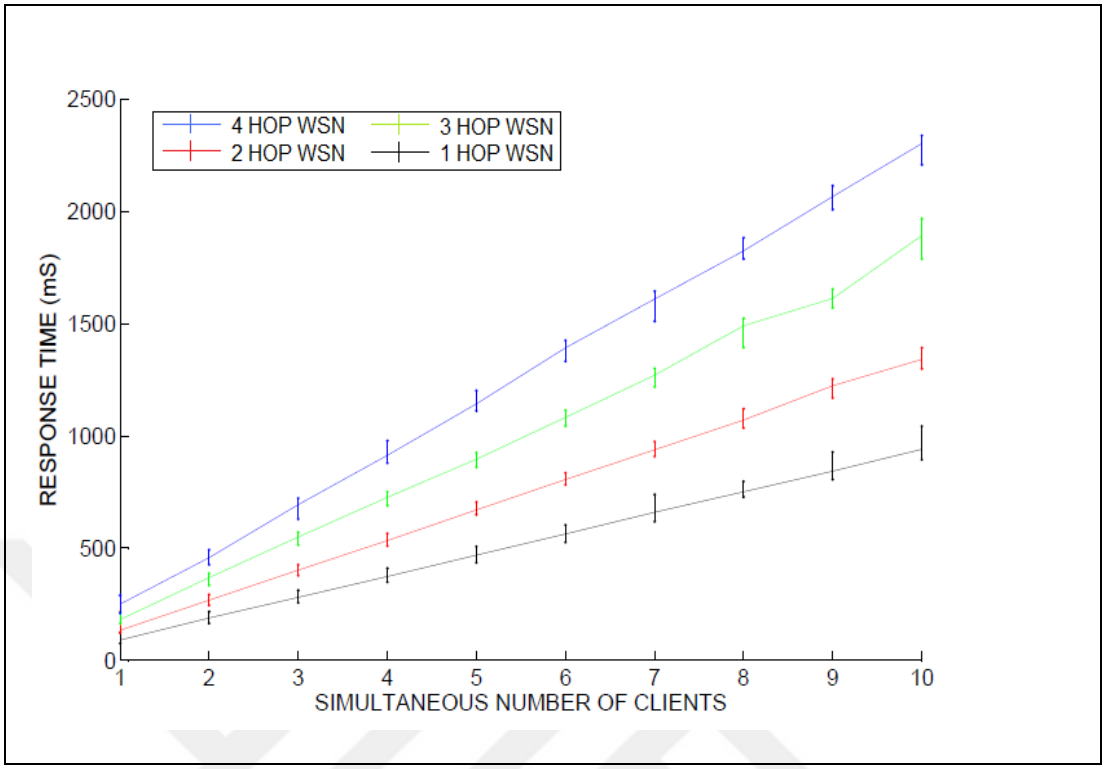


Figure 5.25. Response Time results obtained from real testbed scenario

5.3.2. Round Trip Time (RTT) Tests

In these tests, RTT is used to observe how both transmission media of WSN and web client-to-gateway behave. Figure 5.26. illustrates the time sequence of transmission of each TCP segment that is sent from the test program and the ACKs that are received from the gateway and WSN.

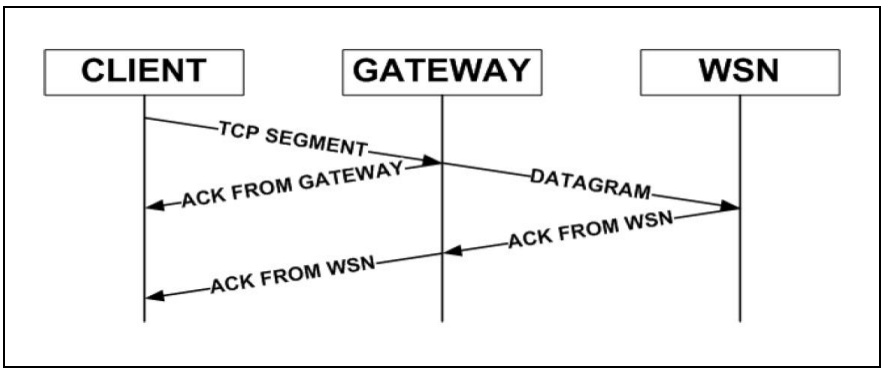


Figure 5.26. Timing sequence for RTT tests in real testbed

On the remote client, a test program was implemented to divide a data stream into chunks that can fit into a standard TCP segment and send them to the server consecutively. During the tests, 50ms is introduced between consecutive sending operations.

For the analysis, we have conducted tests in which clients send 50 consecutive TCP segments to the gateway and WSN which has up to four hops. Figure 5.27. illustrates the RTT results of the consecutive samples for increasing number of hops. RTT variations are due to the uncertain transmission behavior of the propagation media of the radio channels in WSN. As seen in this figure, RTT times between gateway and the client is constant because of the wired transmission medium between these hosts.

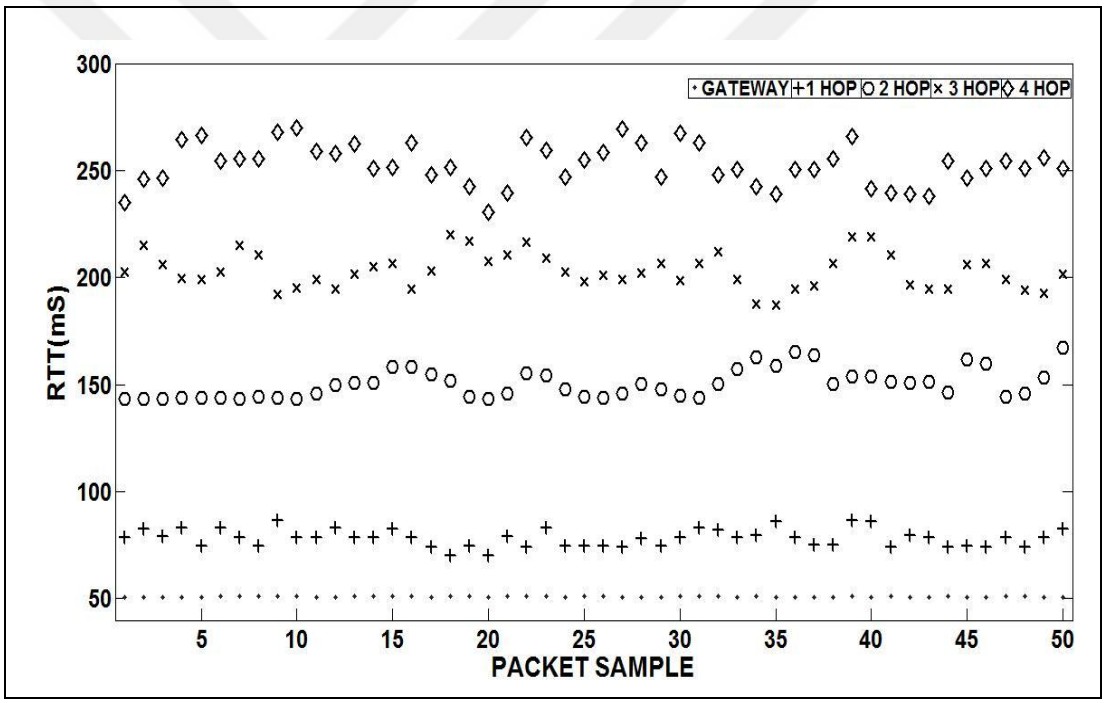


Figure 5.27. RTT results of consecutive segment transmission for increasing number of hops

5.4. COMPARISON OF WISEGATE

The memory footprint of WiSEGATE is compared with memory usage of gateway-based models given in the Related Work Chapter. As mentioned earlier, these gateway models integrate IP to the resource constrained nodes with some conventional TCP/IP stack. The

sensor node maintains end-to-end reliable or unreliable communications with the help of the protocols of these stacks. On the other hand, WiSEGATE uses only a lightweight service layer which only handles 6LoWPAN datagrams and forwards the requests towards the destination node. In the proposed interconnection scheme, the gateway node maintains the reliability of the end-to-end connections between multiple clients and the sensor service on a WSN node.

Table 5.5 gives the memory footprint comparison of WiSEGATE with uIPv6 and lwIP [28]. As seen from the table, if we use WiSEGATE in a TinyOS compatible node, at least, 2 Kbytes of more memory is remained for other operations on this sensor node. Since WiSEGATE do not require memory usage for handling end-to-end reliable communication on a sensor node, it eliminates the uncertainity of the memory usage which a gateway model has for handling end-to-end reliable connections for multiple simultaneous clients. So, it guarantees its memory usage in a sensor node even multiple simultaneous clients access sensor data on this sensor node.

Table 5.5. Memory footprint comparison results

Function	**Code Size (bytes)**
Lightweight Middleware	2848 Bytes (ROM)
	406 Bytes (RAM)
uIPv6	5188 Bytes (Totally)
lwIP	14588 Bytes (Totally)

6. CONCLUSION AND FUTURE WORK

In this thesis, an interconnection model, named as WiSEGATE, addressing the seamless interconnection of multiple simultaneous clients in IP network to WSN is proposed. The current interconnection approaches for the proposed model is examined and a prototype of a new web server which supports three tier service scheme to access the physical data in a particular location by remote entities is developed. Our approach does not require any TCP/IP stack in a resource constrained node for end-to-end interconnection. For determining limits of the proposed model, firstly, we examined the steps for request/response mechanism and formulized the queuing system. By doing this, we derived a definition of the request traffic. For a bursty traffic, a caching mechanism was defined to obtain more reasonable response times. Secondly, we have performed tests in simulation and real testbed environment for proof of the concept. WiSEGATE can achieve reasonable response time up to 80 simultaneous connections when WiFi PER is less than 0.2. In addition, the comparison of the memory footprint for WiSEGATE have a good indication that the relieving the sensor node from a TCP/IP stack is important to gain more free code space.

As a future work, we will extend the performance evaluation of the model with several scenarios. Firstly, we will evaluate the proposed model with a multi-hop WSN with considering dynamics of the network. Secondly, with a heterogenous network of static or mobile clients, we will evaluate the proposed service model. Then, we will examine dynamic caching models and derive a dynamic buffer and caching scheme for the evaluation of WiSEGATE in a bursty traffic. In addition, we will extend our work to observe interconnection model with different service traffics over WSN. For these observations, we will extend the gateway application to support different traffic types such as streaming a real time traffic between an RTP client and a sensor node. This study will aim to integrate our interconnection model with WMSN applications.

In our model, we evaluated the gateway using a traffic scheme based on a request push to the WSN. It can be argued that this is a simplistic method for periodic data acquisition. However, our observations aim to evaluate the proposed approach with different traffic

rates of simultaneous client request on the front-end of the gateway and observe it with bursty traffic conditions. For the future work, we will integrate a forward server scheme to the proposed approach. This scheme will enable a sensor node to send periodic sensor data to remote entities without need of request push. With this scheme, when a client expects periodic sensor data acquisition from a sensor node, it will only send a structured message which represents this periodic sensor data acquisition expectation.

In tests which we performed to observe the scalability of the proposed model, we used default TCP implementation defined for the wired networks. As a future work, we will examine several TCP implementations on sensor nodes and their performance and propose a new TCP implementation for end-to-end reliable communication for the interconnection. Also, we will extend our gateway-to-sensor node reliable communication mechanism considering the dynamics of the physical radio channels in WSN and MAC in the sensor node.

REFERENCES

1. Atzori, L., A. Iera and G. Morabito, "The Internet of things: A survey", *Computer Networks*, Vol. 54, pp. 2787-2805, 2010.

2. Roman, R., J. Lopez and C. Alcaraz, "Do Wireless Sensor Networks Need to be Completely Integrated into the Internet", https://www.nics.uma.es/system/files/papers/Roman2009.pdf, [retrieved 12 May 2012].

3. Castellani, A.P., M.I. Ashraf, Z. Shelby, M. Luimula, J. Yli-Hemminki and N. Bui, "BinaryWS: Enabling the Embedded Web", *Future Network and Mobile Summit, 2010*, Florence, 16 June-18 June 2010, pp. 1-8.

4. Akyildiz, I.F., W. Su, Y. Sankarasubramaniam and E. Cayirci, "Wireless sensor networks: A survey", *Computer Networks*, Vol. 38, pp. 393-422, 2002.

5. Hui, J.W., D.E. Culler, "Extending IP toLow-Power, Wireless Personal Area Networks", *IEEE Internet Computing*, Vol.12, Issue 4, pp. 37-45, 2008.

6. Serdaroglu, K. C., S. Baydere, "Seamless interconnection of WSN and Internet", *20th International Conference on Software, Telecommunications and Computer Networks (SOFTCOM) 2012*, Split, Croatia, 11 September – 13 September 2012, pp. 1-6.

7. Stankovic, A., "When Sensor and Actuator Networks Cover the World", *ETRI Journal*, Vol. 30, no. 5, pp. 627-633, 2008.

8. Rodrigues, J. J. P. C, P. A. C. S. Neves, "A survey on IP-based wireless sensor network solutions", *International Journal of Communication Systems*, Vol. 23, Issue 8, pp. 963-981, 2010.

9. Singh, D., U. S. Tiwary, H. Lee and W. Chung, "Global healthcare monitoring system using 6LoWPAN Networks", *Advanced Communication Technology, 2009. ICACT 2009, 11th International Conference,* Phoenix Park, 15 February – 18 February 2009, Vol.1, pp. 113-117.

10. Safavi A. A., A. Keshavarz-Haddad, S. Khoubani, S. Mosharraf-Dehkordi and A. Dehghani-Pilehvarani, "A remote elderly monitoring system with localizing based on Wireless Sensor Network", *Computer Design and Applications (ICCDA), 2010 International Conference,* 25 June – 27 June 2010, Vol. 2, pp. V2-553 – V2-557.

11. Tolstikov A., X. Hong, J. Biswas, C. Nugent, L. Chen and G. Parente, "Comparison of Fusion Methods Based on DST and DBN in Human Activity Recognition", *Journal of Control Theory and Applications,* Vol. 9, pp. 18-27, 2011.

12. Suryady, Z., M. H. M. Shaharil, K.A. Bakar, R. Khoshdelniat, G. R. Sinniah and U. Sarwar, "Performance evaluation of 6LoWPAN-based precision agriculture*", Information Networking (ICOIN), 2010 International Conference,* Barcelona, 26 January – 28 January 2011, pp. 171-176.

13. Gungor, V. C., G. P. Hancke, "Industrial Wireless Sensor Networks: Challenges, Design Principles and Technical Approaches", *Industrial Electronics, IEEE Transactions,* Vol. 56, Issue 10, pp. 4258-4265, 2009.

14. Sadlacek, T., J. Jokinen, C. Postelnicu and J. L. M. Lastra, "Safety monitoring system for a manufacturing using 6LoWPAN technologies", *Systems, Man and Cybernetics (SMC), 2012 IEEE International Conference,* Seoul, 14 October – 17 October 2012, pp. 1581 – 1585.

15. Dorge, B.M., T. Scheffler, "Using IPv6 and 6LoWPAN for home automation networks", *Consumer Electronics – Berlin (ICCE-Berlin), 2011 IEEE International Conference,* Berlin, 6 September – 8 September 2011, pp. 44-47.

16. Kovatch, M., M. Weiss and D. Guinard, "Embedding internet technology for home automation", *Emerging Technologies and Factory Automation (ETFA),2010 IEEE International Conference*, Bilbao, 13 September – 16 September 2010, pp. 1 – 8.

17. Saqib, M., C. Lee, "Traffic control system using wireless sensor network", *Advanced Communication Technology (ICACT) 2010, IEEE 12th International Conference*, Phoenix Park, 7 February 2010 – 10 February 2010, Vol. 1, pp. 352 – 357.

18. Sasidharan, S., F. Pianegiani, and D. Macii, "A protocol performance comparison in modular WSNs for data center server monitoring", *Industrial Embedded Systems (SIES), IEEE International Conference*, Trento, 7 July – 9 July 2010, pp. 213 – 216.

19. Park, M., S. Chung and C. Ahn, "TCP's dynamic adjustment of transmission rate to packet losses in wireless networks", *EURASIP Journal on Wireless Communication and Networking 2012*.

20. Ting, H., "A new interconnection scheme for WSN and Ipv6-based internet", *Information, Computing and Telecommunication, 2009, YC-ICT '09, IEEE Youth Conference*, Beijing, 20 September – 21 September 2009, pp. 34 – 37.

21. Gopinath, R. S., Z. Suryayd, U. Sarwar and M. Abbas, "A gateway solution for IPv6 wireless sensor networks", *Ultra Modern Communications & Workshops, 2009, ICUMT '09, IEEE International Conference*, St. Petersburg, Russia, 12 October – 14 October 2009, pp. 1 – 6.

22. Montenegro, G., N. Kushalnagar, J. Hui and D. Culler, "RFC 4644: Transmission of IPv6 Packets over IEEE 802.15.4 Networks", http://tools.ietf.org/html/rfc4944, [retrieved 27 April 2011].

23. Hui, J. W., D. E. Culler, "IPv6 in Low-Power Wireless Networks", *Proceedings of the IEEE*, Vol. 98, Issue 11, pp. 1865-1878, 2010.

24. Piccolo, F. L., D. Battaglino, L. Bracciale, A. Bragagnini, M. S. Turolla and N. B. Melazzi, "On the IP support in IEEE 802.15.4 LR-WPANS: Self-configuring solutions for real application scenarios", *Ad Hoc Networking Workshop (Med-Hoc-Net),2010, The 9th IFIP Annual Mediterranean Conference* , Juan Les Pins, France, 23 June – 25 June 2010, pp. 1 – 10.

25. Chen, Y., W. Shen, H. Huo and Y. Xu, "A smart Gateway Design for Health Care System Using Wireless Sensor Network", *Sensor Technologies and Applications (SENSORCOMM), 2010 IEEE Fourth International Conference*, Venice, 18 July – 25 July 2010, pp. 545-550.

26. Narmada, A., P. S. Rao, "WSN and IP based parking management system", *Sensing Technology (ICST), 2012 IEEE Sixth International Conference*, Kolkata, 18 December – 21 December 2012, pp. 434 – 438.

27. Harvan, M., "Connecting Wireless Sensor Networks to the Internet – 6lowpan implementation for Tiny OS 2.0", www.mharvan.net/papers/fgsn-2007.pdf, [retrieved 26 June 2011].

28. Dunkels, A., "Full TCP/IP for 8-bit architectures", *Proceedings of the 1st International Conference on Mobile Systems, applications and services, MobiSys '03,* San Francisco, CA, USA, 5 May 2003 – 8 May 2003, pp. 85 – 98.

29. Durvy, M.., J. Abeillé, P. Watterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne and A. Dunkels, "Making sensor network IPv6 Ready", *Proceedings of the 6th ACM Conference on Embedded network sensor systems, SenSys '08*, Raleigh, North Carolina, USA, pp. 421-422.

30. Yoon, I., S. Chung and J. Kim, "Implementation of Lightweight TCP/IP for Small, Wireless Embedded Systems", *Advanced Information Networking and Applications, 2009. AINA '09, IEEE International Conference*, Bradford, 26 May – 29 May 2009, pp. 965 – 970.

31. Dunkels, A., T. Voight, N. Bergman and M. Jönsson, “, The Design and Implementation of an IP-based Sensor Network for Intrusion Monitoring”, http://dunkels.com/adam/sncnw2004.pdf, [retrieved 4 July 2012].

32. Zhou, P., X. Lei and Z. Lv, “Study on Integrating BACNet with IPv6-based Wireless Sensor Networks”, *Procedia Engineering*, Vol. 29, pp. 275-279, 2012.

33. Han, G., M. Ma, “Connecting Sensor Networks with IP with a Configurable tiny TCP/IP protocol stack”, *Information, Communication & Signal Processing, 2007 6th IEEE Iinternational Conference*, Singapore, 10 December – 13 December 2007, pp. 1-5.

34. Levis, P., “TinyOS 2.0 Overview”, http://www.tinyos.net/tinyos2.x/doc/html/overview.html, [retrieved 24 June 2009].

35. Crossbow Technology, “TelosB Mote Platform”, http://www.willow.co.uk/TelosB_Datasheet.pdf, [retrieved 24 June 2009].

36. Crossbow Technology, “MicaZ: Wireless Measurement System”, http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf, [retrieved 28 October 2013].

37. Kransnyansky, M., “Universal TUN/TAP Device Driver”, http://www.kernel.org/doc/Documentation/networking/tuntap.txt, [retrieved 01 December 2012].

38. The Stanford Information Networks Group, “Blip 2.0”, http://tinyos.stanford.edu/tinyos-wiki/index.php/Blip_2.0, [retrieved 03 December 2012].

39. The Stanford Information Networks Group, “TinyOS Documentation Wiki”, http://tinyos.stanford.edu/tinyos-wiki/index.php/MainPage, [retrieved 13 December 2012].

40. Narten, T., E. Nordmark, W. Simpson and H. Soliman, “RFC 4861: Neighbour Discovery for IP version 6”, http://tools.ietf.org/html/rfc4861, [retrieved 27 April 2011].

41. Pirttikangas, S., K. Fujinami and T. Nakajima, “Contiki – a lightweight and flexible operating system for tiny networked sensors”, *Local Computer Networks, 2004. 29th Annual IEEE International Conference*, Tampa, FL, USA, 16 November – 18 November 2004, pp. 455 – 462.

42. Dunkels, A., J. Alonso, T. Voigt and H. Ritter “Distributed TCP Caching for Wireless Sensor Networks”, *3rd Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2004)*, Bodrum, Turkey, 27 June – 30 June 2004.

43. Hui, J., P. Thubert, “RFC 6282: Compression format for IPv6 Datagrams over IEEE 802.15.4-Based Networks”, http://tools.ietf.org/html/rfc6282, [retrieved 27 April 2011].

44. Hinden, R., S. Deering, “RFC 4291: IP Version 6 Addressing Architecture”, http://tools.ietf.org/html/rfc4291, [retrieved 27 April 2011].

45. Rensfelt, O., F. Hermans, C. Ferms, L. Larzon and P. Gunningberg “Sensei – a flexible testbed for heterogenous wireless sensor networks”,*Testbed and Research Infrastructures for the Development of Networks & Communities and Workshops, 2009. TridentCom 2009, 5th International Conference*, Washington, DC, 6 April – 8 April 2009, pp. 1-2.

46. Fielding, R., J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, “RFC 2616: Hypertext Transfer Protocol – HTTP/1.1”, http://tools.ietf.org/html/rfc2616, [retrieved 27 April 2011].

47. Bormann, C., A. P. Castellani and Z. Shelby, "CoAP: An application protocol for Billions of Tiny Internet Nodes", *Internet Computing, IEEE Proceedings,* Vol. 16, Issue 2, pp. 62-67, 2012.

48. Colliti, W., K. Steenhaut, N. De Caro B. Buta and V. Dobrota, "REST Enabled Wireless Sensor Networks for Seamless Integration with Web Applications", *Mobile Adhoc and Sensor Systems (MASS), IEEE 2011 8th Conference,* Valencia, 17 October – 22 October 2011, pp. 867-872.

49. Chander, R. P. V., S. Elias, S. Shivashankar, and P. Manoj, "A REST Based Design for Web of Things in smart environments", *Parallel Distributed and Grid Computing, 2012, IEEE 2nd International Conference,* Solan, 6 December-8 December 2012, pp. 337-342.

50. Trifa, V., Guinard D., "Web of Things", http://www.webofthings.org/index.php [retrieved 01 October 2013].

51. Laum, N., C. Lerche and D. Timmermann, "A Web service-based communication architecture for smartphone/WPAN sensor ensambles", *Emerging Technologies & Factor Automation(ETFA), 2012 IEEE 17th International Conference*, Krakow, 17 September – 21 September 2012, pp. 1-7.

52. Rajesekaran, P., R. P. Janardhan and R. P. V. Chander, "A smarter toll gate based on Web of Things", *Electronics, Computing and Communication Technologies (CONECCT), 2013 International Conference,* Bangalore, 17 January – 19 January 2013, pp. 1-6.

53. Chatzigiannakis, I., H. Hasemann, M. Karnstedt, O. Kleine, A. Kröller, M. Leggieri, D. Pfisterer, K. Römer, C. Truong, "Demo: True Self-Configuration for the IoT", *Internet of Things 2012, 3rd International Conference for Industry and Academia (IoT 2012),* Wuxi, China, 24 October – 26 October 2012.

54. Park, Y., N. Dinh, Y. Kim, "A networking monitoring system in 6LoWPAN networks", *Communications and Electronics (ICCE), IEEE 4th International Conference,* Hue, 1 August – 3 August 2012, pp. 69 – 73.

55. Sanchez, J. D. G., "Introduction to simulation with OMNET++", http://web.univ-pau.fr/~cpham/ENSEIGNEMENT/PAU-UPPA/PROTOCOLES/omnetp.pdf, [retrieved 01 October 2012].

56. Comer, D. E., *Internetworking with TCP/IP Vol. 1: Principles, Protocols and Architecture 4th Edition,* Prentice Hall, New Jersey, 2000.

57. Levis P. A., N. Lee, M. Welsh, D. E. Culler, "TOSSIM: accurate and scalable simulation for entire TivyOS applications", *Embedded Networked Sensor Systems, SenSys, 1st International Conference 2003*, Los Angeles, CA, USA, 5 November – 7 November 2003, pp. 126 – 137.

58. Texas Instruments, "CC2420: 2.4 GHz IEEE 802.15.4/ZigBee-ready RF Transceiver", http://www.ti.com/lit/ds/symlink/cc2420.pdf, [retrieved 27 June 2009].

59. Levis, P., D. Gay, *TinyOS Programming,* Cambridge University Press, Cambridge, 2009.