Stony Brook University

Design and Optimization of Node Architecture for Application Specific Multi-hop Wireless Networks

A Dissertation Presented

by

Sangkil Jung

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

December 2007

Stony Brook University

The Graduate School

Sangkil Jung

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree,
hereby recommend acceptance of this dissertation.

Sangjin Hong, Dissertation Advisor
Professor, Department of Electrical & Computer Engineering

Thomas G. Robertazzi, Chairperson of Defense
Professor, Department of Electrical & Computer Engineering

Hang-Sheng Tuan,
Professor, Department of Electrical & Computer Engineering

Hongshik Ahn
Professor, Department of Applied Mathematics

This dissertation is accepted by the Graduate School

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Design and Optimization of Node Architecture for Application Specific Multi-hop Wireless Networks

by

Sangkil Jung

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

2007

In this dissertation, we focus on the design and optimization of the node architecture for target specific applications under the 802.11-based multi-hop wireless network that has been recently spotlighted from the public domain. We consider two target applications: compression and packet aggregation algorithms on the Wireless Mesh Network (WMN), and a multi-modal tracking system underlying the multi-hop wireless network.

To conduct the design and optimization tasks of the compression and packet aggregation, we propose a profile-based network and hardware co-simulation method to characterize the global WMN performance as well as the real-timing nodal behaviors.

The WMN is equipped by a dedicated hardware platform possibly configured as a network processor. For the compression, RObust Header Compression (ROHC) is adopted. The co-simulation method integrates the network level simulator, NS-2 and hardware level simulator, SystemC. In this approach, we first insert the modules of ROHC and packet aggregation algorithms into the network simulator hierarchy, and measure the packet arrival times. Then, the corresponding hardware architecture is designed by SystemC for profiling the hardware delay appeared in encoding and decoding packets. Finally, the traced hardware delays are applied into the network level simulator to extract real-timing WMN behaviors changed by the hardware operation in each mesh router. Additionally, to accurately predict the capacity of a hardware design, we propose a numerical analysis method by using the open Jackson queueing network. The modeled queue systems are one-to-one mapped into the constructed hardware components to characterize the concurrent operations and interactional relationship between encoding and decoding paths in ROHC.

For the tracking system, we develop a multi-modal sensor-based tracking model where acoustic sensors mainly track the target objects and visual sensors compensate the tracking errors. We initially discover a *network synchronization problem* caused by the different location and traffic characteristics of multi-modal sensors, and non-synchronized arrival of the captured sensor data at a processing Server. We show the improved tracking accuracy from visual compensation in ideal case is severely degraded when the synchronization problem is involved in real situations. For the possible solution for the problem, we differentiate the service level of sensor traffic

based on Weight Round Robin (WRR) scheduling at the Routers. The weighting factor allocated to each queue is calculated by a proposed Delay-based Weight Allocation (DWA) algorithm. In addition, to numerically predict the number of success of the visual compensation, we propose a Statistical Estimation Algorithm (SEA) which is based on traffic measurement, random generation of transmission delay for sensor data, and statistical estimation. Based on the SEA, we propose an on-line version of the SEA called Statistical Estimation and Adaptation Algorithm (SEA^2), in which the acoustic sensor's object sampling interval is automatically adapted to achieve a certain level of tracking accuracy.

To my LORD who has thus far helped me, church, and family

Table of Contents

List of Figures

List of Tables

Acknowledgements

I am now looking back on the first year in Stony Brook. I was so nervous and strived to adapt myself to the change in the new surroundings. From that time, the laborious studies and lonely life have not seemed to overcome. Though, my sincere God has looked on me with His love and embraced, and I was able to recline on His bosom. His endless love makes me enter into rest and rise again with the new power. Thus, I first shall give thanks to Him and praise His name. From the time when I was leaving the home, my parents have prayed for me, given their love, and encouraged me to keep going my way. My brothers and sister have kept praying for my study and endurance in hard time as well. I thank God for allowing this precious family, and would like to say I love them so much. I can not forget the church members who have comforted and prayed for me for the last four and a half years here. Especially, I would like to appreciate Pastor Taejun Suk for his unceasing prayer, and my lovely brother Yougjun Lee without whom I can not finish the long race in this strange land. I also have thanksgiving to my fellow workers, Dongsan Lee, Kyunghoon Kim, and Seunghwan Choi. I also give thankful words to Yousung church members who are like my family. They have prayed for me and helped me to build healthy spirit and soul.

I am deeply grateful to my advisor, Prof. Sangjin Hong, for his concern and advice. He has given the right direction to tackle the problems in my research works, and helped complete this dissertation. When I first started my research in Mobile Systems Design Laboratory, I worked with Kyoung-Su Park, Junhee Mun, Jinseok

Lee, Yuntai Kyong, and Shung Han Cho. I additionally give thankful words to them. I also express my gratitude toward Prof. Thomas G. Robertazzi, Prof. Hang-Sheng Tuan, and Prof. Hongshik Ahn for their endeavors to listen and comment on this dissertation as the committee members.

For the final acknowledging words, I would like to include the Bible Word, Psalm 23 that has been staying in heart and giving the strength to my soul and spirit.

"The LORD is my shepherd, I shall not want.

He makes me lie down in green pastures; He leads me beside quiet waters.

He restores my soul; He guides me in the paths of righteousness For His name's sake.

Even though I walk through the valley of the shadow of death, I fear no evil, for You are with me; Your rod and Your staff, they comfort me.

You prepare a table before me in the presence of my enemies; You have anointed my head with oil; My cup overflows.

Surely goodness and lovingkindness will follow me all the days of my life, And I will dwell in the house of the LORD forever."

Chapter 1

Introduction

Ever since the Internet has emerged, the conventional network has become complicated and specialized to satisfy various user requirements. For example, target specific applications such as compression, aggregation, audio/video, security and object tracking have gradually hosted the wireless networks as well as the traditional wire-line networks.

Since each mobile host could send data at 300Mb/s rate in the next generation wireless LAN environment such as 802.11n, the handling of an application-specific function in a network processor could have processing limitation and make a significant impact on the overall network performance. This means that we need to know how the capacity and working behavior of a single node router affects the overall network performance in addition to the impact of the global network behavior. In other words, an accurate analysis of a network should be obtained from not only the overall network characteristics but also the capacity of a single node router. Hardware

implementation and evaluation for the RObust Header Compression (ROHC) and packet aggregation is one example to understand the nodal behavior in a multi-hop wireless environment such as the Wireless Mesh Network (WMN) [1]. The ROHC is a compression algorithm to alleviate the packet header overhead compared with small-size payload in wireless environment featured by high error rate and long Round Trip Time (RTT). The packet aggregation also fosters the radio resource savings by collecting the compressed packets in a specific amount, which leads to the reduction of data transmissions and MAC header overhead.

In order to investigate the real-timing characteristics of a WMN adopting the hardware realization of the ROHC and packet aggregation, we propose a profile-based network/hardware co-simulation method in this chapter. To justify that the hardware realization is essential for WMN backhaul routers, we preliminarily conduct extensive simulation studies by using NS-2 [2]. From the simulation under the user application like 20Bytes UDP sensor data and voice encoded by G.729a codec, we observe that the packet aggregation without ROHC improves WMN performance, but it produces large variation and unstable results. This disadvantage can be eliminated by associating the packet aggregation with ROHC such that 40 packet aggregation with ROHC provides high and reliable sensor throughput and successful voice calls. However, if ROHC and packet aggregation are processed by a general purpose processor (GPP), which is a software-driven implementation, even the association of the two algorithms cannot provide the significant WMN performance enhancement due to the processing power limitation of the GPP. We show the processing overhead problem gets alleviated by

the proposed hardware architecture.

The co-simulation method integrates a network level simulator (NS-2) and a hardware level simulator (SystemC), and is performed by 3 step profiling procedures. In the first step, we insert the modules of ROHC and packet aggregation algorithms into the network simulator hierarchy, and measure the packet arrival times. Then, in step 2, the measured times are injected into the SystemC module to record the hardware processing time. In the final step, we recursively apply the hardware processing time into the NS-2 simulation to profile the network level simulation results that include the nodal behaviors. Since the simulation-based or real implementation of the hardware architecture is a time-consuming task, we can use a numerical analysis model to predict the capacity of the hardware architecture. For this work, we use the open Jackson queueing network consisting of $G/M/1$, $M^{[K]}/M/1$, $M/M/1$, $M/M/\infty$ queue systems, where G means a general arrival, M is a poisson arrival or an exponential service, and $M^{[K]}$ is an arrival with bulk size K. In the model, each queue is one-to-one mapped into the corresponding hardware element to characterize the concurrent and parallel operation of the hardware system.

In object tracking applications, acoustic sensors have been widely used due to the advantages such as the low cost of deployment and flexibility. However, they are not only sensitive to reverberant indoor environment which frequently generates extraordinary signals, but also have difficulty in satisfying the requirement of consistent data. Thus, other type of sensor to assist the acoustic operation is necessary to obtain more reliable data. Among a variety of sensors, a visual sensor can be a good candidate

to collect consistent and reliable data. In a multi-modal tracking environment, the audio-video joint processing provides more accurate object movement estimation by mutually complementing the errors occurring in the middle of tracking. As a kind of the multi-modal system, we developed a tracking application model in which acoustic sensors mainly track the objects and visual sensors compensate the tracking errors inherent in the acoustic estimate [3] [4]. It has advantage such as the on-the-fly error correction with low computing complexity. When an acoustic sensor with adjacent microphones is sampling object signal, the Particle Filter (PF) algorithm [5] associated with the acoustic sensor obtains the object coordinates. A visual sensor supports the tracking task by correcting the PF estimation error based on the localization algorithm with parallel projection model. Since the processing overhead of PF is a few microseconds [6] and the localization algorithm for visual compensation is rarely performed compared with the PF calculation, the tracking model can minimize the overall processing overhead. Based on the developed tracking application model, we initially discover a *network synchronization problem* caused by the unbalanced delivery time among multiple sensor types. For example, visually captured image size is larger than the acoustic sensor data size, so that visual images arrive at a processing Server later than acoustic sensor data. In this situation, the Server performs the visual compensation only when the sampling times of the sampled data satisfy the successful condition for visual compensation. We show the improved tracking accuracy from visual compensation in ideal case is severely degraded when the *network synchronization problem* is involved in real situations. For the possible solution for

the non-synchronization problem in a network, we separate the network queues to differently serve the sensor traffic and non-sensor traffic. The traffic differentiation model is achieved by Weight Round Robin (WRR) where the weights are allocated based on the proposed Delay-based Weight Allocation (DWA) algorithm. We also investigate the behavior of the tracking system in terms of acoustic sampling interval and visual image size. In addition, in order to estimate the number of *success* of visual compensation anticipated in the multi-modal tracking system, we propose a Statistical Estimation Algorithm (SEA). The SEA takes the acoustic sampling interval and transmission delays of multi-modal sensor data as the algorithmic parameters. To formulate the algorithm, we investigate the transmission delays between the multi-modal sensors and the Server. From the observation, the delays are modeled by Gaussian and Exponential Probability Density Function (PDF). The SEA generates random values to mimic the delays in real environment from the PDFs, and based on the results, it changes its status according to the transition rules. There are R, L, B, and U modes and s and f states in the transition rules. Another proposed algorithm, Statistical and Estimation and Adaptation Algorithm (SEA^2) is suitable for answering how to maintain the tracking system at a certain level of tracking accuracy. The SEA^2 is able to maintain the tracking accuracy by automatically adapting acoustic sampling interval since the accurate tracking depends on the *success* in the visual compensation which subsequently depends on the acoustic sampling interval. The algorithm is composed of Phase 1 and Phase 2. In the first Phase, it runs the SEA to obtain an initial sampling interval. Then, exponential increase and decrease of the

obtained interval are performed to fast adapt the sampling interval to accomplish the target tracking accuracy level.

The outline of this dissertation is as follows. In Chapter 2, we illustrate the network/hardware cross-layer evaluation for compression and packet aggregation on wireless mesh networks. The node algorithm design and optimization for accurate object tracking in multi-modal tracking system is explained in Chapter 3. In Chapter 4, we finally conclude the research works in this dissertation, and mention the future works.

Chapter 2

Network/Hardware Cross-layer Evaluation for Compression and Packet Aggregation on Wireless Mesh Networks

2.1 Introduction

The Wireless Mesh Network (WMN) [1] has recently emerged as a self-configured, low cost, and multiple purpose communication network. It consists of two types of nodes, i.e., mesh routers and mesh clients as shown in Figure 2-1. Mesh routers become merge points of traffic from peer mesh routers or mesh clients. Among the routers, a network service provider might select some as gateways for edge points to connect

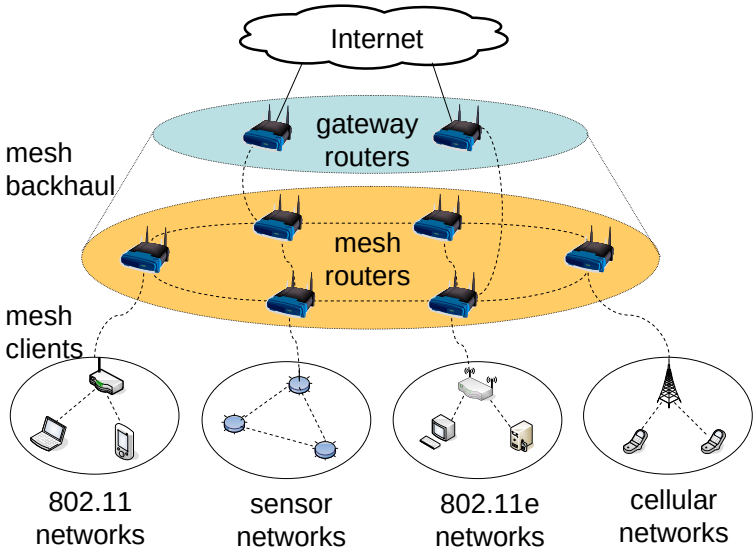


Figure 2-1: Overall architecture of a wireless mesh network.

the outside network. The network constructed by the mesh and gateway routers is called mesh backhaul. Mesh clients access the internet and a processing server, or communicate with other mesh clients via mesh routers. Various types of wireless technology such as WLAN, sensor network, and cellular network might be configured for the mesh client networks.

The WMN capacity is enhanced by the hardware platform refinement, efficient algorithm development, network configuration optimization, etc. For example, a single mesh router could be equipped with network processor to manipulate the CPU-intensive algorithms such as QoS, encryption, and fire-walling. The network processor has a lot of advantages like wire speed implementation and software programmability of specific functions, and consists of several Processing Elements (PE) to handle packets in parallel or pipeline processing. A new metric for establishing routing path helps the selection of high-throughput path from a source to a destination, and a power-

saving router scheduling algorithm extends the overall lifetime of battery-powered WMN. Furthermore, the adoption of efficient hop-by-hop communication algorithm like CRTP, RObust Header Compression (ROHC) [7] and packet aggregation could be alternative solutions for the high throughput and real-timing packet delivery.

As algorithms for general or special applications (such as compression, aggregation, security, audio/video processing modules, etc.) in communication networks have become more complicated, hardware supports for these algorithms are becoming essential parts in the network system design. Before performing hardware/software partitioning and constructing real hardware implementation for an algorithm, it is necessary to anticipate not only the efficiency of the hardware implementation, but also overall network performance affected by the hardware under the system constraints such as speed, complexity and power consumption. Therefore, we strongly believe that it is necessary to have a new type of network simulation methodology for evaluating and optimizing the networks by considering network protocol behavior, application characteristics, and hardware structures.

In this chapter, we propose a network and hardware co-simulation method that is performed and evaluating the real-timing behaviors of WMNs featured by a dedicated hardware platform. In this approach, the co-simulation architecture is implemented by incorporating NS-2 [2] simulator to model the network behavior and SystemC [8] node-level simulator to emulate the hardware behaviors. We use ROHC and packet aggregation for the target algorithms underlying a hardware platform since the sophisticated hop-by-hop operations in ROHC provide robustness and radio resource

savings in a WMN, but it requires hardware support to manipulate the processing-intensive functions. Before advancing to the co-simulation formulation, we first draw the motivation for the hardware realization of the two algorithms in the preliminary works, in which the performance enhancement from ROHC and packet aggregation is initially investigated under small-sized UDP sensor and G.729a encoded voice data. These works are conducted by inserting ROHC and packet aggregation functions into the network module hierarchy within NS-2. In packet aggregation, a fixed number (called aggregation level) of compressed packets are collected to make a large MAC payload data. To explore the output patterns from the simulator, we use various WMN scenarios like the daisy-chain and grid topologies equipped with single and multiple channels. From the extensive simulations, we reach the following results: (1) ROHC itself supports insignificant sensor throughput and voice calls, (2) only packet aggregation without ROHC improves WMN performance, however, its performance has unpredictable patterns under various network and data type environments, and (3) the packet aggregation is required to be cooperating with ROHC to support conspicuous WMN performance in reliable manner. However, the advantages from the algorithm cooperation do not include the algorithm processing overhead on a general purpose processor (GPP). This aspect is investigated by measuring the ROHC execution time from Intel Pentium 4 and RouterBOARD [9], and applying the measured time into NS-2 simulations. From the simulations, we find the high and reliable WMN performance severely deteriorates as the processing overhead in a single mesh router increases. To overcome this problem, we design a hardware architecture for the ROHC

and packet aggregation algorithms, which is conducted by SystemC. The hardware consists of three Processing Elements (PE) configured by packet classifier, parser, en_decoder, (de)aggregator, CRC and memory modules. The architecture is configure by 1 Ghz master clock and 333 Mhz, 16Bytes/cycle read/write single memory BUS structure to reduce the complexity. From this configuration, we get the constant hardware processing times for the two algorithms, and observe a WMN performance after applying the processing times to daisy-chain and grid network scenarios. Based on the previous network configuration and hardware design, the cross-layer evaluation work is conducted by three step profiling procedures. In the first step, we run NS-2 simulation and profile the arrival time of packets to be compressed or decompressed. The second step is the hardware level simulation in which the profiled packet arriving time and other information such as encoding or decoding identifier are processed in hardware modules. While handling packets in this step, the hardware processing delay should record for each packet. Then, the profiled processing time of the hardware components subsequently applies to NS-2 simulation in step 3. At this final step, we can obtain the network level simulation results containing the hardware module delays. From the co-simulation method, we measure the hardware encoding and decoding times which are distributed between 671 ns and 22839 ns, and between 557 ns and 1137 ns, respectively. The sensor throughput and voice quality are also observed from the co-simulations, from which we understand how overall WMN performance is affected by the hardware operations in each mesh router.

In order to characterize the concurrent operations as well as the interaction be-

tween encoding and decoding paths in the hardware architecture, we conduct the numerical analysis by using open Jackson queueing network [10] consisting of $G/M/1$, $M^{[K]}/M/1$, $M/M/1$, and $M/M/\infty$ queue systems, where G means a general arrival, M is a poisson arrival or an exponential service, and $M^{[K]}$ is an arrival with bulk size K. Each queue is mapped into the corresponding hardware element. Since the hardware architecture increases the processing power such that there is no packet drop in the ROHC and packet aggregation module, it is feasible to regard the numerical model as a non-blocking queueing network. From the analysis model, we can get the average system size (L_i) and average system waiting time (W_i) for the individual queue. With sum of the waiting time of each queue and globalized waiting time derived by γ_i and little's formula, we derive the average system waiting time of compression and decompression paths as W_c and W_d, respectively, where γ_i is the arrival rate from outside of the queueing network. From the derived average system waiting time, we can predict the degree of speedup from the designed hardware installation. To validate the accuracy of the numerical model, we use the network/hardware co-simulation method. From the co-simulation results, we show the proposed numerical model accurately predicts the processing delay of the hardware architecture.

The research works in this chapter are based on our previous research outcomes from [11–15].

2.2 Related Works

2.2.1 Network and Hardware Co-simulation

Conventional algorithm design strictly considers a software perspective targeting network processor or general purpose processor. However, it is envisioned that the hardware capability is becoming a limiting factor as algorithms are getting complicated; for example, aggregation, compression, mobility support, and multiple interface router. In order to investigate the impact of the hardware capability such as speed, complexity and power consumption of a platform on an overall network performance, it is necessary to integrate hardware features in a network-level simulation model. This work is done by integrating node-level [16–18] and network-level [2, 19, 20] simulators. The importance of this co-simulation framework has recognized by [21–26]. As indicated in [25,26], the integration of NS-2 and SystemC is much easier to benchmark specialized hardware and processors affecting network performance due to their highly coupled simulation frameworks. However, it is not open to research domain, and hard to use for find-grained hardware component mapping in an architecture design, which will be captured by the co-simulation method proposed in this chapter.

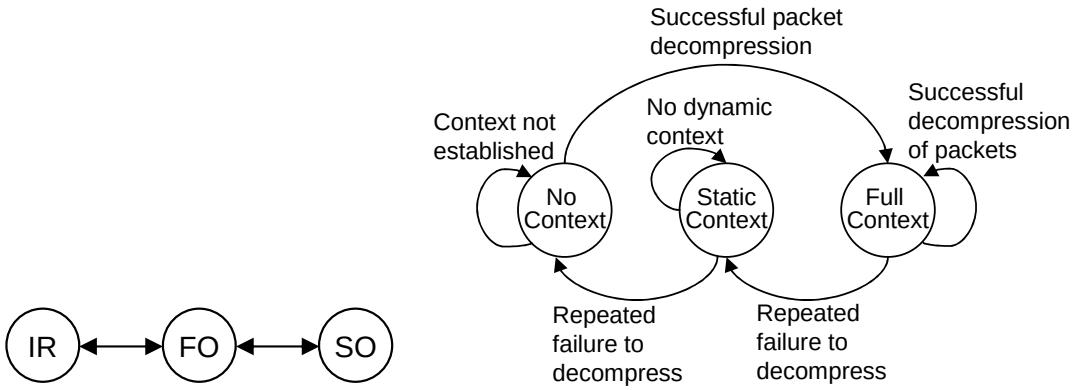
2.2.2 Compression and Packet Aggregation

The research works on network compression in [27–30] are based on delta coding which shows poor performance in high error rate and long round trip time (wireless) links. In order to achieve the stable compression advantage in wireless environment, ROHC [7]

has been proposed. The authors in [31] evaluate GSM encoded voice with ROHC under wireless link. They show ROHC cuts the bandwidth for voice communication over wireless links in half. [32] proposes routing-assisted header compression algorithm in ad-hoc wireless environment. The algorithm relies on information from routing protocol, in which all the ad-hoc nodes update a context information depending on the current network topology and the routing path. Authors from [33] mention static configuration of the ROHC parameters, and propose negotiated dynamic parameter setup of ROHC under UMTS radio networks.

ROHC is currently standardized and built in software-driven commercial wireless products. It provides compression profiles for RTP/UDP/IP, UDP/IP, ESP/IP, etc. In these protocol suits, there are statistically unchanged fields like source/destination IP addresses and TCP ports. Some fields like IP identifier field are changed in a specific pattern, and other fields such as UDP checksum are randomly changed. ROHC performs hop-by-hop operations, so that its operations should work between only two nodes and remain transparent to the other network entities. The network nodes maintain a Context which is a collection of states such as current status of static and dynamic header field in a stream.

The encoder (compressor) and decoder (decompressor) operate based on three states as shown in 2-2. The encoder has Initialization and Refresh (IR), First Order (FO), and Second Order (SO) states. The encoder should start in IR state where it sends the complete packets containing static and dynamic field information to establish a new Context. In FO state, static fields are recognized between encoder



(a) Encoder states (b) Decoder states

Figure 2-2: Encoder and decoder state transition diagrams in ROHC.

and decoder while the dynamic fields are not. The SO state performs more reduction of information by establishing dynamic fields as well as static fields in the Context. In this case, only minimum information to advance the normal sequence of encoded header fields is delivered. In case of decoder, it makes transition between No Context, Static Context, and Full Context in response to the packet information from encoder. In No Context state, any decoding information in the decoder Context is not available. After a decoder receives IR packet, it moves to Full Context state where the decoder moves down to Static Context state, repeatedly No Context state whenever error conditions happen. In Static Context state, the decoder might receive FO packet from encoder. At this point, it moves back to Full Context after restoring the Context information from the FO packet.

In wireless network, various communication link status could be existing. Therefore, the ROHC adopts three modes to offer the fast adaptation in the link changes; Unidirectional mode (U-mode), bidirectional Optimistic mode (O-mode), and bidirectional Reliable mode (R-mode). In U-mode, an encoder begins in IR state and

autonomically moves up to the other states after it successfully sends a specific number of IR and FO packets. The decoder downwards to lower states when its time-out mechanism indicates a transition, or feedback packet is received from a decoder. In O-mode, an encoder can change its state transition by autonomic approach of U-mode and decoder's feedback requesting for error recovery or indicating the successful Context update. The R-mode frequently uses feedback channel to avoid the information damage in the Context. The encoder can change it states only after it receives feedback from a decoder, so that this mode provides highest confidentiality in wireless communications.

When small-sized packets like sensor and VoIP are delivered on a network, the relatively large packet header size causes the waste of network resources to deliver a unit number of payload data. To resolve this problem, some researchers propose packet aggregation methods. [34] mentions end-to-end and hop-by-hop aggregation, and hybrid aggregation schemes on wireless mesh network with CRTP compression algorithm. Two aggregation algorithms, namely, forced and adaptive algorithms are proposed to mention 802.11-based protocol overhead and the importance of the packet aggregation in an ad-hoc network in [35]. The effective delivering of voice signals over IP networks based on packet aggregation is proposed in [36]. [37] investigates the relationship between the number of voice sources, out-link rate and certain teletraffic metrics in wireless networks based on aggregation schemes of multiple VoIP flows. Even though these research works enhance the quality of voice and data transmission over radio link, their solutions are software-driven approach, and do not mention the

effect of hardware design of compression and aggregation under multi-hop wireless mesh networks, which is mentioned in this chapter.

2.2.3 Functional Analysis of ROHC and Packet Aggregation

In this section, we analyze interaction between compression, decompression, and aggregation as well as functional complexity of ROHC. Even if the behaviors of ROHC and packet aggregation depend on design decision, the analysis can provide fundamental insight the reason why the adoption of ROHC and packet aggregation affects the multi-hop mesh network performance.

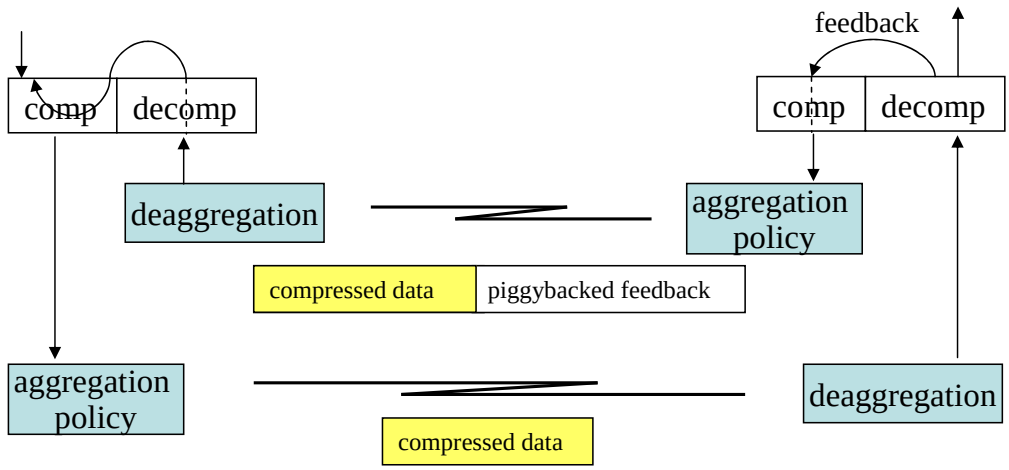


Figure 2-3: Interaction between functional blocks via wireless link.

We first look into the interaction between compression, decompression, and packet aggregation in Figure 2-3. After compressing a packet header, a mesh router aggregates the compressed packets according to its aggregation policy. Peer routers receiving the aggregated packet perform deaggregation and decompression, sequen-

tially. In the meanwhile, it sends a feedback packet in the form of piggyback. In this interaction, we note that aggregation policy and the frequency of the feedback affect the mesh network performance. A multi-hop mesh network could differently responses to the number of aggregated packet and aggregation policy. Since the radio resources in a wireless environment is essential factor for communication, we need to decide which mode of ROHC must be used for a mesh network. In this chapter, we use U and O modes so that compressor and decompressor respond to peer routers while saving the radio resources. Another factor for the mesh network performance

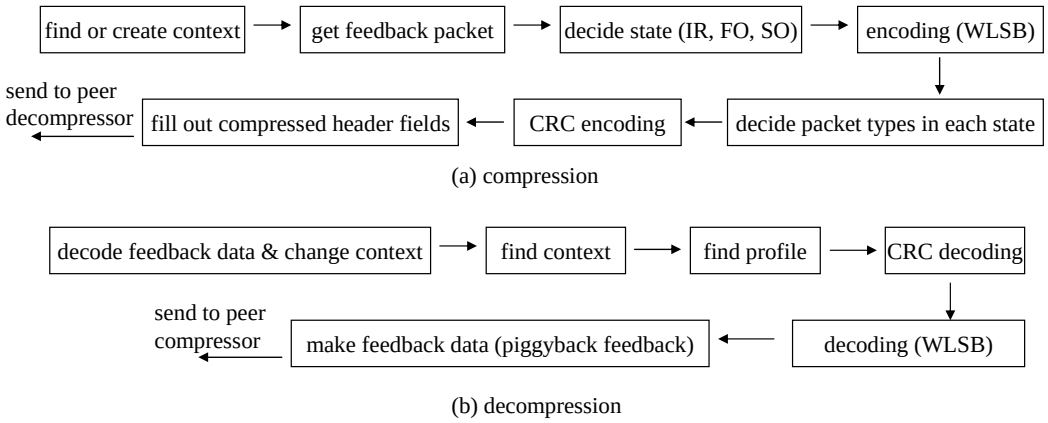


Figure 2-4: Functional complexity of compression (a) and decompression (b).

is the functional complexity of compression and decompression modules. We break down the ROHC functions in several steps in Figure 2-4. In conjunction with the elaborated and sophisticated interaction between compressor and decompressor, the functional steps lead to the increase of complexity for the compression algorithm, which requires relatively high memory access for context, profile and feedback and computation power such as WLSB and CRC calculation. Therefore, in the following section, we measure the ROHC execution time on a GPP, and verify the impact of

ROHC processing time in a single mesh router on overall mesh network performance.

2.2.4 Hardware Realization

Network processor has been highlighted as the wire speed and programmable alternative to ASIC implementation in current multi-demanding network service environment. It places between network and switch fabric interfaces, and undertakes a processing-intensive or application specific functionalities independent of general purpose processor (GPP). The authors in [38] perform the evaluation of Differentiated Service [39] implementation on IXP1200 [40] network processor. They identify the processing bottleneck in micro-engines and SRAM in the classification module as well as SDRAM in IP forwarding engine. The MPLS implementation in the same network processor is investigated in [41], in which the authors Label Switching Router (LSR) in the processor provides smaller or larger processing delay against an IP router according to packet size and outgoing link rate.

In the next generation cellular network, Base Station System (BSS) has to handle large volume of voice and data, in which the complicated functions in ROHC becomes a processing overhead in a GPP. Therefore, [42] designs a network processor incorporated with primary ROHC function block to resolve the drawbacks of the software-driven ROHC implementation targeting Xilix Vertex-E (FPGA) and IBM CU-11 (ASIC). It analyzes required memory bandwidth for ROHC functions, and measures hardware resource usage of the ROHC functional components. However, each component has to interact with memory and other components to compress/decompress

a packet, which may cause systematic variation on the hardware performance. This aspect will be examined in this chapter.

Even though the previous research outcomes mention the performance improvement by hardware realization in the form of network processor, their results are concerned with wired networks or one-hop based cellular networks. In order to explore the impact of the hardware design in multi-hop wireless mesh network, we observe the effect of the ROHC and packet aggregation in our previous work [13]. However, its evaluation is only based on single data type, UDP, and the hardware architecture is partially designed such that only encoding path of the ROHC is implemented. Thus, in [11], we evaluate the hardware platform of the two algorithms by using proposed network/hardware co-simulation method. The hardware platform consists of full dimension of the compression and decompression paths. Additionally, we numerically analyze the hardware architecture by using queueing model in [12].

2.2.5 Voice over IP (VoIP)

IP networks transfer voice data cheaper than PSTN network by sharing network resources, so that it has recently been popular for voice transmission. In order to transform analog to digital signal, VoIP applications use codec such as G.729a, G.723.1 etc, which creates almost constant bit rate voice data, but has different traffic characteristics. For example, G.729a generates 20 bytes packets by the 20 *ms* interval; G.723.1 does 30 bytes packet by 38.46 *ms* interval. The G.729a codec is popularly used by VoIP applications like Zyxel Prestige and Senao S7800H. When a VoIP ap-

plication uses silence suppression where no packet is transmitted when silence in a call is detected, it reduces 50% network traffic generated. In this chapter, we assume a VoIP application uses G.729a codec and the silence suppression is used to save the limited wireless resource by reducing void data created during silence period. In order to measure voice quality, we use R-factor for G.729a as indicated in [43] as follows.

$$\begin{aligned} R = {} & 94.2 - 0.024(d_{network} + 85) \\ & - 0.11(d_{network} - 92.3)H(d_{network} - 92.3) \\ & - 11 - 40ln[1 + 10(e_{network} \\ & + (1 - e_{network})e_{dejitter})] \end{aligned}$$

, where $d_{network}$ is network delay, $e_{network}$ is network loss, $e_{dejitter}$ is jitter loss, and

$$H(x) = \begin{cases} 1, \text{ if } x > 0 \\ 0, \text{ otherwise.} \end{cases}$$

In this calculation, it is assumed that voice play starts after $60ms$ dejitter buffer delay. If the R-factor is over 70, VoIP applications obtain tolerable (good) quality for communication. [44] indicates good quality of voice call is sustained by below 150 ms network delay and 0.05 of network plus jitter loss, and very sensitive to the losses. In this chapter, a voice call is considered as success call if R-factor is over 70.

2.3 Preliminary Works

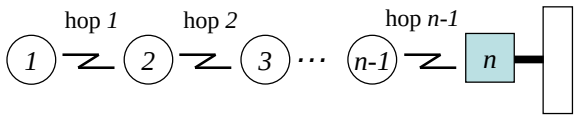
This section investigates the effect of ROHC and packet aggregation when they are applied to multi-hop WMNs. The evaluation works are performed by NS-2 under sensor and voice traffic.

Table 2.1: Simulation parameters.

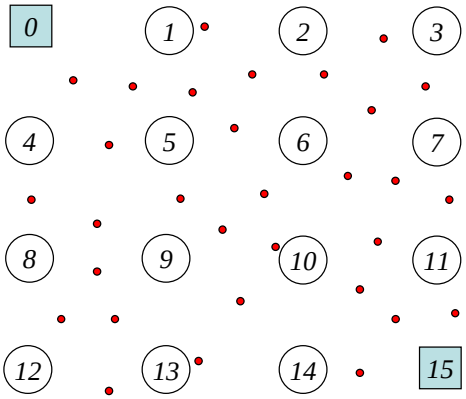
traffic type	sensor data (Constant Bit Rate), voice traffic encoded by G.729a codec
payload size	20 bytes
interface card type	802.11g (54Mb/s) PC card
queue size	200 packets
wireless error	uniformly distributed random bit error rate (BER) = 0.0005
routing protocol	AODV

We modify and port real implementation of ROHC [45] into the simulator, and use simulation parameters as shown in Table 2.1 throughout this section. For sensor data, we assume each sensor is composed of Mica Mote [46] or Telos Mote [47], and all the sensors generate aggregated Constant Bit Rate (CBR) sensor traffic. Each voice is encoded by G.729a codec, and consists of two uni-directional flows, i.e., caller and callee. Therefore, a call is considered as success when two flows have more than 70 R-factor values. For data transmission, single and multiple channel 802.11g (54Mb/s) wireless links are used for mesh routers. We use 0.0005 for uniformly distributed random bit error rate (BER) for the wireless link error.

Daisy-chain and grid topologies are used for the simulations as shown in Figure 2-5, where we assign router id to each mesh router for readability. For the daisy-



(a) Daisy-chain topology with single or multiple channel



(b) Grid topology with single channel

Figure 2-5: Simulation scenario constructed by wireless mesh routers. The router drawn by rectangle is gateway, and red points represent senor and voice traffic sources

chain topology, single and multiple channel are installed, and *router n* has a role in a gateway. The grid topology uses single channel for the communications. Since there is severe channel interference among router transmission in the topology, we place gateway functions into *router 0* and *router 15*, so that half of the routers access wired network through *router 0*, and the other half do via *router 15*. In these simulation works, we assume that the configured WMNs are fully utilized, so that sufficient amount of sensor and voice data are created during the simulations.

2.3.1 Effect of ROHC

In this section, we identify the effect of ROHC itself on the WMN performance. The ROHC behaviors are observed under single channel daisy-chain topology which

delivers sensor or voice data. The daisy-chain topology is composed of 1 to 7 hop counts. For sensor data, we assume 40 Telos Motes are attached into *router 1* and a sensor server is accessed via gateway, so that total 10Mb/s sensor traffic is forwarded to the sever. In case of voice traffic, 100 voice calls generated by G.729a codec are randomly located on the topology.

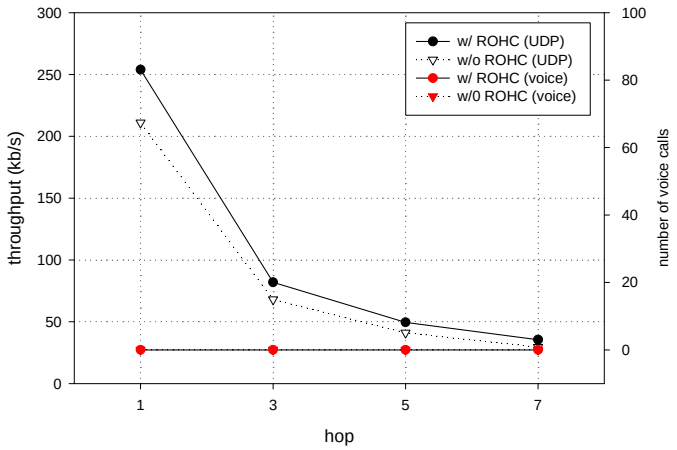


Figure 2-6: Sensor throughput and the number of voice calls supported when only ROHC is used or not.

Figure 2-6 shows simulation results for the case only ROHC is used (w/ ROHC) or not (w/o ROHC). We find out the high sensor throughput is achieved under 1 hop scenario for both w/ ROHC and w/o ROHC. However, as the hop count increases, the throughput shows steep degradation, which stems from the channel interference inherent in 802.11-based MAC protocol. Regardless of this gradual throughput degradation, we observe the WMN with ROHC attains more throughput than non-ROHC network. The actual ROHC usage gain in sensor throughput is 20% for all the different hop scenarios. In case of voice call, the WMN does not support successful voice calls. From the measurement of voice quality parameters, we find this is because

voice quality is mainly affected by large $e_{network}$ value which results from a lot of wireless collision when 100 voice calls contend to use the limited wireless channel. This phenomenon causes each voice flow to have R-factor value smaller than 70.

2.3.2 Effect of Packet Aggregation Cooperating with ROHC

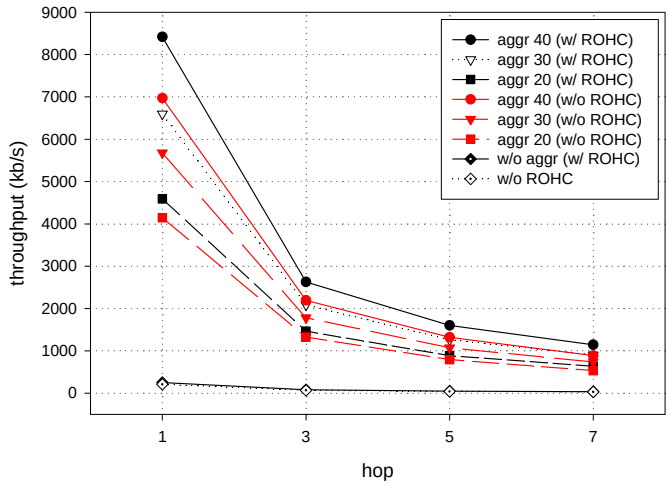
In this section, it is observed how the packet aggregation accelerates the WMN performance. This work is performed under two types of data scenarios, single data type (sensor or voice) delivered by the same network configuration as Section 2.3.1 and multiple data types composed of both sensor and voice. In multiple data transmission, we have two network scenarios: (1) single channel grid topology, where we assume 4 Mica Motes generate constant UDP-sensor data within the wireless coverage of each mesh router except for two gateways that have destinations of the Motes, and 50 voice sources are randomly positioned on the network, and (2) 2 and 3 hop daisy-chain topology equipped with multiple channel interface, in which it is assumed that 20 Telos Motes construct sensing region under each mesh router except the gateway router, and 200 voice calls are randomly located on the topology. In this section, we use a simple next-hop based aggregation algorithm, in which each mesh router maintains aggregation table containing peer routers as the next hop. The table keeps aggregation level which means how many compressed or normal packets are collected. After a router searches the packet's next hop based on its destination, the packets are added into accumulation buffer until the number of aggregated packets reaches the aggregation level. Immediately after arriving at the aggregation level, the mesh

router sends the aggregated packet to its next-hop router.

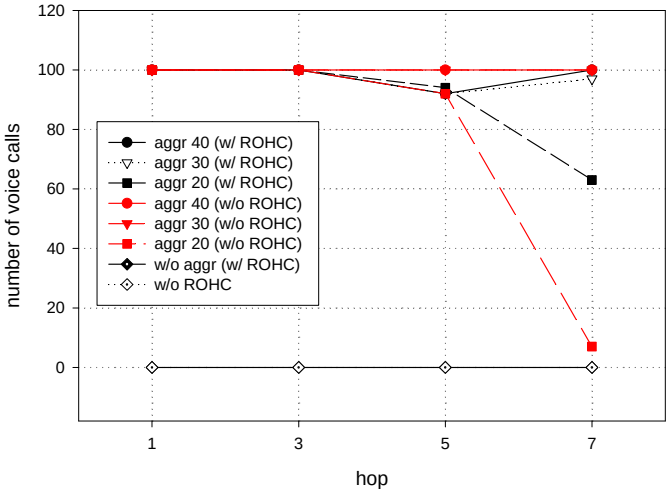
In the following subsections, we show only packet aggregation (aggregation w/o ROHC) improves WMN performance, however, its performance has unpredictable aspect when network environment and data type (sensor or voice) change. On the other hand, packet aggregation cooperating with ROHC (aggregation w/ ROHC) provides high and reliable sensor throughput and successful voice calls under an appropriate aggregation level even in an extreme WMN environment.

Single Data Type Scenario

In Figure 2-7, we compare WMN performance from various aggregation level (20 to 40) when packet aggregation w/o ROHC or w/ ROHC is used at mesh routers to process the sensor or voice traffic. For comparison, the simulation results from Section 2.3.1 are also included. We can understand that packet aggregation w/o ROHC provides significant improvement for sensor throughput and successful voice calls as denoted by red point lines. In one hop scenario, around 27 times sensor throughput and 100 times voice call improvements are achieved at 40 aggregation level compared with the case without aggregation. Even if the absolute value of the sensor throughput decreases as the number of hop count increases, the relative sensor throughput gain compared with the two non-aggregation cases is maintained by more than 25 times improvement. For the voice calls, the aggregation provides significant improvement in the number of successful voice calls by keeping the R-factor value over 70 except for aggregation level 20 that shows steep degradation in



(a) Sensor throughput



(b) Number of voice calls

Figure 2-7: Sensor throughput and the number of voice calls supported when single data type is delivered over daisy-chain network.

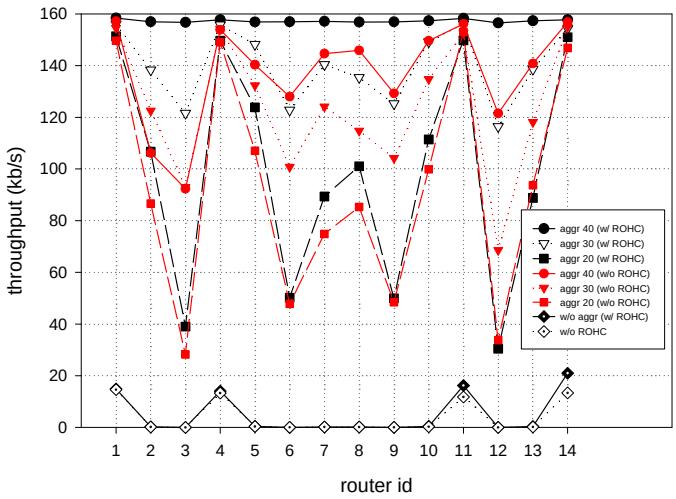
7 hop scenario as indicated by red rectangle point in Figure 2-7(b). The effect of aggregation is accelerated by reduction of packet size contributed by ROHC. The actual gain from packet aggregation w/ ROHC compared to packet aggregation w/o ROHC is over 20% on aggregation level 40, and 12% for aggregation level 20 for sensor throughput. The voice calls of aggregation w/ ROHC case have similar trends

to those of aggregation w/o ROHC. However, 20 aggregation w/ ROHC supports 60 voice calls which is an outstanding improvement compared with 20 aggregation w/o ROHC. The small gap of voice calls between packet aggregation w/ ROHC and w/o ROHC is to be the random placement of voice sources.

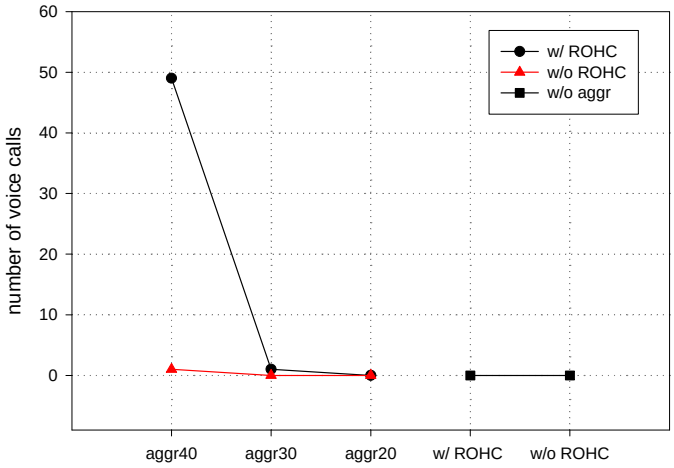
Multiple Data Type Scenario

In this scenario, each mesh router accepts data from both sensor and voice sources, so that WMN is more heavily loaded and realistic than single source scenarios in previous sections.

Figure 2-8 shows sensor throughput and successful voice calls supported by the single channel grid network. In Figure 2-8(a), we can observe that the aggregation w/o ROHC provides significant throughput improvement similar to Figure 2-7(a). However, the achieved throughput has unstable and large variance among WMN routers in all the aggregation levels. In the aggregation w/ ROHC, the unstable throughput is gradually balanced as the aggregation level increases. Note that in aggregation level 40, the aggregation w/ ROHC provides approximately same total sensor throughput (160kb/s) generated by 4 Mica Motes for all the mesh routers. In case of voice, we find out the number of successful voice calls is severely affected by aggregation level and whether ROHC exists or not as shown in Figure 2-8(b). 40 aggregation w/ ROHC supports almost all the injected voice calls; however, the aggregation w/o ROHC achieves around zero calls. This result stems from loss, delay, and jitter sensitiveness of voice traffic, that is, as aggregation level increases,



(a) Sensor throughput



(b) Number of voice calls

Figure 2-8: Sensor throughput and the number of voice calls supported when multiple data type is delivered over grid network.

aggregated packet size in aggregation w/o ROHC increases faster than w/ ROHC, which in sequence, leads to small R-factor value due to large values of $d_{network}$, $e_{network}$, and $e_{dejitter}$.

Figure 2-9 shows the R-factor distribution of each voice flow in the grid topology with 40 aggregation level. Since the grid topology accepts 50 voice calls, and one voice call is composed of two uni-directional flows, 100 R-factor values are plotted

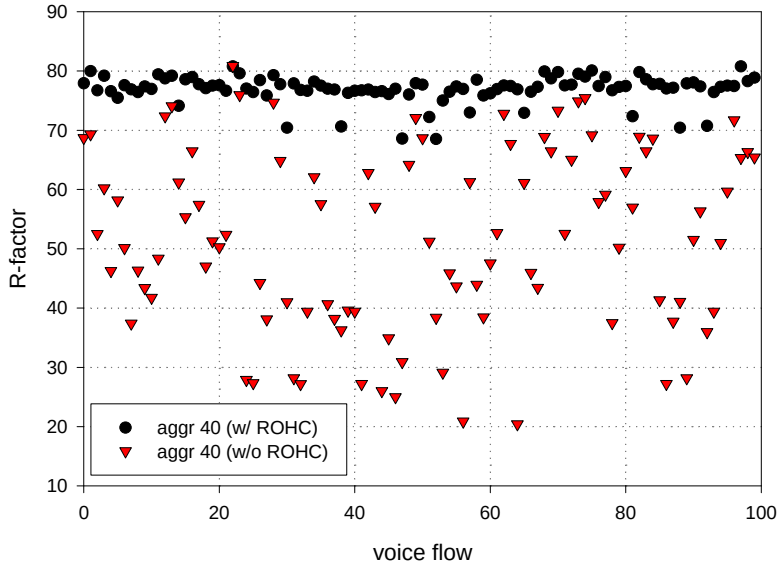


Figure 2-9: R-factor distribution of voice sources appeared in grid topology.

in the figure. We can understand why the aggregation w/ ROHC provides higher voice quality than w/o ROHC case. Most of the voice flows in the aggregation w/ ROHC are around the 80 R-factor. Though, the voice flows in aggregation w/o ROHC denoted by red triangle achieve R-factor values below 70 with large variations.

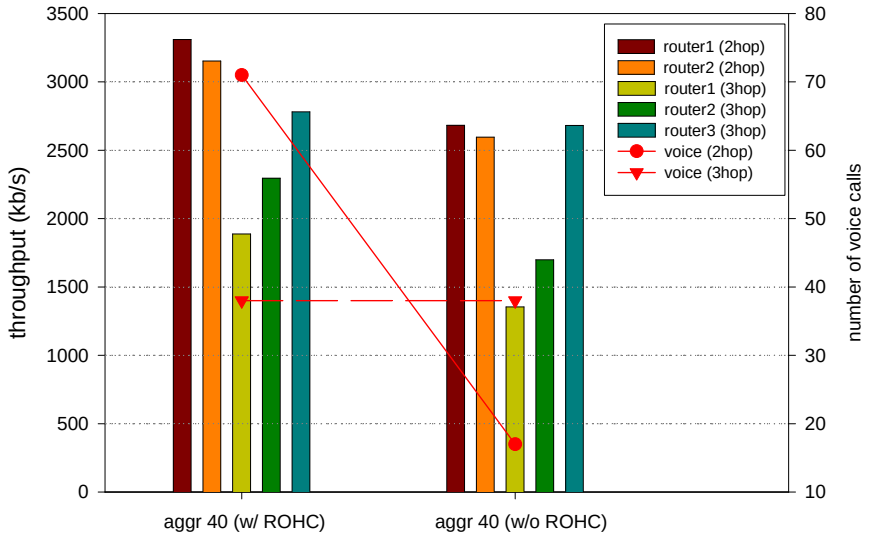


Figure 2-10: Sensor throughput and the number of voice calls supported when multiple data type is delivered over daisy-chain network with multiple channels.

For the case mesh routers are equipped with multiple interfaces, we plot the results for sensors and voices in Figure 2-10, in which we show only 40 aggregation level case. Similar to grid scenario, aggregation w/ ROHC shows higher sensor throughput than aggregation w/o ROHC in 2 hop and 3 hop scenarios. In case of voice, we accomplish around 70 successful voice calls in 2 hop network, and around 40 calls in 3 hop case when a WMN adopts both ROHC and packet aggregation indicated by aggr 40 (w/ ROHC). However, the network with only aggregation (aggr 40 (w/o ROHC)) shows different results. We have below 20 voice calls in 2 hop scenario, and around 40 calls in 3 hop network. If we compare this result with Figure 2-7(b), we find the voice call change from 2 hop and 3 hop case is unexpected. Since the supported voice calls are 100 for 1 and 3 hop cases in Figure 2-7(b), we might guess we can get 100 calls in hop 2, and this is true. However, the guess is not applicable to the multiple data type environment with multiple channels in Figure 2-10. Note the voice sources in 2 hop topology reside in narrower space than those in 3 hop case, and the voice quality is affected by the sensor data. We infer the real-timing sensitiveness of voice is highly affected by this combinational network circumstances, which leads to the unexpected result in the voice quality measurement. Another factor to affect the high and reliable performance from ROHC and packet aggregation is the processing-intensive characteristics of the ROHC algorithm. In following section, we investigate this aspect.

2.3.3 Effect of Processing Time of ROHC and Packet Aggregation

When we run the complex ROHC and packet aggregation algorithms in a GPP-based system, we need to identify the encoding or decoding overhead, and the overall performance of a WMN. In order to conduct these works, we first measure the ROHC encoding (compression) and decoding (decompression) times in Intel Pentium 4 and RouterBOARD 230.

Table 2.2: Pentium 4 and RouterBOARD specification and measured processing times for ROHC encoding and decoding functions.

system type		Pentium 4	RouterBOARD
CPU speed		3.2 Ghz	266 Mhz
main memory		1 Gbytes	256 Mbytes
peripheral devices		L2 cache: 2 Mbytes	BIOS: 2 Mbytes
operating system		Linux	Linux
ROHC processing time per packet (en/decoding)	UDP/IP	$10.107\mu s/11.904\mu s$	$66.774\mu s/50.938\mu s$
	Voice (RTP/ UDP/IP)	$26.613\mu s/21.260\mu s$	$248.929\mu s/168.258\mu s$

Table 2.2 shows the the system specifications of the two commercial products and the measured times. The RouterBOARD is suitable for building up an enterprise WMN by installing multiple 802.11 b/g/a Wi-Fi interface cards. The measured encoding and decoding times are the average values from 3000 packets. Voice processing time is larger than UDP time since the GPPs in both products need to handle RTP fields in addition to UDP/IP. Note the increment of RouterBOARD processing time from UDP/IP to RTP/UDP/IP is larger than that of Pentium 4. In order to inves-

tigate the effect of the single router's processing overhead on WMN performance, we apply the measured processing times into NS-2 simulations. This application from real measurement to NS-2 is justified from [44] where the authors use RouterBOARD and click router [48] for wireless mesh testbed, and indicate the number of voice calls in the real testbed is almost same as NS-2 simulations. The simulation results in this section include the waiting time to (de)aggregate packets as well as ROHC processing time. We assume that a mesh router uses 40 aggregation level.

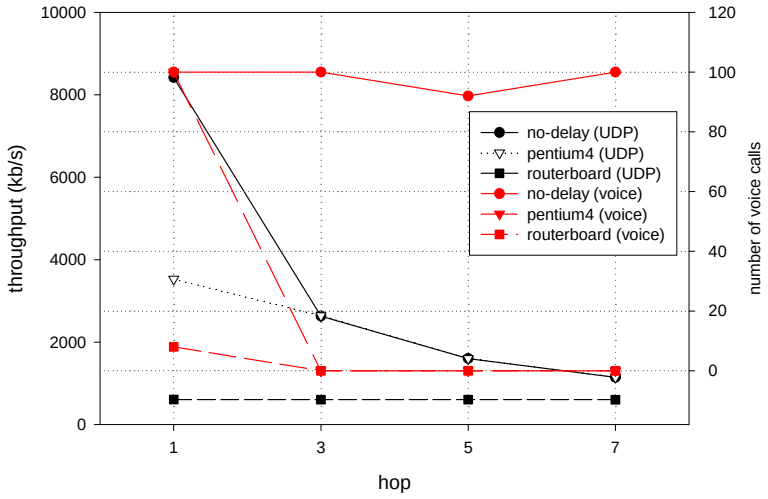


Figure 2-11: Sensor throughput and the number of successful voice calls affected by processing overhead: single data type under daisy-chain topology.

Figure 2-11 to 2-13 show the simulation results after we apply the measured ROHC processing times to the NS-2 simulations. The simulations are performed based on the scenarios described in Section 2.3.1 and 2.3.2. In the figures, we also plot no-delay case, which is the ideal case that ROHC encoding and decoding have no processing times, so that a GPP has zero time to handle the encoding and decoding functions. Figure 2-11 is for the daisy-chain topology with single data type and single channel interface. We can observe that in one hop case, the sensor throughput in no-delay case

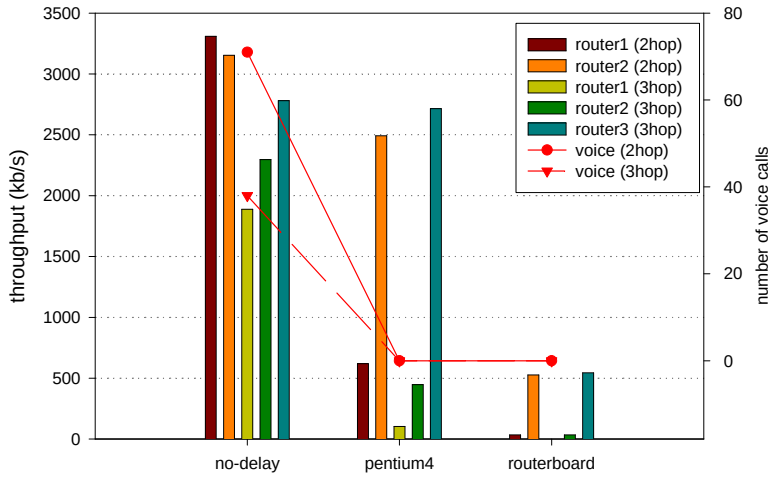
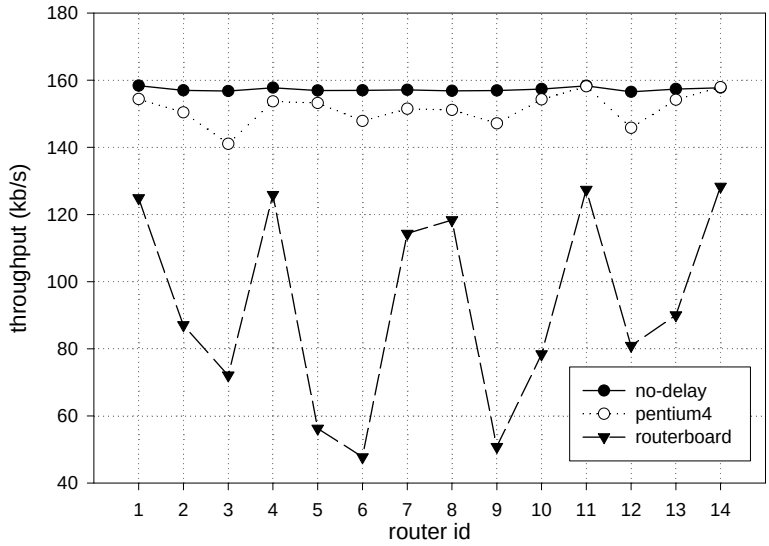


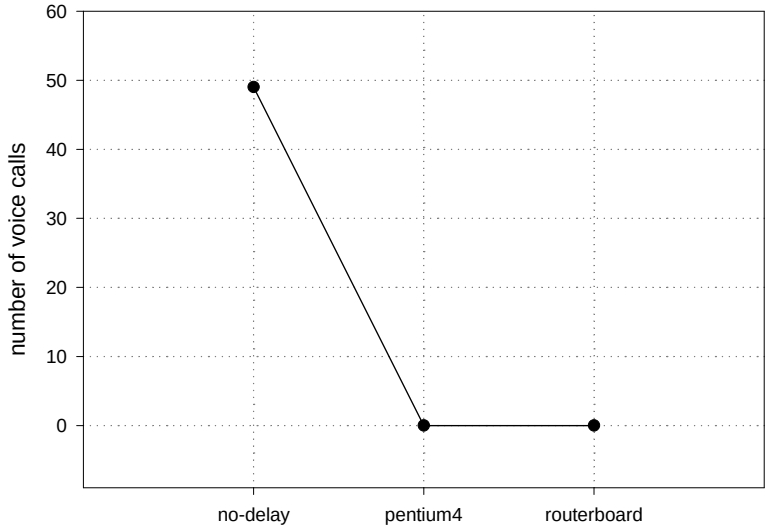
Figure 2-12: Sensor throughput and the number of successful voice calls affected by processing overhead: multiple data type under daisy-chain topology.

is around 2.4 and 5.8 times larger than Pentium 4 and RouterBOARD, respectively. However, as the hop count increases (channel interference increases), the processing overhead has minor effect on the sensor throughput achievement. In case of voice call, we do not obtain the outstanding increase of successful voice calls from ROHC and packet aggregation in both Pentium 4 and RouterBOARD in more than one hop count scenarios. However, if the interference is eliminated by adopting multiple interface as shown in Figure 2-12, we understand processing overhead has major impact on the sensor throughput and voice performance. In ideal case, more than 2Mb/s sensor throughput is achieved throughout the 2 and 3 hop cases, and around 70 and 40 voice calls are supported in both cases. In case of Pentium4, we can observe that there is a significant drop in the throughput and the voice calls. If the RouterBOARD is used, the degradation is more serious such that the WMN supports below 500 kb/s in sensor throughput and zero voice call.

Figure 2-13(a) and Figure 2-13(b) show the sensor throughput and voice calls un-



(a) Sensor throughput



(b) Number of voice calls

Figure 2-13: Sensor throughput and the number of successful voice calls affected by processing overhead: multiple data type under grid topology.

der grid topology with multiple data type scenario. For the sensor data stream, the GPP in Pentium 4 provides lower throughput than the ideal case, and the RouterBOARD processor shows significant performance degradation with large variation among the routers. In case of the voice having real-timing characteristics, there exists more serious deterioration of the supported calls in the two commercial products.

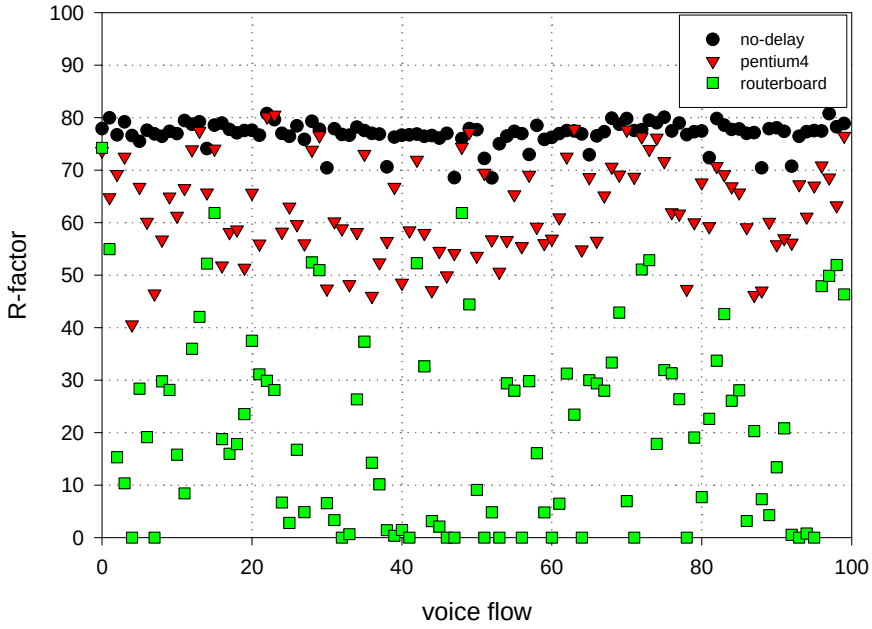


Figure 2-14: R-factor distribution appeared in the WMN built by GPP-installed commercial products.

Figure 2-14 shows the R-factor distribution appeared in the grid topology equipped by the Pentium 4 and RouterBOARD. The figure gives the reason why high voice quality is not supported in the two commercial products. The R-factor values in Pentium 4 have variations between 40 and 80, where most of the values are below 70. In case of RouterBOARD, all the R-factor values are smaller than 70. Note Pentium 4 processor is not appropriate but RouterBOARD has relatively suitable

processor for mesh routers in small/large enterprise mesh networks. Here, we have insight that we can not obtain the high and reliable performance achievement from the two algorithms due to the lack of processing power if we construct a wireless mesh network by using current software-driven commercial products, and adopt ROHC and packet aggregation to enhance the mesh performance.

2.4 Hardware Architecture Design and Evaluation

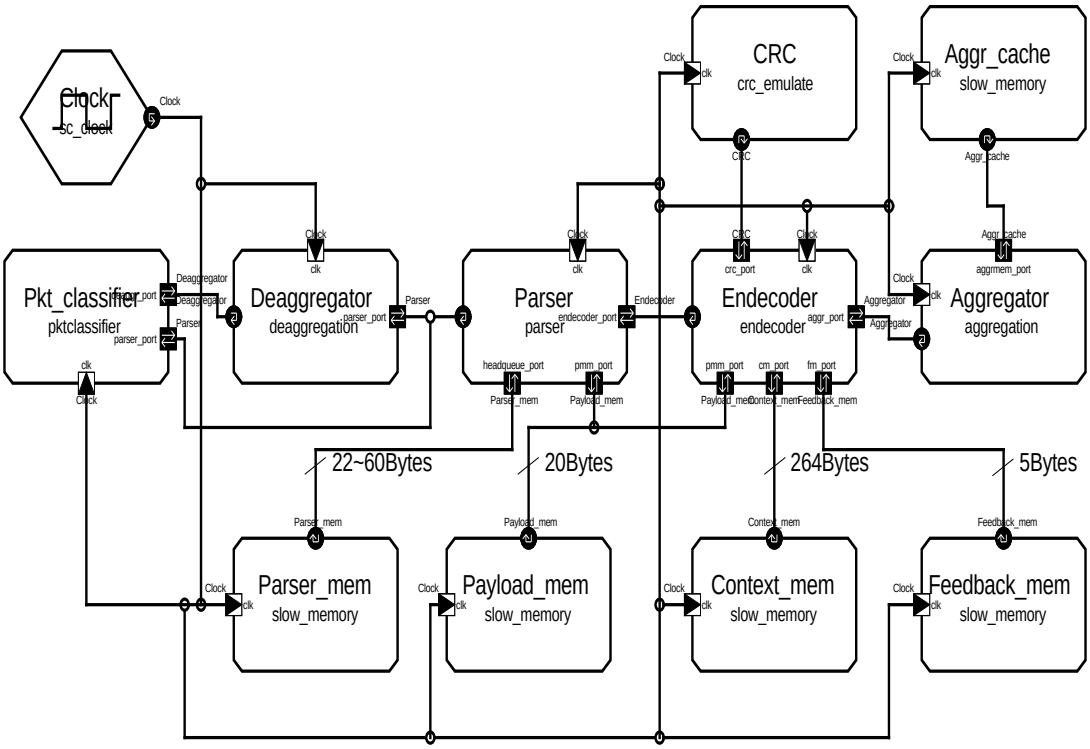


Figure 2-15: Hardware architecture for ROHC and packet aggregation, which is modeled and simulated by SystemC.

The hardware architecture design is conducted by SystemC, which makes it possible to design and simulate concurrency, parallelism and real-time communication of

a hardware system. Figure 2-15 shows the proposed hardware architecture, in which we separate ROHC functions as packet classifier, deaggregator, parser, en_decoder, CRC, aggregator and memory modules. Each module is concurrently executed while accessing memory for parser, payload, ROHC context, feedback, and aggregator.

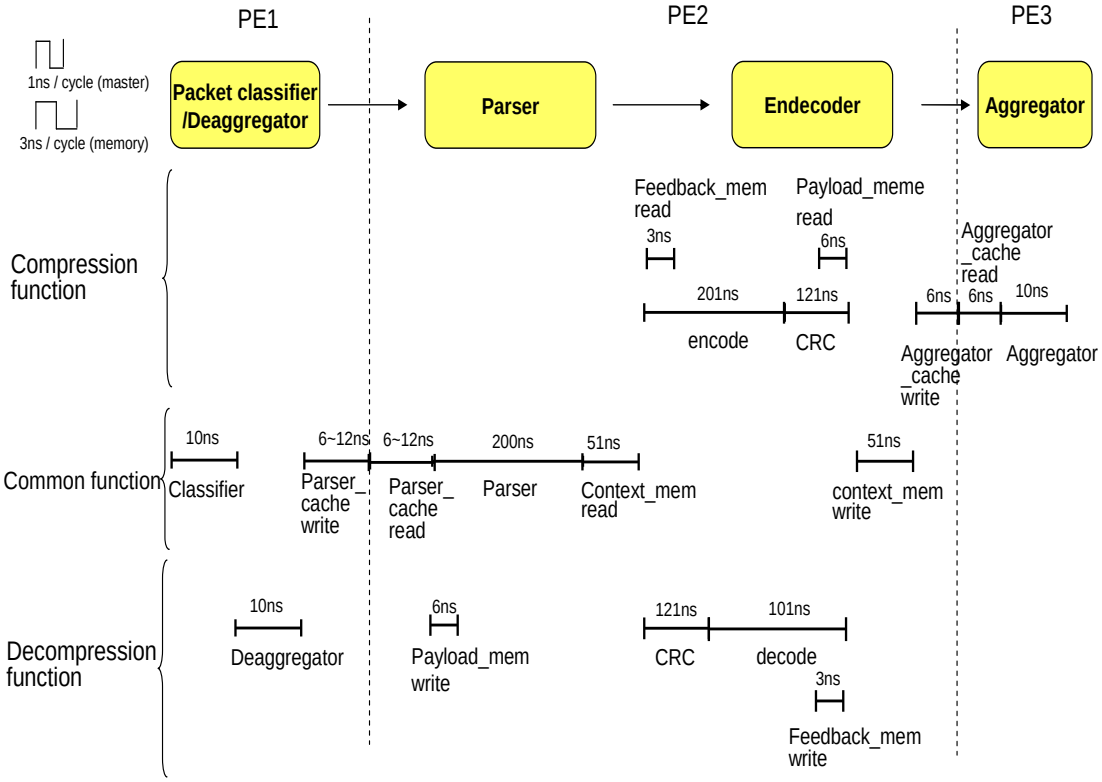


Figure 2-16: Temporal and operational sequence of the hardware architecture. We assume three PEs are used to manipulate each component of the system.

We also present the operational steps and the processing delay of each hardware component in Figure 2-16. We obtain the delay times for the components after analyzing the iterative FPGA en_decoder implementation proposed in [42] which breakdowns resource usages of each ROHC functional component made by FPGA and ASIC implementation. As indicated in Figure 2-16, the designed hardware system is composed of 3 Processing Elements (PEs). PE1 handles packet classifier and deag-

gregator. The next complicated parts, i.e., parser, en_decoder, CRC are handled by PE2. And the aggregator is done by PE3. In realistic implementation, the 3 PEs independently manipulate the allocated functional modules.

When a packet from routing module (upper layer) or MAC (lower layer) arrives at ROHC and packet aggregation unit, packet classifier first identifies whether the packet is forwarded toward encoding or decoding path. On the encoding path, a packet is directly forwarded to the parser. However, an aggregated packet from MAC layer is first handled by the deaggregator which extracts each compressed packet by the amount of aggregation level and passes the packet to the parser. After the separate processing, the packets from routing module or deaggregator are buffered into parser memory. As soon as the parser is available for a packet, it detaches packet header fields. Here, we assume the payload of a packet waiting for encoding has already saved into payload memory by a router processor in front of ROHC and packet aggregation module, and the payload is attached into the compressed header at the end of encoding procedure. Therefore, while a packet is manipulated by the parser, payload of the packet is saved into payload memory only if the packet is on decoding path; otherwise, the memory access is jumped over. After parsed, the packet is processed by en_decoder, which performs encoding and decoding. The en_decoder accesses the context and feedback memory while encoding and decoding packets. CRC module is also called by the en_decoder to check or fill out the CRC field in a ROHC header. When the en_decoder uses ROHC context, it accesses context memory before starting encoding or decoding. The fetched context information becomes a critical

section; therefore, other packets can not proceed to en_decoder while a packet is encoding or decoding. This exclusive usage of context information is necessary since it is possible for more than one packets to access the same context contents. In case a decoded packet contains feedback data, en_decoder accesses feedback memory to save the data, which is sequentially used during encoding process to determine the ROHC mode or state. After finishing en_decoding procedure, the modified context is saved into context memory. All the encoded headers are then attached by payload after accessing payload memory, and the completed packet is put into aggregator cache. In case of the decoded header, it is forwarded into routing module for the next manipulation. Note that we do not access the payload memory at the end of decoding path since we assume the routing module at the upper layer handles the payload after finishing routing-related works such as routing table lookup, internal switching, etc. If the number of encoded packets reaches an aggregation level, the aggregator creates an aggregated packet and forwards it to MAC layer. We define the parser, payload, context, and feedback data size as 22-60, 20, 264, and 5 bytes, respectively. In this design, we use 1 ns for the master clock cycle, and 3 ns for memory access which models the 333 Mhz and 16 bytes/cycle read and write memory BUS. This means that access time for parser memory is 6-12 ns; payload memory is 6 ns; context memory is 51 ns; feedback memory is $3ns$; aggregator cache is 6 ns. Note that memory accesses for 5 memories are not overlapped by making each access parallel to computation units. This means the designed architecture uses single BUS system which reduces the hardware complexity. From the architecture design, we get the

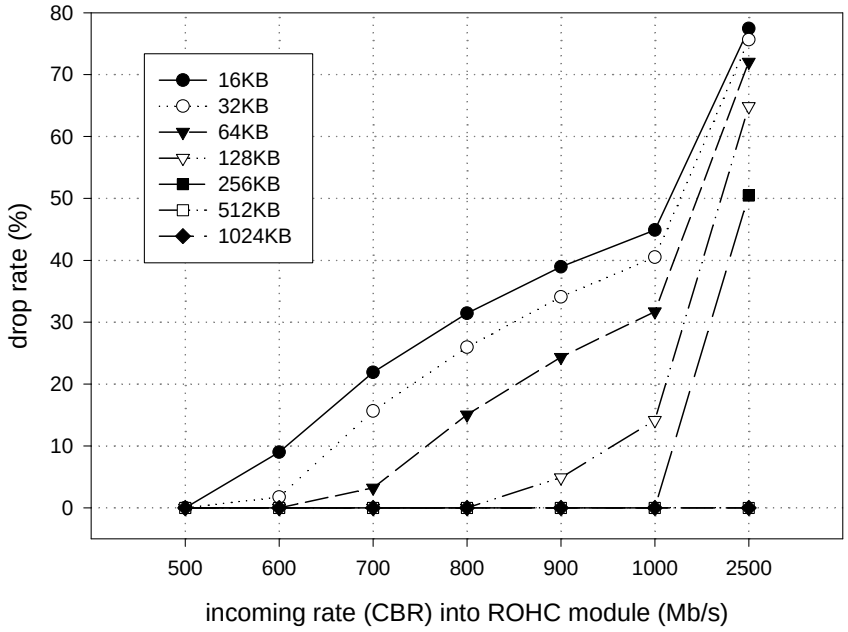


Figure 2-17: Drop rate (%) on encoding path of the hardware architecture. The dropping is caused by the lack of processing power against the large traffic volume injected into the hardware modules.

estimated processing time for encoding as 680 ns, and decoding as 556 ns.

In order to observe the performance of the hardware architecture installed into a single mesh router, we get the drop rate (%) at the parser cache on encoding path under various Constant Bit Rate (CBR) rate as shown in Figure 2-17. The drop rate implies the degree of processing speedup taking place when the built-in processor safely handle the incoming packets. In the figure, we plot the drop rate for various parser memory sizes. Until the 500 Mb/s arriving rate, the hardware tolerates for the packet processing. However, dropping starts at the point of 600 Mb/s in 16 and 32 kbytes parser memory size. At the extreme case of 2.5 Gb/s arriving rate, most of the packets are dropped within the hardware. This indicates ROHC hardware implementation to cut down the processing time must be carefully designed

to support the next generation wireless networks that have high traffic volume by adopting MIMO and high bandwidth radio links. However, the voice applications in 54Mb/s 802.11g do not generate such a high traffic volume, so that the packet dropping does not exist at the ROHC and packet aggregation hardware modules.

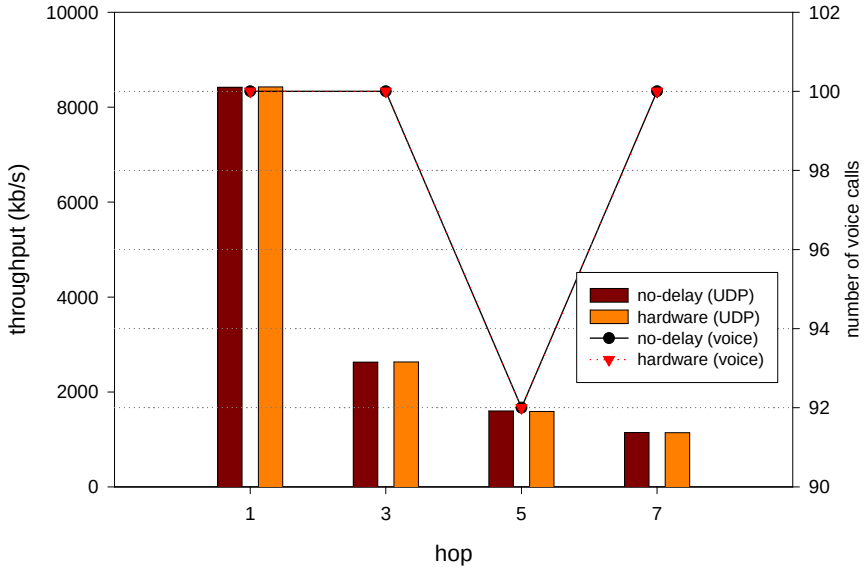
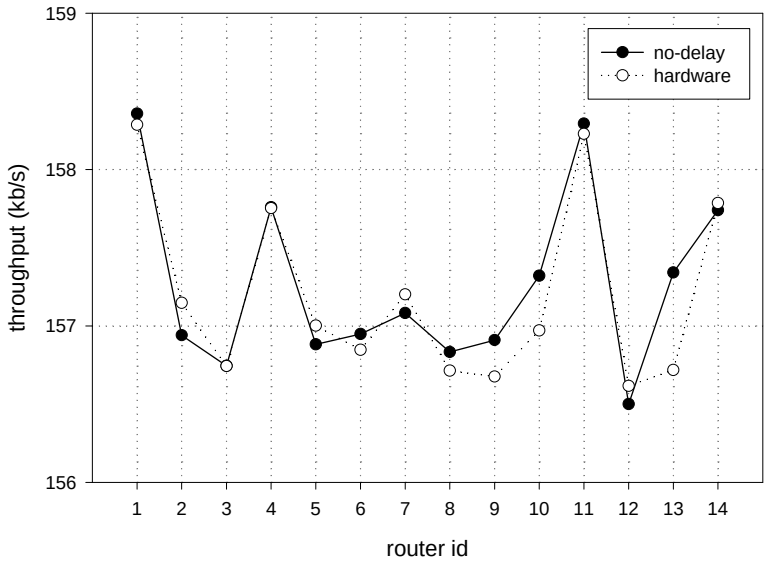


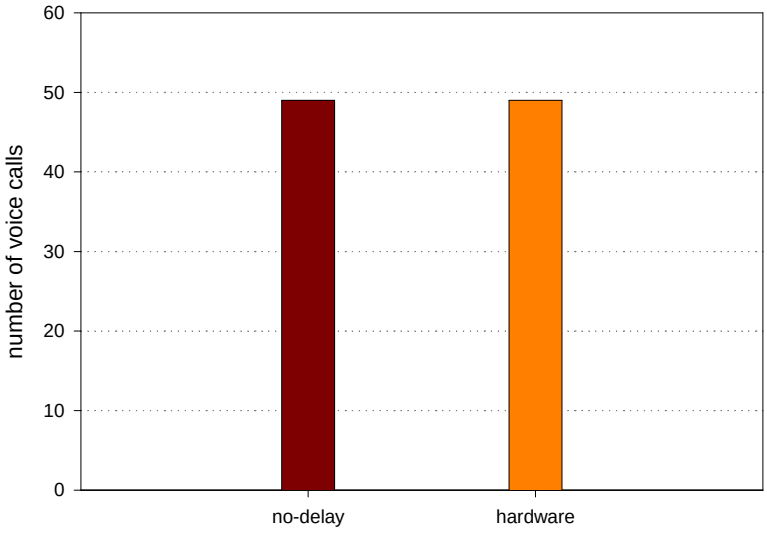
Figure 2-18: Sensor throughput and the number of voice calls supported by the hardware architecture: single data type under daisy-chain topology.

Figure 2-18 to 2-20 show the sensor throughput and the number of voice calls supported under various scenarios when the estimated processing times of the hardware architecture for encoding and decoding packets are applied to NS-2 simulations. We understand the proposed hardware design provides the similar trends as the no-delay case in all the scenarios by effectively speeding up the processing components within the ROHC and packet aggregation algorithms.

Figure 2-21 shows the R-factor distribution achieved from grid topology with multiple data types when the hardware architecture is adopted into the WMN. We observe the R-factor values from the hardware have almost the same pattern as the



(a) Sensor throughput



(b) Number of voice calls

Figure 2-19: Sensor throughput and the number of voice calls supported by the hardware architecture: multiple data type under grid topology.

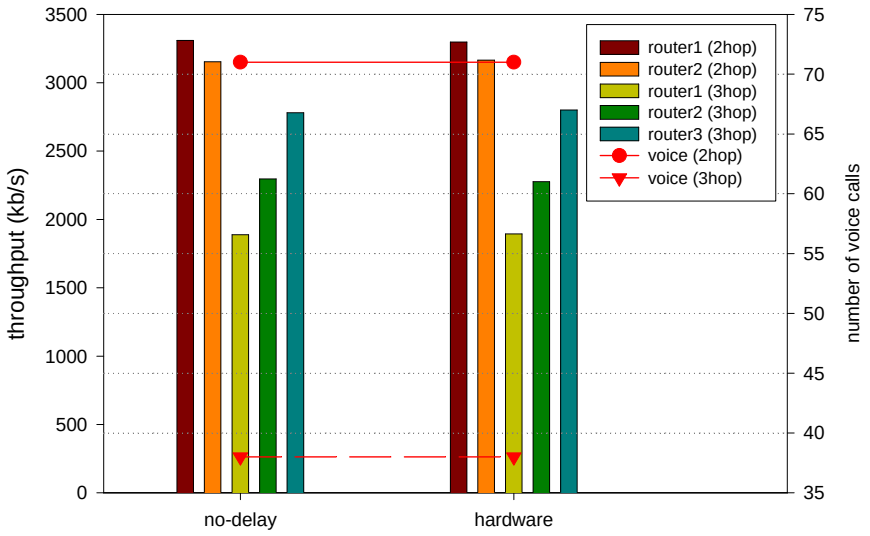


Figure 2-20: Sensor throughput and the number of voice calls supported by the hardware architecture: multiple data type under daisy-chain topology.

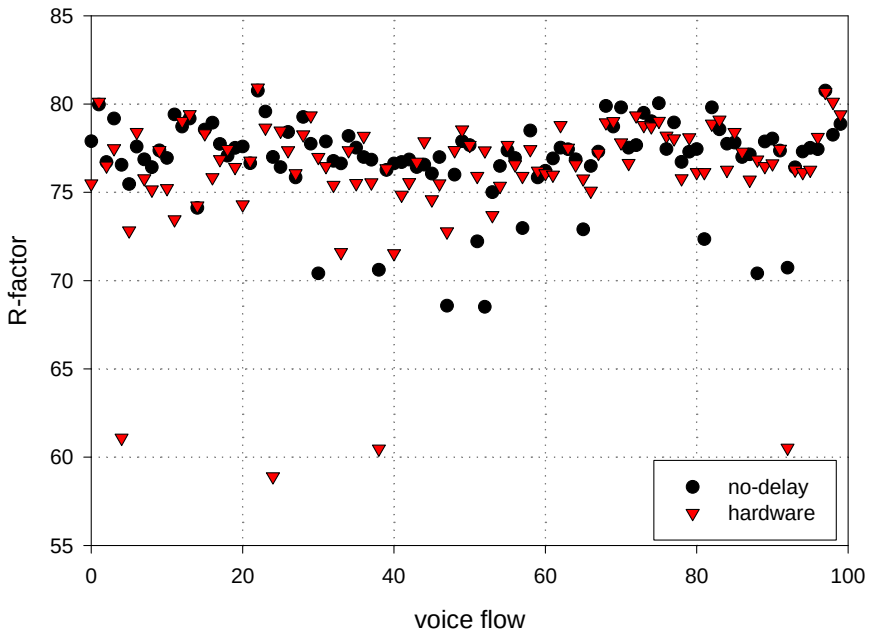


Figure 2-21: R-factor distribution appeared in the WMN equipped by the hardware architecture.

no-delay case regardless of a few low values around the 60 R-factor.

2.5 Network/Hardware Co-simulation Method

The network and hardware co-simulation is conducted by devising profile-based co-simulation method which merges network level simulator (NS-2) and hardware level simul

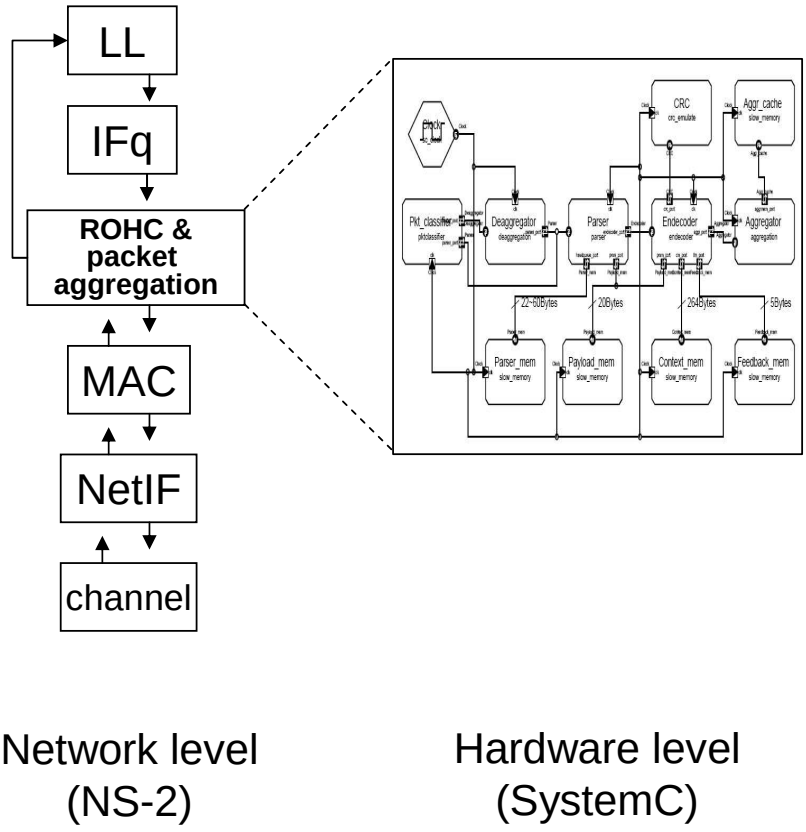


Figure 2-22: Network/hardware co-simulation model for ROHC and packet aggregation. The integration is done by NS-2 and SystemC.

Figure 2-22 shows the overall architecture of the proposed co-simulation method. Network layer hierarchy including ROHC and packet aggregation module is configured

by NS-2, and hardware level design is conducted by SystemC. In the co-simulation, a packet has to be manipulated with hardware level operations as well as network level operations when a packet from routing module or MAC enters ROHC and packet aggregation module.

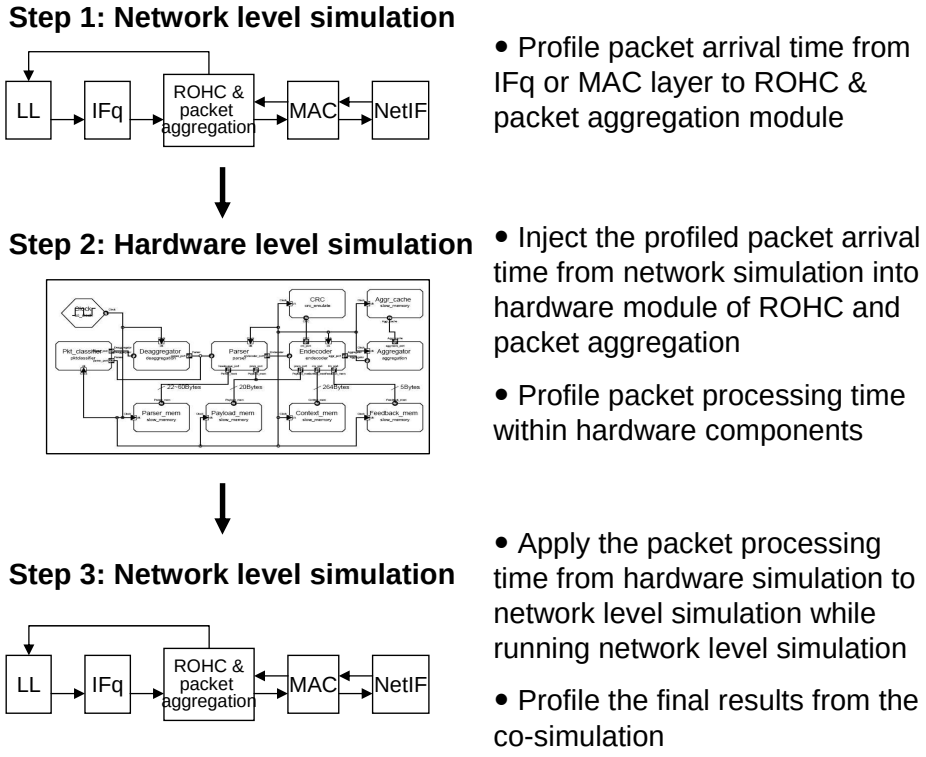


Figure 2-23: Procedure for performing network/hardware co-simulation.

For the cross-layer work, we use 3 step profiling procedures which are illustrated in Figure 2-23. Since we focus on reducing the processing overhead of the ROHC and packet aggregation functions, the profiling parameter becomes processing time in the co-simulation. In step 1, we first run NS-2 simulation, and profile packet arrival time from routing module (LL/IFq) or MAC layer to the ROHC and packet aggregation module. The profiled packet arriving time is injected into SystemC model in step 2,

which is done by packet classifier in the hardware architecture. According to inter-packet arrival time generated by NS-2, the packet classifier passes over the profiled packets to deaggregator if it is from MAC layer; otherwise, to parser. At the end of hardware level processing where a packet processed by all the hardware components comes out, hardware level processing delay is profiled again. Then, the processing time obtained from the hardware components recursively applies to NS-2 simulation in step 3; that is, when the ROHC and packet aggregation module receives packets from LL/IFq or MAC in network level simulator, it handles the packets according to processing delay created in step 2. After the third NS-2 simulation, we can profile the final NS-2 simulation results which include hardware module delay.

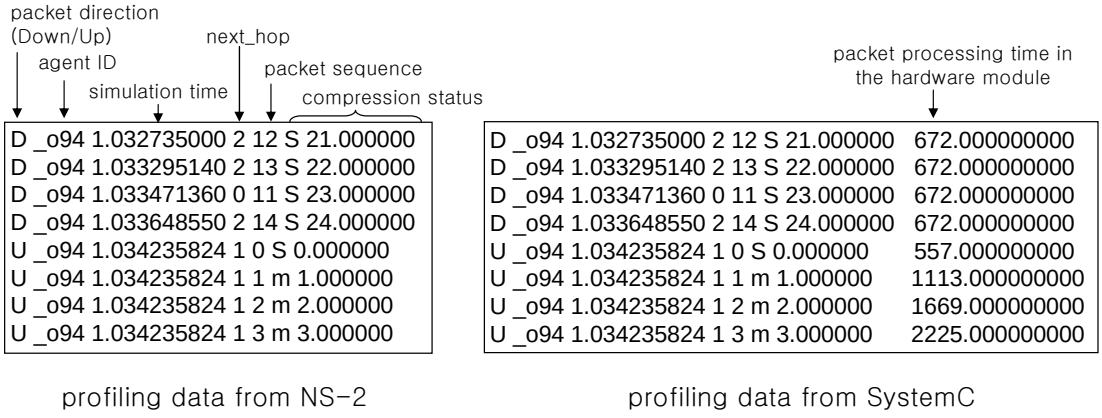


Figure 2-24: Trace file format created during the co-simulations.

Figure 2-24 shows the trace file format to profile the co-simulation parameters. The trace file from NS-2 contains the profiling data such as packet direction, agent ID, network level simulation time, etc. to represent the network level behaviors. Based on the NS-2 profiles, the SystemC creates the profiling data containing the packet processing time consumed in the hardware modules. The recorded hardware times

are subsequently read from NS-2 to apply the hardware delay into the network level simulation in step 3.

We envision this co-simulation method is applicable to other conventional algorithm design that needs to consider hardware capability, where the profiling parameter becomes power consumption, complexity, etc. For example, consider a power critical architecture design such as battery powered wireless networks. The design approach to this system could have two perspectives: single mesh router and network perspectives. If we want to add a new algorithm into a single router, we can profile the power consumption in a single router from hardware simulation and apply the profile to the network level simulations. Based on this profile information, the feasibility of the algorithm within a limited power capacity can be suggested.

We perform the co-simulation under 2 hop daisy-chain topology to get the performance difference between the application of the constant hardware delay in Section 2.4 and the network/hardware co-simulation method. Here, single channel interface instead of multiple channel is used, and we assume 8 Telos Motes generate UDP-sensor traffic within each mesh router radio range, and 100 voice calls are randomly positioned.

Figure 2-25 shows the hardware level processing time profiled at the end of step 2 in router 2. We can observe that the processing times for encoding and decoding scatter along the simulation. The encoding times are between 671 ns and 22839 ns, and decoding times reside between 557 ns and 1137 ns. The delay variations are different compared with the constant hardware delays obtained in the previous

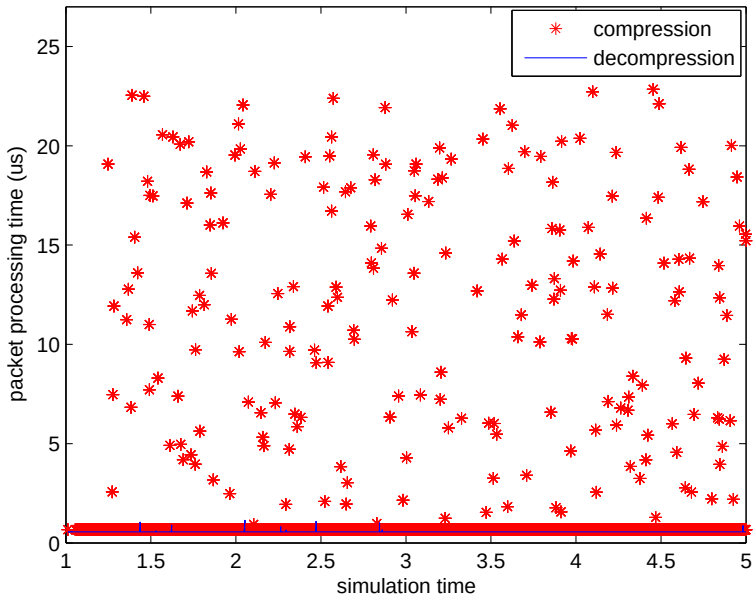


Figure 2-25: Distribution of packet processing time of ROHC and packet aggregation at router 2. Encoding and decoding times scatter 0.671 to 22.839 μs, and 0.557 to 1.137 μs, respectively

section, which is caused by the concurrent operations among PE1, PE2 and PE3. The reason why packet processing times for decoding (decompression) have narrow range over the encoding (compression) is that deaggregated packets are injected into the hardware modules by short interval, and the packets waiting for encoding have long inter-packet arrival times. This fact forces the packets for encoding to be preempted by the decoding packets. Therefore, the packets to be encoded wait for computation in memory relatively larger amount of time than packets to be decoded.

Figure 2-26 shows sensor throughput and the number of voice calls achieved from the co-simulation, in which we also plot the outputs from the application of constant hardware processing time. The two methods show different sensor throughput but the same number of voice calls. The co-simulation method supports more sensor

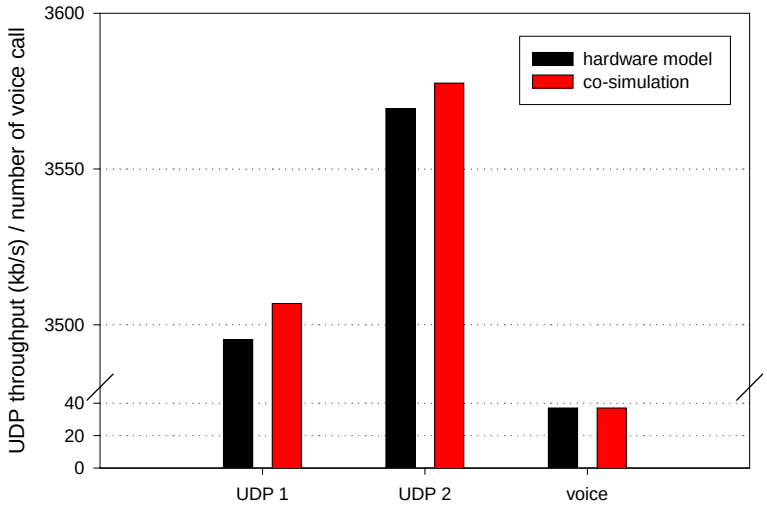


Figure 2-26: Sensor throughput and the number of voice calls obtained from the application of the constant hardware processing time, and network/hardware co-simulation method.

throughput since the concurrently operating hardware module consumes shorter time than the constant hardware processing time. Note the number of voice call supported in the mesh network in this scenario decreases from 100 to 37 and has no difference in the two simulation methods since greedy sensor traffic monopolizes mesh network resources. This co-simulation results tell us how overall WMN performance might be changed by the concurrent operations of hardwares installed into mesh routers.

2.6 Numerical Analysis of the Hardware Design

This section illustrates the proposed numerical analysis model suitable for predicting the processing delay of designed hardware architecture. Even if the target application in this section is the compression algorithm in wireless environment, we envision the numerical analysis procedure is applicable to other applications that need the

hardware assistance.

2.6.1 Queueing Model

Since our numerical model is derived from open Jackson queueing network [10], this subsection illustrates the average system size and system waiting time of: (1) general-arrival, exponential-service, single-server queue ($G/M/1$), (2) poisson bulk-arrival, exponential-service, single-server queue ($M^{[K]}/M/1$) where K is bulk size, (3) poisson-arrival, exponential-service, single-server queue ($M/M/1$), and (4) poisson-arrival, exponential-service queue with unlimited server capacity ($M/M/\infty$). In the queue systems, μ is a service rate, and λ is an arrival rate.

In $G/M/1$, there is no assumption on the arrival pattern except successive inter-arrival time is independent and identically distributed (IID). Let q_i be a steady-state arrival point probability, that is, the probability that an arriving customer finds n customers in the queue system. Then,

$$q_i = \sum_{j=1}^{\infty} q_j p_{ji}, \tag{2.1}$$

where $\sum_{i=1}^{\infty} q_i = 1$. Let $\mathbb{P}$ be the matrix form of the p_{ij}, then

$$\mathbb{P} = \begin{pmatrix} 1-x_0 & x_0 & 0 & 0 & 0 & ... \\ 1-\sum_{i=0}^{1} x_i & x_1 & x_0 & 0 & 0 & ... \\ 1-\sum_{i=0}^{2} x_i & x_2 & x_1 & x_0 & 0 & ... \\ \vdots & \vdots & \vdots & \vdots & \vdots & \end{pmatrix},$$

where x_i is an instance value of X_n that is the number of customers serviced during an inter-arrival time (T) of nth and $(n+1)$th customers, and

$$x_m = \Pr(X_n = m) = \int_0^\infty \frac{e^{-\mu t}\mu t)^m}{m!} dC(t), \tag{2.2}$$

where μ is the service rate, $C(t)$ is a cumulative distribution function (CDF) of T which is assumed to be IID random variable. From Equation (2.1), we can find $q_i = (q_{i-1}x_0 + q_i x_1 + q_{i+1}x_2 + ...) = 0, i \geq 1$. Denote $Aq_i = q_{i+1}$. Then, we have $q_{i-1}(A - x_0 - Ax_1 - A^2x_2 - A^3x_3 - ...) = 0$, from which we have $A = \sum_{m=0}^{\infty} x_m A^m$. Note the right term is simply a probability generating function of x_m, and denote it as $\beta(z)$. Then we have $z = \alpha(z) = L[\mu(1-z)]$, where $L(z)$ is a Laplace-Stieltjes Transform of CDF of T. Let r_0 be a single root value of the equation such that $0 < r_0 < 1$. Then, $q_n = (1-r_0)r_0^n,\ (n \geq 0, \lambda/\mu < 1)$. Consequently, average system size (L) and average waiting time (W) for $G/M/1$ are:

$$L = \frac{r_0}{1-r_0}, \quad W = \frac{1}{\mu(1-r_0)}. \tag{2.3}$$

When an arrival occurs according to poisson distribution but contains bulk customers with size of K, and single server's service rate follows exponential distribution, a system is designed by $M^{[K]}/M/1$. If K is geometrically distributed, the average system size is

$$L = \frac{\rho + rE[K^2]}{2(1-\rho)}, \tag{2.4}$$

where $\rho = \lambda E[K]/\mu$, and $E[X]$ means arithmetic average of X. In case K is a

constant value,

$$L = \frac{K+1}{2}\frac{\rho^{'}}{1-\rho^{'}} \quad (\rho^{'} = \lambda K/\mu). \tag{2.5}$$

The general $M/M/1$ queue system has single arrival with poisson distribution and exponentially distributed service time in a single server. Its average system size and waiting time are

$$L = \frac{\lambda}{\mu - \lambda}, \quad W = \frac{1}{\mu - \lambda}. \tag{2.6}$$

If a queue system has a server with unlimited capacity, infinite number of customers can be serviced at a time. This model is represented by $M/M/\infty$ which has

$$L = \frac{\lambda}{\mu}, \quad W = \frac{1}{\mu}. \tag{2.7}$$

2.6.2 Numerical Model Design

This section constructs a numerical model for the hardware architecture proposed in Section 2.4, and estimates average hardware processing delay from the numerical model. Since the model is composed of 7 components and each component is connected in series, we use open Jackson queueing network [10]. Each node of the network is modeled by $G/M/1$, $M^{[K]}/M/1$, $M/M/1$, or $M/M/\infty$. Figure 2-27 shows the numerical model for hardware model of ROHC and packet aggregation, where we model PE1 components by 4 nodes, PE2 by 3 nodes, and PE3 by 1 node, and service rate (μ) of each queue model follows the processing time from the hardware architecture. The figure indicates nodes on compression path by red line and decompression

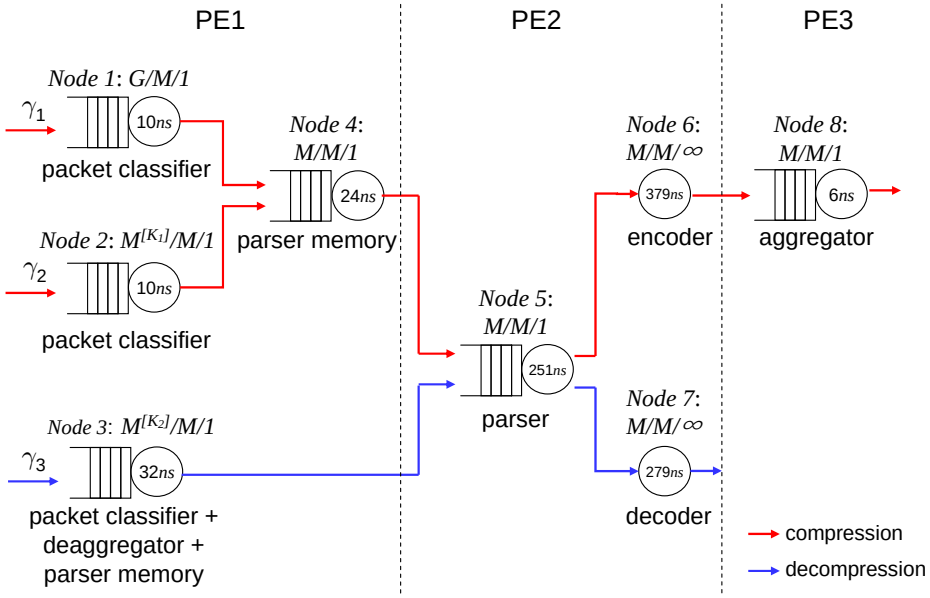


Figure 2-27: Numerical model of the hardware architecture for ROHC and packet aggregation, which is constructed by open Jackson queueing network. The arrival rate of each queue is modeled by general or poisson single/bulk arrival, and the service rate of *Node* i (μ_i) follows the processing delay of the hardware architecture in Figure 2-16.

path by blue line. General Jackson network assumes there is no limit in queue capacity, which establishes non-blocking system, for example, if a network is built up by a sequential two stations and customers at station 1 have to wait until customers in station 2 are completed, the queue network is called blocking system. Since the hardware speeds up the processing power such that there is no packet drop in the ROHC and packet aggregation module in which each components are concurrently executed, our numerical model is non-blocking queueing network.

Let μ_i be the service rate at *Node* i, and λ_i be the total average arrival rate from outside or other nodes to *Node* i. To satisfy flow balance at each node, λ_i in the

Jackson network is represented by

$$\lambda_i = \gamma_i + \sum_{j=1}^{m} \lambda_j r_{ji}, \tag{2.8}$$

where m is the number of nodes in the network, γ_i is arrival rate from outside of the network, r_{ij} is transition rate such that the packets finished at *Node i* go to *Node j* with probability of r_{ij}, where we have probability r_{i0} that a packet leaves the network at *Node i*. Equation (2.8) is rewritten as vector form like:

$$\overrightarrow{\boldsymbol{\lambda}} = \overrightarrow{\boldsymbol{\gamma}} + \overrightarrow{\boldsymbol{\lambda}}\mathbb{R}. \tag{2.9}$$

In our model, all the packets from previous node are totally injected into the next node with probability of 1 except *Node 5* in which we assume 50 % of the outgoing packets are on compressing path, and the rest are on decompressing path. Therefore, $\mathbb{R}$ in our model has following form.

$$\mathbb{R} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Since our model has at least one node with non-zero r_{i0} value and all the nodes have no absorbing states which have only arrival but no departure, Equation (2.9) can be solved as $\overrightarrow{\boldsymbol{\lambda}} = \overrightarrow{\boldsymbol{\gamma}}(\mathbb{I} - \mathbb{R})^{-1}$, where $\mathbb{I}$ is an 8×8 identity matrix. Let N_i be the random variable that represents the number of packets at *Node i* in steady state. Jackson has proved the joint probability function of N_i is

$$\begin{aligned} &\Pr(N_1 = n_1, N_2 = n_2, ..., N_m = n_m) \\ &= p_{n_1,n_2,...,n_m} \\ &= (1-\rho_1)\rho_1^{n_1}(1-\rho_2)\rho_2^{n_2}...(1-\rho_m)\rho_m^{n_m} \end{aligned} \quad , \tag{2.10}$$

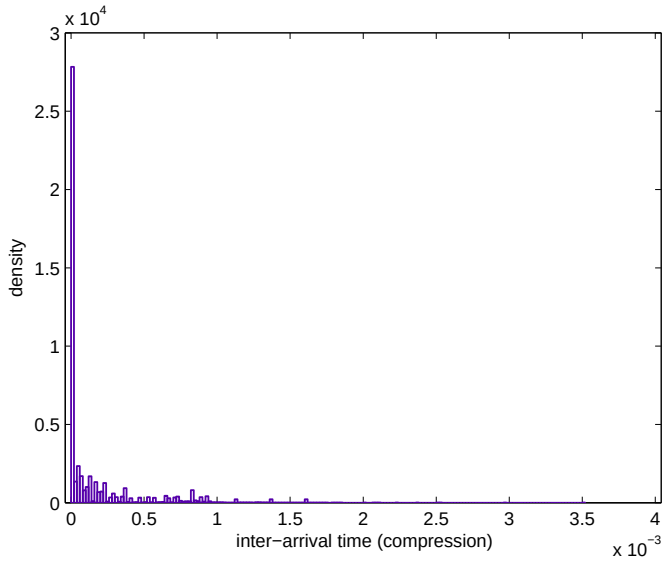
where $\rho_i = \lambda_i/\mu_i$. In our numerical model, individual node has average system size $L_i = \rho_i\ (1-\rho_i)$, and from little's formula, the average system waiting time $W_i = L_i/\lambda_i$. In order to estimate average compression and decompression processing delay within the hardware, we derive average system waiting time of a packet before

the packet departs from the queueing network. Let the average system waiting time of compression and decompression paths be W_c and W_d, respectively. Then, they are sum of waiting time of each queue system or globalized waiting time calculated by γ_i and little's formula. We define them as:

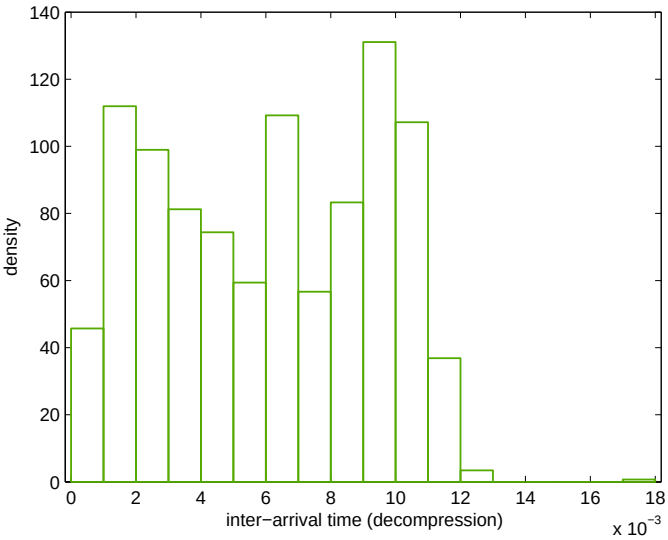
$$\begin{aligned} Wc &= W_1 + (L_2 + L_4 + L_5 + L_6 + L_8)/\gamma_2, \\ Wd &= (L_3 + L_5 + L_7)/\gamma_3, \end{aligned} \tag{2.11}$$

where $W_1 = 1/\mu_1(1 - r_0)$ for *Node 1* is calculated from Equation (2.3), and L_i $(i = 2, ..., 8)$ follows Equation (2.4) to Equation (2.7).

We differently model PE1 components for compression and decompression paths, which results from the inequivalent arrival traffic pattern on the two paths. Figure 2-28 shows the density function for inter-arrival time of packets to be compressed and decompressed, which is measured from NS-2 simulator under 40 aggregation level. Note there exist two arrival patterns on the compression path. The inter-arrival times for around 50% of compressing packets are near to zero, which means packets with bulk size K_1 simultaneously arrive whenever an arrival occurs. Actually, the bulk size measured from the simulation is different for each bulk arrival, so that we assume that K_1 is geometrically distributed random variable. The rest of the compressing traffic has arbitrarily distributed inter-arrival time. Therefore, we model the traffic pattern with bulk size K_1 as $M^{[K_1]}/M/1$ with $\mu_1 = 10\ ns$, the other one as $G/M/1$ with $\mu_2 = 10\ ns$. To cover parser memory write/read, we model the parser memory as a separate $M/M/1$ queue with $\mu_4 = 24\ ns$.



(a) Inter-arrival time of compressing packets.



(b) Inter-arrival time of decompressing packets.

Figure 2-28: Density function for the inter-arrival time of compressing and decompressing packets. Since the inter-arrival time of the compressing packets consists of two patterns, i.e., single or bulk arrival in (a), we represent it as general-arrival ($G/M/1$) or poisson-bulk-arrival ($M^{[K_1]}/M/1$). The arrival pattern of decompressing packets in (b) is modeled by poisson-arrival with bulk size K_2.

Let K_2 be the aggregation level. Then, a packet from MAC contains K_2 number of compressed packets. We can infer there exist bulk packets with size K_2 when an arrival happens on the decompression path. Therefore, we model *Node 3* as $M^{[K_2]}/M/1$ queue with $\mu_3 = 32\ ns$ which includes packet classifier time (10 ns), deaggregator time (10 ns), and parser memory write/read time (12 ns). The measured inter-arrival time of packets with bulk size $K_2 = 40$ on the decompression path is shown in Figure 2-28(b). The decompressing packet's arrival is similar to exponential distribution if we ignore some exceptional cases like 9 and 10 inter-arrival times.

In the Jackson network, the internal flow could not be poisson. Actually, if there is feedback from one node to any nodes, the internal flows can not be modeled by poisson distribution. However, fortunately, the Equation (2.11) holds regardless of the internal flow's traffic pattern, and a Jackson network behaves as if its nodes were independent poisson-based queueing system. From this consequence, we can model *Node 5*, *6*, *7*, and *8* by using $M/M/1$ and $M/M/\infty$ queue systems. The parser (*Node 5*) is designed by $M/M/1$ since it simply accepts compressing and decompressing packets from *Node 3* and *Node 4* and services the packets. The service time (μ_5) for *Node 5* is 251 ns. Since the parser does not release a packet until the encoder or decoder completes a compressing or decompressing, we use $M/M/\infty$ queue system for en_decoder. The two queue systems have $\mu_6 = 379\ ns$ and $\mu_7 = 279\ ns$, which include CRC and context memory access times. The aggregator uses aggregation cache, so that we make use of $M/M/1$ for the aggregator in which we include only cache access time, so that $\mu_8 = 6\ ns$.

2.6.3 Performance Evaluation

In this section, we evaluate the accuracy of the numerical model by comparing the numerical analysis results with simulation results from NS-2/SystemC co-simulations.

Since the queueing network accepts traffic from outside and the traffic determines the behavior of the network, we need to define arrival pattern and parametric values for entry points, i.e., *Node 1*, *2*, and *3* for our numerical model. To do this work, we analyze packet arrival time profiled from NS-2 simulations with respect to 10, 20, 30, and 40 aggregation levels. The simulation is performed under daisy-chain topology consisting of 3 mesh routers, and the data is measured at an intermediate router.

Let $\mathbb{G}$ be a matrix that contains γ_i $(i = 1, ..., 8)$ which is arrival rate at *Node i* from outside of the network. Then, we have

$$\mathbb{G} = \begin{pmatrix} 1539.4 & 990.6 & 653.4 & 0 & 0 & 0 & 0 & 0 \\ 2545.5 & 533.1 & 327 & 0 & 0 & 0 & 0 & 0 \\ 2838.9 & 376.3 & 218.1 & 0 & 0 & 0 & 0 & 0 \\ 2924.5 & 285.07 & 163.60 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

where the first row is for aggregation level 10, the second one is for 20, and so on. For *Node 1*, we get r_0 values of $G/M/1$ for 4 aggregation levels. Let $\overrightarrow{\boldsymbol{\rho}}$ be a vector including r_0's. Then, we obtain $\overrightarrow{\boldsymbol{\rho}} = (0.044, 0.022, 0.028, 0.0104)$.

We show parametric values related with K_1 and K_2 in Table 2.3. Since traffic to *Node 2* is assumed to be geometric, we measure $E(K_1)$ and $E(K_1^2)$ from simulations. We use half of the aggregation level for K_2 values of *Node 3* since the values give

Table 2.3: Arithmetic average values of K_1 and K_1^2 for *Node 2*, and K_2 values for *Node 3*.

Aggregation level	10	20	30	40
$E(K_1)$	5	7	9	12
$E(K_1^2)$	32	67	121	207
K_2	5	10	15	20

optimal results in the mathematical calculation of W_d. This adjustment of the values results from the assumption of exponential arrival for real traffic pattern from outside of *Node 3*.

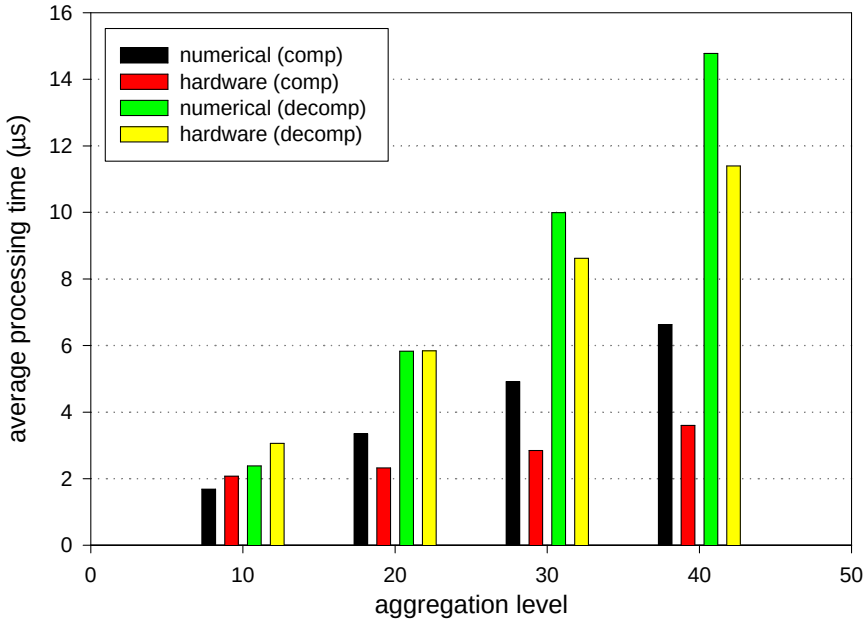


Figure 2-29: Hardware and numerical analysis results for ROHC and packet aggregation module. The average processing time of numerical model approaches to the time of hardware simulated from NS-2/SystemC co-simulation.

In order to validate the accuracy of the numerical model, we conduct extensive NS-2/SystemC co-simulation introduced in Section 2.5. In Figure 2-29, we plot average compression (comp) and decompression (decomp) processing time (μs) from hardware

and numerical models. The results from the hardware are derived from NS-2/SystemC co-simulation, in which we average hardware level processing delays on around 10^6 packets generated at the end of the co-simulations. We find out our numerical model closely approaches the simulation results at the aggregation level 10. Even if the difference between two models gets larger as the aggregation level increases, it is around 3 μs in worst case (40 aggregation level) which is consider to be small value.

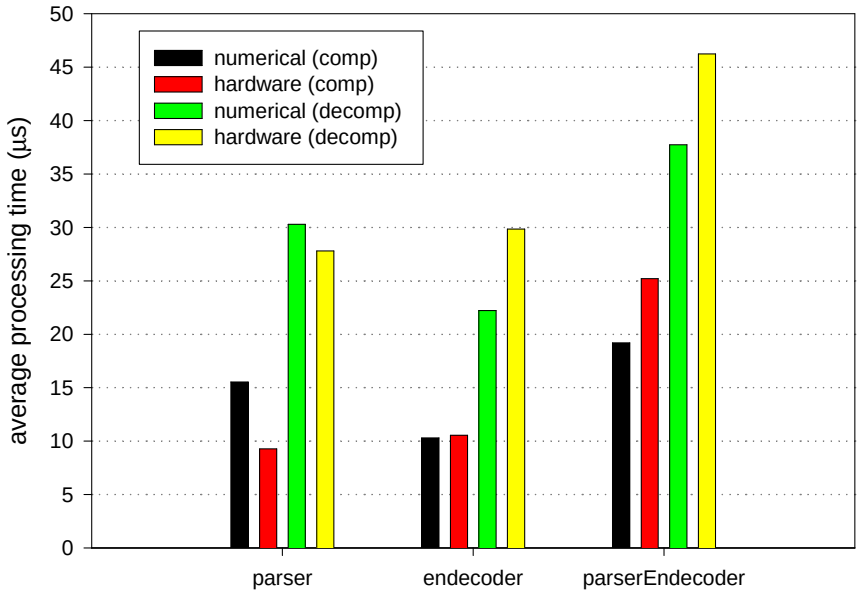


Figure 2-30: Average processing delay of ROHC and packet aggregation module for three cases: (1) only parser processing delay increases to 1000 ns, (2) the processing times of both encoder and decoder are incremented to 1000 ns, and (3) three components (parser, encoder, and decoder) slow down to 1000 ns.

To observe the accuracy of our numerical model when the processing delay of a hardware component is changed, we shows numerical and simulation results in Figure 2-30 under 40 aggregation level, where we provide three cases. In the first case denoted by parser, only parser's processing delay increases from 251 ns to 1000 ns. The second case (endecoder) means both encoder and decoder's processing delay

increases to 1000 ns. The case that processing delays of the three components (parser, encoder, decoder) extend to 1000 ns is denoted by parserEndecoder. We can observe the numerical model results closely match to the co-simulation results such that in the worst case, parserEndecoeer, the difference is around 8 μs.

Consequently, from the accuracy of our numerical model, we can answer following questions without any implementations or simulations; (1) how much processing time the proposed hardware architecture consumes under a specific aggregation level and (2) if we adjust the processing time of any hardware component such as parser or en_decoder, how much processing delay the hardware architecture shows. From this information, we can configure the hardware by adjusting each components' characteristics under the bottom line of a total hardware system delay.

Chapter 3

Node Algorithm Design and Optimization for Accurate Object Tracking in Multi-modal Tracking System

3.1 Introduction

In addition to sensor devices passively and single-functionally operating in general sensor networks, more complicated functions such as object tracking and reliable monitoring functions are added into the sensor nodes to construct a navigation and surveillance system.

For tracking methods, the authors in [49] propose the time-delay estimation meth-

ods that approximate location based on the time delay of arrival of signals at the receivers. On the other hand, direct tracking methods are proposed in [50] [51], in which the frequency-averaged output power of a steered beamformer are used. These two traditional tracking methods have the drawbacks under the reverberant indoor environment which frequently generates extraordinary signals. In order to overcome this problem, Particle Filtering(PF)-based state-space approaches are proposed in [5, 52–54]. The PF method considers to be a powerful methodology for nonlinear and non-Gaussian signal processing problems [5, 55–59]. However, in many PF-based tracking systems, initial state is not clear, so that PF tracking system may have lost the target object even with a known dynamic model. Moreover, incorrect dynamic model and corrupted observation lead to continuous wrong estimation of the object trajectory which is called trajectory divergence problem. Other than the PF-based models, the tracking systems with visual cameras are also investigated in [60–62]. In these research efforts, multiple cameras are used to extract the real position information of the target objects.

If multiple types of sensor data are combined, the weakness of each type of sensor can be compensated by other types and also they can assist each other to have better measurement. Moreover, they are more adaptive and robust in diverse environment as a specific sensor can be less sensitive to a certain condition of environment.

The authors in [63] have proposed PF-based tracking architecture for the multimodal sensor fusion to track people in video-conference application. They use video data for the main modality and the audio signal for a complementary operation.

However, the video image processing requires high processing complexity, so that in real-timing environment with high volume of images, the method may have processing overhead that can result in the degradation of tracking accuracy. In order to reduce the computational complexity, we have developed a new tracking system model in which the low computational acoustic-based PF primarily tracks the objects and two visual sensors solve the unclear initial state and trajectory divergence problems inherent in the PF algorithm [3]. However, the previous work assumes that both visual sensors capture the tracking space with no time difference, the PF and visual algorithms activate as soon as the acoustic and visual sensors are sampling object information, and the visual compensation results are immediately applying to the next PF state generation. However, the assumptions are not applicable to real situations since the sampling and calculation points generally locate in different places, and the non-synchronized sampling and data arrival take place in the middle of tracking. This chapter addresses a *network synchronization problem* caused by the absence of the aforementioned assumptions. After both visual sensors capture the tracking space independently, they need to send the images to a processing Server. At the different time, the PF estimates from acoustic sensors also arrive at the Server with different transmission delay. The Server should determine whether the visual compensation process needs to be performed based on the acoustic and visual sampling times. If the Server generates the compensated position estimation based on PF estimates and visual images, it sends feedback data to acoustic sensor to correct the possible estimation errors in the PF calculation. In order to model the developed tracking

system, we configure a distributed wireless tracking network in which Routers have a role in backbone nodes and acoustic sensors are sampling object information under the communication range of the Routers. Both visual sensors are located at appropriate positions to efficiently capture the tracking space with sufficiently different angles. When the Routers deliver the acoustic data, they use the proposed time-based packet aggregation algorithm. In the algorithm, a Router checks whether the sampling time of the packet is the most recent one after it receives a packet from an acoustic sensor. If it is true, the packet is saved for future aggregation operation. After the Router receives new sampling times from all the acoustic sensors, it aggregates all the packets and sends the aggregated packet to a Server. Based on the aggregation algorithm, the PF estimates from acoustic sensor are efficiently delivered to a Server with removing unnecessary network resources consumed to deliver a number of acoustic data.

From the simulation study, we show the increased tracking accuracy from joint operation of multiple sensor types severely deteriorates when acoustic sensors use short sampling interval, and non-sensor traffic volume flowing the wireless tracking network increases. For the possible solutions of the performance degradation, we propose a traffic differentiation model. The basic idea of the model is that we can solve the skewed transmission delay among the sensor traffic and non-sensor traffic by adopting different network queues and Weighted Round Robin (WRR) scheduling mechanism at Routers. The weight allocated to each queue is obtained by a proposed Delay-based Weight Allocation (DWA) algorithm. In the differentiation model, we first get the transmission delay from multi-modal sensors to a Server. Based on the

ratio of the obtained delay, we allocate normalized scheduling weight to each network queue. From the simulations, we show the differentiation model can mitigate the *network synchronization problem*, so that the tracking system provides the sustainable support of visual sensors to correct the PF estimation errors.

To represent the *success* or *fail* of a visual compensation by a numeric value, we define a new performance metric, Successful Compensate Rate (SCR) which is applicable to the tracking system. The SCR is the ratio of the number of PF-error correction assisted by visual sensors over total number of PF-estimate generation from acoustic sampling. The SCR has a role in gauging the accuracy of the tracking task.

Additionally, we propose an algorithm, Statistical Estimation Algorithm (SEA) to approximate the number of *success* to appear in the multi-modal system. In the observation of the tracking system, we identify the SCR of the visual compensation depends on key factors: the sampling interval of acoustic sensors and the transmission delay of multi-modal sensor data. Based on this aspect, the SEA approximates the SCR by using the two key factors as the algorithm parameters.

For the algorithm formulation, we first observe the transmission delay between the multi-modal sensors and the Server. The observation results reveal that the delay from visual sensors to the Server is modeled by Gaussian Probability Density Function (PDF), and delay from Server to acoustic sensor follows Exponential PDF. From the PDFs, the SEA generates random values to imitate the delay in real situation. Then, it uses the random values and pre-determined acoustic sampling interval as

parameters, and determines the next status according to transition rules that have four modes (R, L, B, and U) and two states (s and f). The transition rules have restrictions that only state-to-mode transition is allowed and there are no mode-to-mode, state-to-state, and mode-to-state transitions. After establishing a tracking system, it is necessary to maintain the system at a given level of tracking accuracy. This objective can be achieved by controlling the acoustic sampling interval or the transmission delays since the tracking accuracy depends on the *success* of visual compensation. Thus, we propose another algorithm, Statistical and Estimation and Adaptation Algorithm (SEA^2). The algorithm controls the acoustic sampling interval to adapt the level of tracking accuracy since the parameter is simple to adjust and mainly affects the *success* of the visual compensation. In the parameter adjustment, SEA^2 uses the SEA algorithm to obtain an initial sampling interval (Δt_s^0) and consists of Phase 1 and Phase 2. Since the SEA requires the mean and standard deviation for the PDFs, the Phase 1 perform the estimation of the population parameters. Based on the estimation, the SEA is executed to obtain Δt_s^0. In Phase 2, the algorithm uses exponential increase or decrease of Δt_s^0 to fast reach the target tracking accuracy level.

The validation of both algorithms is performed by NS-2 [2] simulations by constructing daisy-chain and tree scenarios. We verify the algorithmic accuracy of the SEA by showing the mathematical calculation in the SEA properly approximates the simulated SCR results. For SEA^2 validation, we observe how the SEA^2 automatically adjusts acoustic sensors' sampling interval to achieve a target SCR. The simulation

results show the sampling interval adjustment is well performed when we set up the target SCR to 90%. In order to observe the real-timing adaptation capability of the SEA^2, we change the target SCR in the simulations from 30% to 60 and 90%. The results indicate SEA^2 also has a confidential adaptation mechanism in the real-timing case.

The research works in this chapter are based on our previous research outcomes from [4] [64].

3.2 Background and Problem Definition

3.2.1 Tracking by Particle Filter

Particle Filtering (PF) [5] is a powerful method for sequential signal processing for nonlinear and non-Gaussian problems. It is broadly used in applications that need the tracking and detection of random signals. The algorithm is also based on its operations on representing relevant densities by discrete random measures composed of particles and weights, and computes integrals by Monte Carlo methods [57]. In the tracking problem based on PF, the measuring outputs from an acoustic sensor are bearings or angles ($\mathbf{Z}_t$) on the grid along the perpendicular coordinates at time t. Based on the angle information, we can get the estimated position ($\tilde{x}_t$, $\tilde{y}_t$) and velocity ($\tilde{v}_{x,t}$, $\tilde{v}_{y,t}$) in the cartesian coordinate system. At the next acoustic sampling time, $t+1$, we can get the next angle ($\mathbf{Z}_{t+1}$) and corresponding outputs based on the previous PF estimates. In this tracking method, we obtain more accurate position

estimates as we increase the number of particles.

Tracking Problems

In PF-based tracking applications, there are two key problems preventing accurate tracking process. The first one is the initial state problem where initial state may not be reliable and sometimes is not existing. For example, in the beginning of the tracking or when the signal from an object re-appears after silence movement or blocking obstacle, we can consider the cases as the initial state problem. Since the PF application assumes the initial state is clearly given, the PF approximation outputs will show significant deviation from the real object trajectory in the presence of the initial state problem. The trajectory divergence problem is another key problem that appears in many PF applications. The object dynamic model could change in the middle of tracking even with the given initial state. The change with or without the initial state results in the tracking to be diverged. Since the next PF estimation is based on the current state vector, the deviated state vector in current time will lead to further erroneous tracking in the wrong direction.

Possible Solutions

The aforementioned tracking problems can be solved by multiple dynamic model [65,66], multiple acoustic sensor detection [67], and audio-visual multi-modal tracking algorithm [63]. Especially, the last multi-modal algorithm has recently been active research domain due to its accuracy and fast implementation. In [63], the visual sensor mainly tracks the object and an acoustic sensor supports the tracking when

the object disappears from the visual space. However, in realtime point of view, the complexity of image processing becomes an overhead factor. Therefore, our approach in this chapter is to adopt a low computing acoustic sensor for the main tracking device, and the visual sensor compensates the tracking deviation caused by acoustic-based PF outcomes.

3.2.2 Tracking by Visual Sensor

Visual localization algorithm is performed to extract the object position estimation from captured visual image. It is based on the parallel projection model [68], which simply approximates the position with a known reference point, $P_r(x_r, y_r)$. Arbitrary point on a camera or an estimate obtained by PF algorithm could be the reference point. The localization algorithm assumes both cameras can capture the target object at the same time. In the algorithm, the reference point is projected on the viewable planes of both cameras, and the object points appeared in the camera sensing planes are also projected on the viewable planes. Let the projected point of reference point be $P_v^i = (x_v^i, y_v^i)$ and the projected point of sensing plane be $P_s^i = (x_s^i, y_s^i)$, where i is the camera id and takes 1 and 2. Then, we can obtain the distance Δd_i between the projected points as $\Delta d_i = \overline{P_v^i P_s^i}$. If we assume the each viewable plane of camera 1 and 2 forms x and y cartesian coordinate respectively, the estimated object position is obtained by $P_e(x_e, y_e) = (x_r \pm \Delta d_1, y_r \pm \Delta d_2)$.

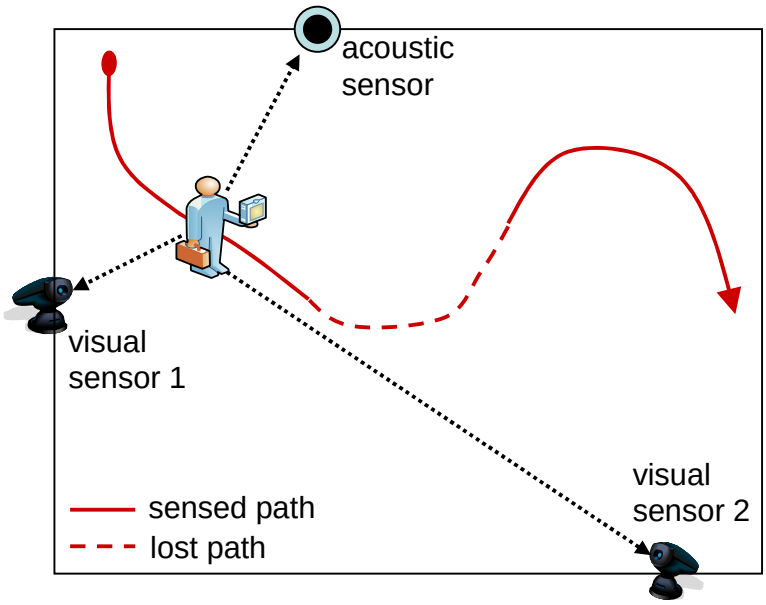


Figure 3-1: Target application model for the object tracking. It consists of an acoustic sensor and two visual sensors to capture the object information. The dashed line means the lost of the acoustic signal in the middle of object moving.

3.2.3 Target Application Model

The target application model in our approach is based on the integration of an acoustic sensor and two visual sensors as shown in Figure 3-1. A three dimensional acoustic localizer is located at an acoustic sensor to get the direction of arrival (DOA) [69]. The localizer detects two angle components (azimuth angle θ, elevation angle ϕ) from the arrival time difference between embedded adjacent microphones. The PF associated with acoustic sensor mainly obtains the coordinate information of the moving object, and supportive tasks such as position initialization, detecting of silent movement, and compensation of the deviated tracking from acoustic signal are done by two visual sensors. The visual sensors require to be located in appropriate positions to capture the object with sufficient angles which are used in the localization algorithm in Section

3.2.2.

3.2.4 Visual Compensation Effect

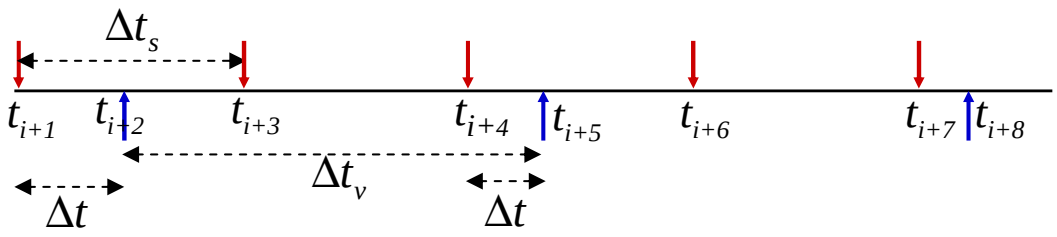
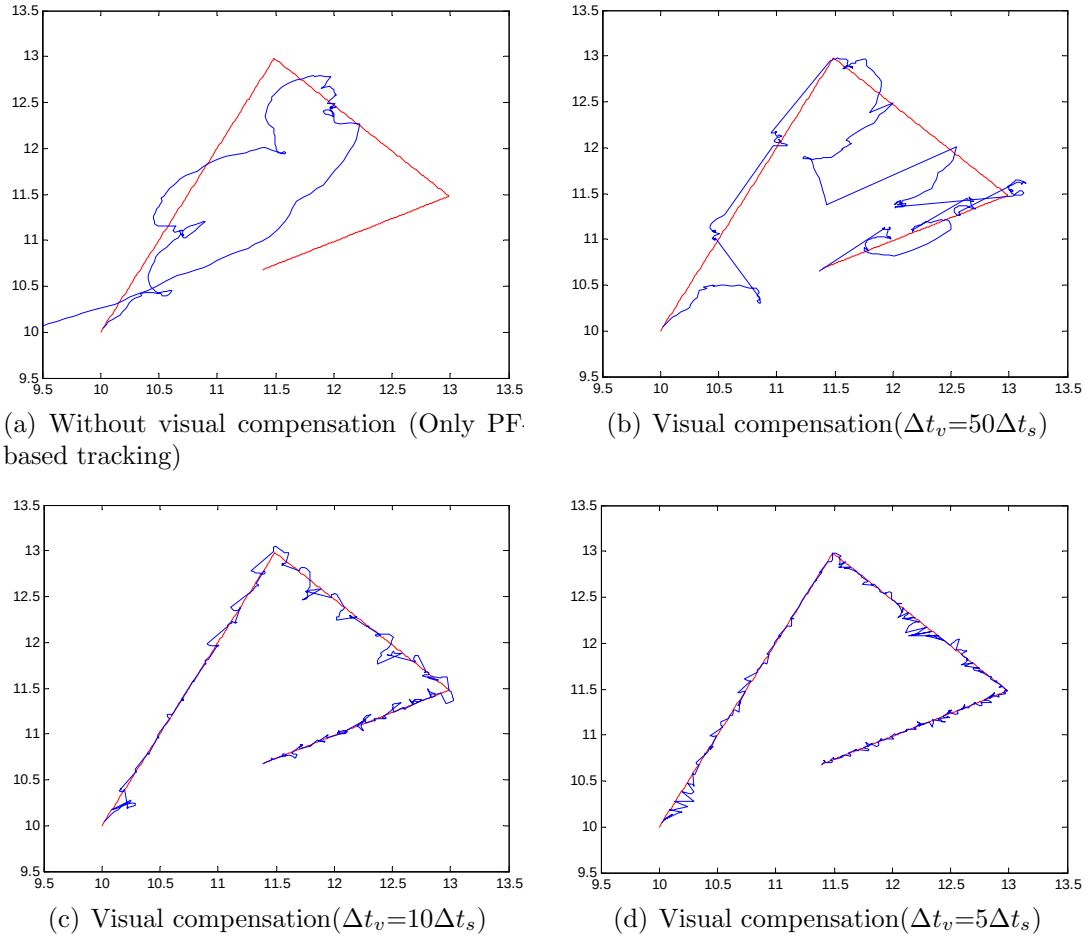


Figure 3-2: Sampling time sequence of an acoustic sensor and visual sensors. Red arrow is the acoustic sampling time and the blue is the sampling point of the visual sensors.

In the visual compensation process, the operation of visual sensors are independent of the acoustic sensor's operation. Figure 3-2 shows an example of sampling time sequence possibly occurring in the application model. The acoustic sensor samples the object signal by Δt_s interval and the visual sensors capture the tracking space every Δt_v. In this case, both sampling tasks have the time difference Δt. If the Δt is in [0, Δt_s], the visual localization algorithm can be successfully performed to correct the tracking error in the PF-estimate. Here, the visual localization algorithm uses the PF estimate for the reference point. For example, if we consider the compensation at time t_{i+5}, acoustic estimate at t_{i+4} becomes the reference point.

We observe the advantage of the visual compensation assisting the PF-based tracking system in the remainder of this section. The advantage appears when the multimodal sensors independently operate like Figure 3-2. For the observation, we use non-linear model with semi-triangular movement where an acoustic sensor is placed

at (0,0) of a cartesian coordinate.



(a) Without visual compensation (Only PF-based tracking)

(b) Visual compensation(Δt_v=50Δt_s)

(c) Visual compensation(Δt_v=10Δt_s)

(d) Visual compensation(Δt_v=5Δt_s)

Figure 3-3: Tracking accuracy when the two visual sensors assist the PF-based tracking task. Red line represents the real object movement, and the blue line is the trajectory estimate obtained by associating an acoustic sensor with two visual sensors.

Figure 3-3 shows the simulation results for the various sampling interval of visual sensors. The Δt_v takes 50, 10, or 5 times longer than the sampling interval of acoustic sensors. The target object is moving along the red line, and the trajectory estimate is indicated by blue line. In Figure 3-3(a), only PF-based estimate has large deviation from real object movement. However, when we associate two visual sensors with an acoustic sensor creating the PF estimate, we can increase the tracking accuracy

in proportion to the visual sampling frequency. When Δt_v=50Δt_s, the trajectory estimate roughly follows the object movement with large variation as shown in Figure 3-3(b). As we increase the visual sampling frequency to 10Δt_s, the trajectory estimate is almost the same as the object movement as shown in Figure 3-3(c). In more visual sampling frequency like Δt_v=5Δt_s, the trajectory estimate becomes more accurate as shown in Figure 3-3(d). Note the processing overhead of PF is a few microseconds as indicated in [6], and the localization algorithm for visual compensation is performed rarely compared with the PF calculation. Therefore, our application model can minimize the overall processing overhead for the tracking task as well as provide the accurate tracking task.

3.2.5 Network Synchronization Problem in the Application Model

Even if we have mentioned the visual compensation provides significant improvement in tracking accuracy in previous section, it requires some assumptions: (1) two visual sensors capture the tracking space with no time difference, (2) as soon as acoustic and visual sensors are sampling object information, the PF and visual localization algorithms should calculate the position estimation without time delay, and (3) the visual compensation results are immediately applying to the next PF state generation. However, the sampling and the calculation points generally locate in different places, so that there exists a synchronization problem caused by the data transmission delay in the tracking system.

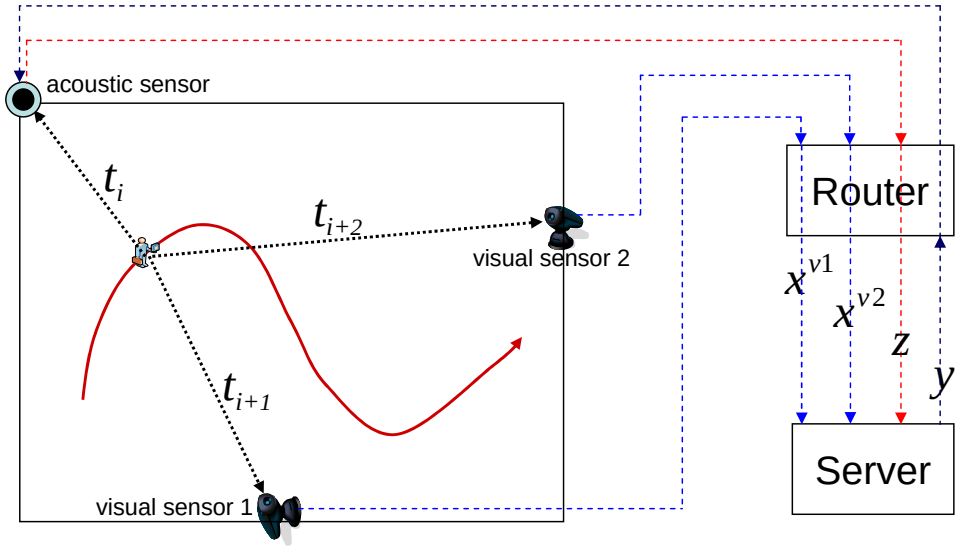


Figure 3-4: Delay factors causing the *network synchronization problem* in the tracking model.

Figure 3-4 indicates the factors to be considered in the tracking model at the network point of view. At t_i, an acoustic sensor receives the object signal, and the visual sensor 1 and 2 take the image at time t_{i+1} and t_{i+2}, respectively. Based on concepts in the previous section, $t_{i+1} = t_i + \Delta t_1$ and $t_{i+2} = t_i + \Delta t_2$, where Δt_1 and Δt_2 are sampling time difference between acoustic and visual sensor 1 and 2. If we assume the PF calculation is done at acoustic sensor, and the visual localization is done at a remote computing machine, namely, Server, the PF-based position estimate and the visually sampled data need to be sent to the Server via network Routers. In this situation, the image frames taken by visual sensor 1 and 2 arrive at the Server after x^{v1} and x^{v2} delays, and the PF estimate requires transmission delay z to arrives at the Server. Additionally, the visual compensation estimate at the Server needs to be re-sent to the acoustic sensor with delay y for the adjustment of the next PF calculation. We define a tracking problem caused by the network transmission delay

in visual compensation process as *network synchronization problem*. The independent data transmission delay in addition to sampling time difference among multi-modal sensors causes the *network synchronization problem*, so that we address how to tackle the problem in the remainder of this chapter.

3.3 Network Synchronization for Object Tracking

3.3.1 Configuration of Wireless Tracking Network

In order to support the tracking task, we consider a wireless tracking network connecting the multi-modal sensors and a Server. This is a type of distributed wireless network since the algorithm processing points are distributed in the network to support the tracking task.

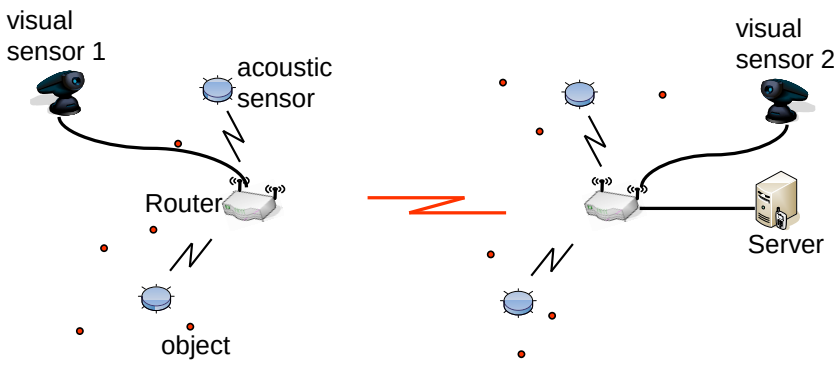


Figure 3-5: An example of the wireless tracking network.

Figure 3-5 shows an expected configuration of the tracking network. Routers are communicating each other by wireless channel and the last mile Router is connected to a Server. More than one acoustic sensor sample and send the object information

to the Router. Two visual sensors are connected to Routers and independently send the visual image to the Server. The PF calculation is done at the acoustic sensor and the visual localization algorithm having more complexity is performed at the Server. The localization algorithm could be performed at the Routers. However, as we have indicated in [70], the fully distributed tracking architecture has large end-to-end transmission delay of the visual image since the visual sensors have to send the same image to all the Routers, which causes heavy traffic in the network. Note the image size from a visual sensor is relatively larger than the packet from acoustic sensor. For example, when we capture the visual space by IP camera [71], the size of image frame is within the range of 30 to 55 KBytes. Therefore, we adopt the server-based architecture to reduce the duplicate transmission of the same image as well as to use the high computational power of the Server.

3.3.2 Time-based Packet Aggregation of Acoustic Sensor Data

The first problem to be solved for the network synchronization is how to deliver to the Server the object information from more than one acoustic sensor in a timely manner. For this problem, we propose a time-based packet aggregation algorithm as described in Algorithm 1. Whenever a packet from an acoustic sensor is coming to the Router, the Router first checks the sampling time (t_i) of the packet is the most recent one. The received packet is inserted into a Queue until the Router receives packets having the most recent sampling time from all the acoustic sensors. If t_i of the packet is older than the previously saved sampling time (T_i), the packet is dropped. If the

Algorithm 1: Time-based packet aggregation algorithm to deliver the data packets from more than one acoustic sensor.

P_i: a packet currently arriving at a Router. It is originated from acoustic sensor i.
Q_i: a packet being queued in a *Queue*. It belongs to acoustic sensor i.
P_a: an aggregated packet to be sent to the next hop Router.
n_a: the number of acoustic sensors in the range of the Router.
$P_a = \{\varnothing\},\ T_i = 0,\ i$=1...$n_a$

```
// Check the sampling time of the incoming packet is the latest
one
```

c=0
for *i=1* **to** n_a **do**
  t_i = sampling time of P_i
  if $T_i < t_i$ **then**
    $c = c$+1
  else drop(P_i)
end

```
// Make an aggregated packet based on sampling time
```

if *c is equal to* n_a **then**
  for *i=1* **to** n_a **do**
    dequeue(Q_i)
    t_i = sampling time of Q_i
    $T_i = t_i$
    $P_a = \{P_a \cup Q_i\}$
  end
  send(P_a)
  $P_a = \{\varnothing\}$
else
  enqueue(P_i)
end

Router receives the packets with the latest time, it makes an aggregated packet (P_a) and sends it to the next hop Router. At this point, we need to save the sampling time of the dequeued packet ($T_i = t_i$) for the next comparison of the sampling times. We assume that the sampling point of the acoustic sensors are same, which can be realized by regularly sending SYN packets from the Router to acoustic sensors to adjust the distorted sampling point. By using the aggregation algorithm, we can

reduce the network traffic and end-to-end delay of the acoustic sensor data as well as simplify the visual compensation process since the acoustic data can arrive at the Server at the same time. Since the sampling interval at acoustic sensors relatively larger than the transmission delay between acoustic sensors and a Router, we can ignore the impact of waiting time to aggregate packets from acoustic sensors.

3.3.3 Visual Compensation Considering Network Synchronization

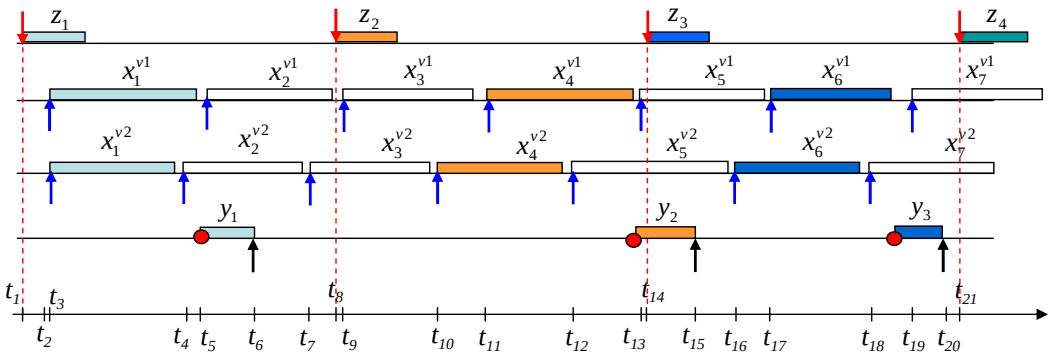


Figure 3-6: Packet flowing example appearing in the tracking model.

The next problem in the tracking task is to identify visual compensation in network synchronization point of view. To clarify the point, we show a packet flowing example in Figure 3-6 possibly appearing in the system. This traffic pattern could happen in a situation that an object's signal frequently disappears, so that visual image is sent as many as possible to detect the object trajectory. Similar to Figure 3-2, the acoustic sampling times are denoted by red arrows and blue arrows are for visual image capturing points. We additionally add red points to represent the calculation

of the visual localization algorithm at the Server side. Since the visual compensation estimates are re-sent to the acoustic sensor for the next PF adjustment, we add the black arrows to represent the feedback arrival at the acoustic sensor. The visual sensor 1 and 2 send the captured image frame to the Server with delay x_i^{v1} and x_i^{v2}, respectively. When a Router is ready to send an aggregated acoustic packet to a Server, it also sends the packet with transmission delay z_k. After finishing visual localization algorithm, the Server sends the compensated PF-estimate to the acoustic sensor. This feedback transmission takes y_j delay. Here, i,j, and k represent the generation sequence of the multi-modal data. Note that the delay size of multi-modal sensor data is different in times since the tracking network has a number of delay factors like router capacity and background traffic volume. In other words, the transmission delay takes randomness property. If we compare this example with Figure 3-2, the visual compensation process needs to be differently interpreted. For example, the t_9 in Figure 3-6 could be a successful visual sampling time under the illustration of Figure 3-2 since it is within t_8 to t_{14}, which means Δt is within the [0, Δt_s]. However, in this example, the sampling time t_7 of the second visual sensor is not between t_8 and t_{14} due to the different image capturing point of visual sensors. This indicates the first assumption in Section 3.2.5 does not valid in real situation. Therefore, the image from second visual sensor may give the wrong information to the estimation procedure. In this case, we would better use visual images captured at t_{10} and t_{11} since they more precisely contain the tracking space between t_8 and t_{14}. Even if the visual images seem to capture the tracking space in timely manner,

they do not provide good information with the visual compensation process when we consider the transmission delay (y_2) of feedback data that reaches the acoustic sensor at t_{15}. The arriving point is between t_{14} and t_{21} that is a new acoustic sampling period. For y_2 delay time, the object can have abrupt moving behavior in which case the information at t_{15} can also give the negative information to the PF estimation. Fortunately, we have another feedback arrival at t_{20}, and the feedback gives a right information to the next PF calculation at t_{21}. This complicated situation takes place since the second and third assumptions in Section 3.2.5 are not applicable to the real environment.

3.3.4 Definition of Success and Fail Conditions

To clearly define when the visual localization algorithm can be executed and what is the success in the visual compensation process, we make two conditions as follows.

- *Condition 1*: The Server sees that the sampling times of both visual sensors are later than the acoustic sampling time of previously arrived acoustic data. At this point, the Server performs the localization algorithm.

- *Condition 2*: The compensated estimate should be feedbacked to acoustic sensors before the next acoustic sampling time.

If above conditions are satisfied at the same time, we define this case as *success* in the visual compensation process. We regard other cases not following above conditions as *fail* even if it actually does not mean the failure of the algorithm calculation.

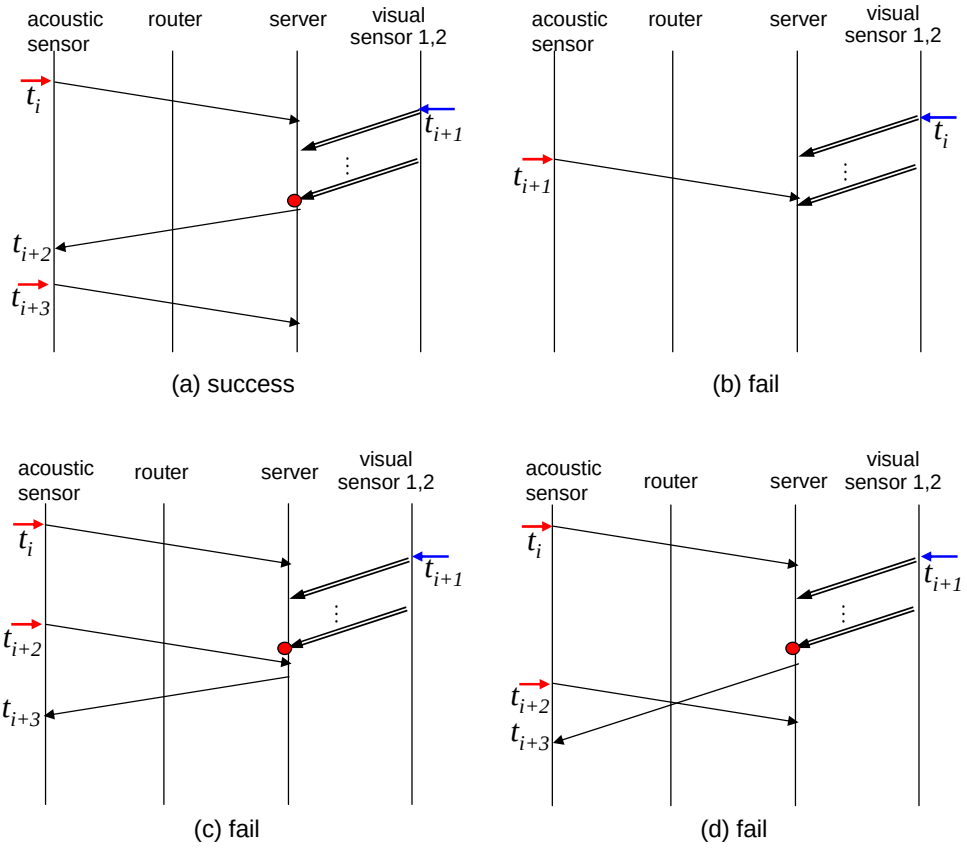


Figure 3-7: An example of *success* and *fail* cases in the tracking model.

Figure 3-7 shows the message flowing diagram between sensors, Router and Server, and possible *success* and *fail* cases occurring in the tracking model. In the figure, the messages related with acoustic sensors are directly delivered from source to destination since they are delivered by UDP. On the other hand, the visual sensor data need reliability so that TCP is used for the transferring. Since the visual image size is larger than Maximum Transmission Unit (MTU) size, more than one packet are exchanged between visual sensors and a Server. Similar to Figure 3-6, we use red and blue arrows, and red point to indicate the generation of the acoustic, visual sampling, and the execution point of localization algorithm. We can find out only Figure 3-7(a) satisfies the *Condition 1* and *Condition 2* at the same time. Note in Figure 3-7(c)

and (d), the final result is not *success* due to the *network synchronization problem* even if the localization algorithm calculation at the Server side is successful.

3.3.5 Impact of Network Synchronization Problem

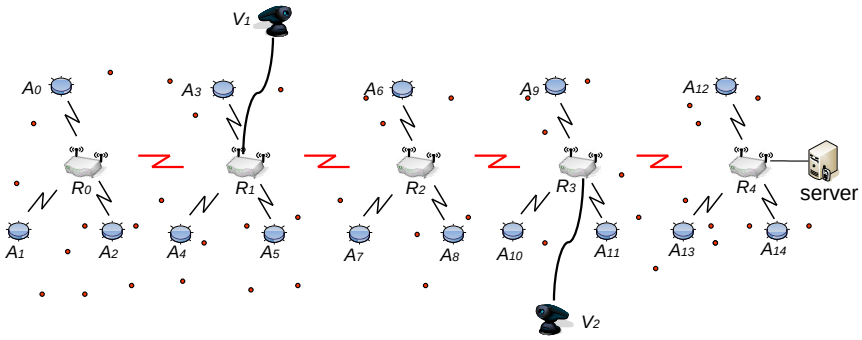
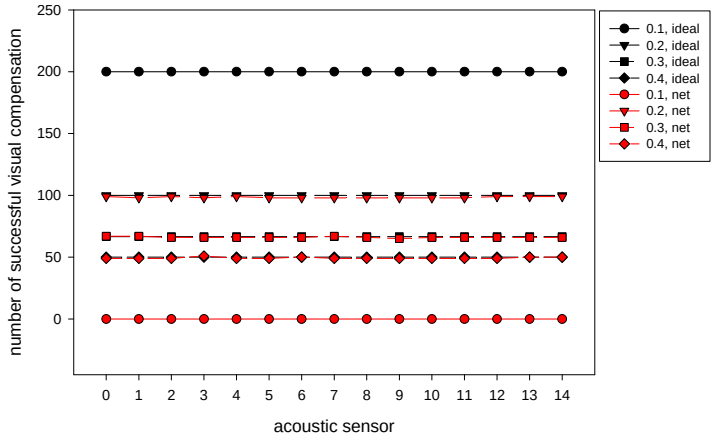


Figure 3-8: Daisy-chain scenario of the tracking model. Acoustic senors, visual sensors and Routers are denoted by A_i, V_i, and R_i $(i = 0, 1, ...)$, respectively.

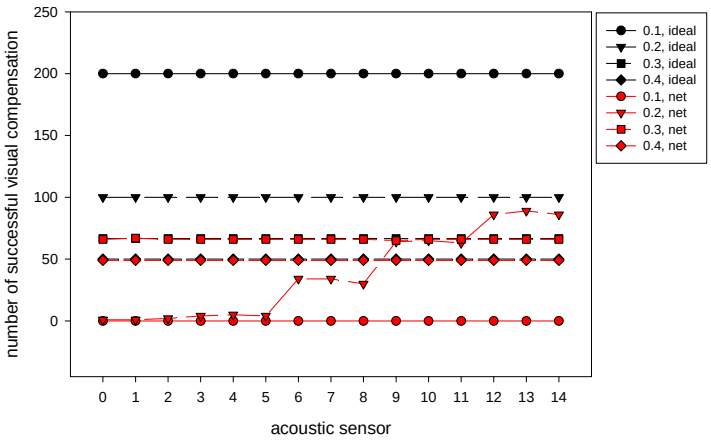
In this section, we investigate the impact of the *network synchronization problem* by simulation study. We use NS-2 simulator to build up a scenario of the developed tracking system as shown in Figure 3-8. Even if its configuration is simple in terms of Routers, the tracking complexity is affected by the number of acoustic sensors and tracking objects. Since the acoustic sensor-Router communication delay is separate from the Router-Router transmission and can be minimized by packet aggregation, we believe it suffices to configure a line of Routers for characterizing the impact of network synchronization and traffic pattern analysis of the developed tracking system. Five Routers are communicating each other with 54Mb/s 802.11a single channel wireless link with 0.0005 uniformly distributed Bit Error Rate (BER). We turn off the RTS/CTS to reduce the network traffic overhead. Three acoustic sensors are located within the communication range of each Router to send and receive data.

The wireless channel between acoustic sensor and Router is different to the channel of Router-Router links to reduce the interference. We assume each acoustic sensor is tracking five moving objects. The Server is connected to last mile router $R4$, and two visual sensors are attached onto $R1$ and $R3$ to effectively capture the tracking space with different angles. The image frames generated by visual sensors are fixed by 40KByte size and we generate 10Kb/s, 0.5Mb/s, and 1Mb/s non-sensor (background) traffic by Constant Bit Rate (CBR). For the visual sampling interval, we set it up as Δt_v=10Δt_s, where Δt_s takes various values: 0.1, 0.2, 0.3, and 0.4 seconds. The simulation time is 200 seconds.

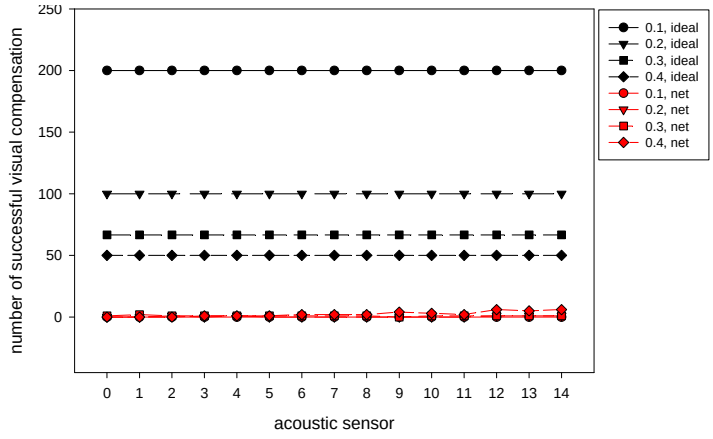
Figure 3-9 shows how many visual compensation process can be successful when the network synchronization is considered in the tracking system. It plots the number of *success* in visual compensation at each acoustic sensor. The black point lines are for the ideal case achieved based on the Figure 3-2. For example, if the acoustic sensors are sampling the object signal with $\Delta t_s = 0.1$ second, correspondingly $\Delta t_v = 1.0$ second, we expect the visual compensation in ideal case is performed 200 times. However, Figure 3-9(a) indicates that in real situation represented by red point line, no visual compensation is performed when $\Delta t_s = 0.1$. This is due to the transmission delay of sensor data, especially, visual images. Note 10Kb/s background traffic could be considered to be equal to a network with only multi-modal sensor traffic. When we measure the transmission delays of 40KBytes visual image under even lower 1kb/s background traffic, we obtain 0.117 and 0.035 seconds of transmission delay for visual sensor 1 and 2. Since the visual sensor 1 is far away from the Server, its delay is



(a) Background traffic: 10Kb/s



(b) Background traffic: 0.5Mb/s



(c) Background traffic: 1Mb/s

Figure 3-9: Impact of *network synchronization problem* in the tracking system. Here, we set $\Delta t_v = 10\Delta t_s$.

larger than that of the second visual sensor and we can conjecture the compensation performance mainly depends on the large delay of the visual sensor 1. This result indicates that we can not realize visual compensation when we try to track a fast moving object with $\Delta t_s = 0.1$ even in no background traffic. When the background traffic increases to 0.5Mb/s and 1Mb/s, the visual sensors can not assist the PF-based object tracking in small Δt_s values. Especially, the *network synchronization problem* leads to around zero visual compensation for all the Δt_s values in 1Mb/s background environment as shown Figure 3-9(c).

3.3.6 Possible Solutions for Network Synchronization Problem

Adjustment of Sampling Time

A simple solution for the network synchronization is achieved from changing the sampling interval of acoustic sensors. For example, in the existence of 1Mb/s background traffic in the previous section, we can eliminate the non-synchronization by increasing the Δt_s to 0.8 second. This solution makes it satisfied with both *Condition*s mentioned in Section 3.3.4. However, this solution has difficulty in supporting the visual-assisted tracking for fast moving objects, so that there exists a limitation of tracking accuracy.

Traffic Differentiation Model

Network transmission delay, especially, the large image exchanging delay among both visual sensors and a Server is critical to the network synchronization. From this fact, we propose a sensor traffic differentiation model by using Weighted Round Robin (WRR) scheduling mechanism to be installed into Routers. The basic idea of the model is that we can balance the network delay among the multi-modal sensor traffic and non-sensor traffic by using WRR, and eliminate the non-synchronization problem appearing in the tracking network.

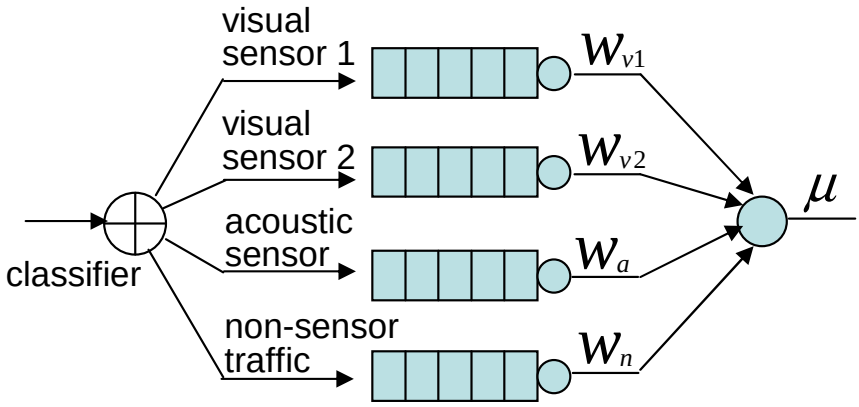


Figure 3-10: Reference model for traffic differentiation.

Figure 3-10 shows the reference model for the traffic differentiation. It has four separate queues and each queue is assigned to visual sensor 1 and 2, acoustic sensor, and miscellaneous non-sensor traffic. The feedback from a Server to acoustic sensors delivering visual localization results is assigned to the second queue. If the service rate of a Router is μ, each queue is served by a service weight factor w_{v1}, w_{v2}, w_a, or w_n. Here, we propose a weight allocation algorithm, Delay-based Weight Allocation (DWA) to determine the weight factor. In DWA, we first categorize the traffic into

sensor and non-sensor traffic and assign weight into them. Let c_s and c_n be the allocated weights for sensor and non-sensor traffic and $c_s + c_n = 1$. Then, we need to perform fine grained weight allocation of c_s into the three different sensor queues, which is conducted based on the transmission delay of each sensor type. The fine grained weight allocation is done by two rounds. In the first round, we need to measure the transmission delay from multi-modal sensors to a Server. Here, we will follow the notation of Figure 3-4. Each value can be easily obtained since at the Service side, we can know the generation time and the arrival time of the sensor data. In the second round, we can obtain the weight allocation based on the measured delays. Let's define the total measured delay d_t as:

$$d_t = x^{v1} + x^{v2} + z. \tag{3.1}$$

Based on the d_t, we get the weight factors as follows.

$$\begin{aligned} w_{v1}^{'} &= c_s \cdot \frac{x^{v1}}{d_t}, \quad & w_{v2}^{'} &= c_s \cdot \frac{x^{v2}}{d_t}, \\ w_{a}^{'} &= c_s \cdot \frac{z}{d_t}, & w_{n}^{'} &= \frac{c_n}{c_s} \cdot d_t, \end{aligned} \tag{3.2}$$

where $w_n^{'}$ is obtained from d_t based on the ratio of the categorized weight allocation. In order to normalize them, we define $w_t = w_{v1}^{'} + w_{v1}^{'} + w_a^{'} + w_n^{'}$, and get the final

DWA formula as:

$$
\begin{aligned}
w_{v1} &= \frac{w_{v1}^{'}}{w_t}, \quad w_{v2} = \frac{w_{v2}^{'}}{w_t}, \\
w_a &= \frac{w_a^{'}}{w_t}, \quad w_n = \frac{w_n^{'}}{w_t},
\end{aligned}
\tag{3.3}
$$

where $w_{v1} + w_{v1} + w_a + w_n = 1$. The accomplished DWA weight factor is now applied for the service differentiation in the next router executions.

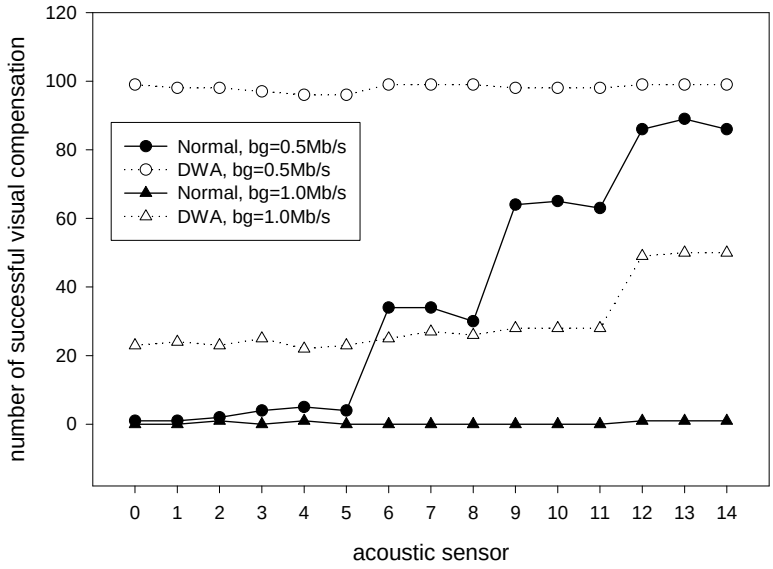
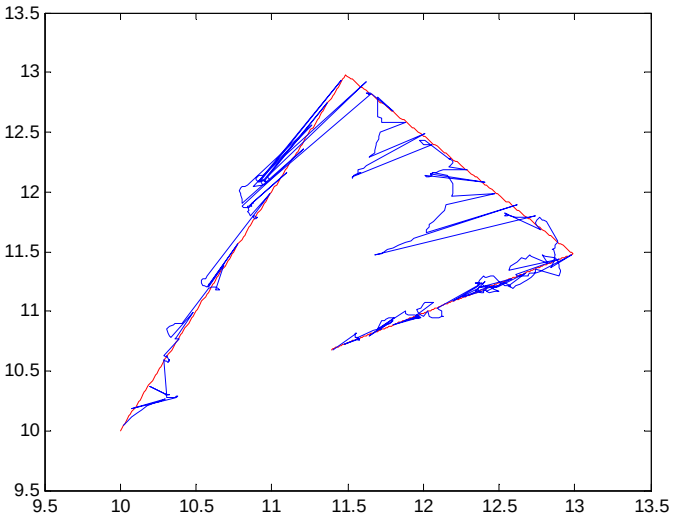


Figure 3-11: Simulation results when traffic differentiation model is applied to the tracking system, where $\Delta t_s = 0.2$ and $\Delta t_v = 10\Delta t_s$.

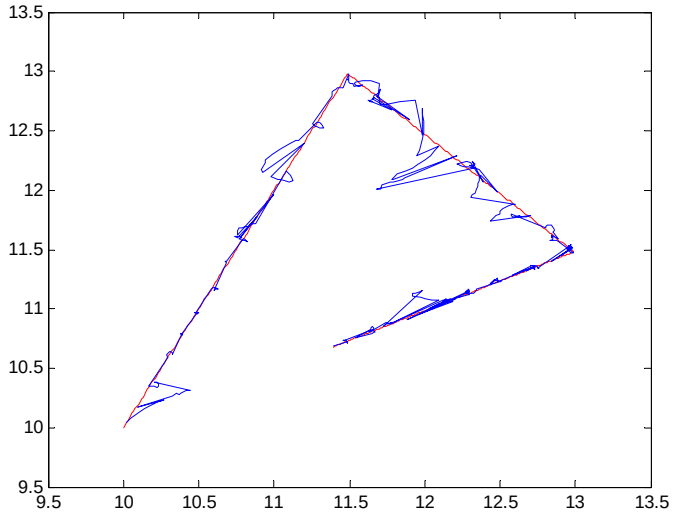
Figure 3-11 shows the simulation result under the 0.5Mb/s and 1.0Mb/s background traffic environments when traffic differentiation model is applied into the tracking system. We compare the the number of *success* of the proposed model with that of normal case in which only one queue serves the sensor and non-sensor traffic. For the categorized weights, we set them up by $c_s : c_n = 0.9 : 0.1$ to support the fast transmission of the sensor data. In the first round, we assign the identical weight for

each queue such as $w_{v1} = w_{v2} = w_a = w_n = 0.25$. In the second round, the initial weights are changed to $w_{v1} = 0.732$, $w_{v1} = 0.222$, $w_a = 0.043$, and $w_n = 0.003$ by means of DWA calculation. The simulation result is obtained only when $\Delta t_s = 0.2$. Note that as indicated in Section 3.3.5, the tracking system achieves no number of *success* in visual compensation at Δt_s=0.1 since the pure transmission delay of visual sensor 1 is larger than 0.1 second. Thus, Δt_s=0.1 case is not plotted in the figure. We also do not plot the cases of Δt_s=0.3, 0.4 under 0.5Mb/s and 1Mb/s background since we achieve the same number of *success* as the ideal case when we apply the traffic differentiation model into the tracking system. When we remind that there are no *success* in Figure 3-9(c) in any Δt_s values, we understand the differentiation model can efficiently mitigate the non-synchronization in network. To save the space, we do not plot the case of 10Kb/s background situation since the result is same as Figure 3-9(a). When we observe the plot for 0.5Mb/s background traffic case, the differentiation model applying the DWA achieves almost the same as the ideal case even if the normal case which reflects the *network synchronization problem* shows the unbalanced visual compensation. Since the acoustic sensor 0 to 5 are far from the Server, their packet transmission delays are larger than the delays of other acoustic sensors. Therefore, we obtain the unbalanced curve in normal case. For the 1.0Mb/s background environment, differentiation model also provides more number of *success* than the normal case even if the result is affected by the background traffic volume.

In order to investigate how the trajectory estimate of object movement changes when the differentiation model is applied, we show the tracking results in Figure



(a) Acoustic sensor 0 to 11



(b) Acoustic sensor 12 to 14

Figure 3-12: Estimated trajectory of a target object under traffic differentiation model.

3-12. This is the result only for DWA,bg=1.0Mb/s case in Figure 3-11. Since the differentiation model provides the same results as the ideal case in DWA,bg=0.5Mb/s, the trajectory estimation in the case is almost the same as the real object movement. The trajectory estimation in Figure 3-12(a) reveals that the differentiation model provides reasonably accurate tracking outcome in acoustic sensor 0 to 11 even with

a little bit large deviation from real object movement. Note that the trajectory estimation of acoustic sensor 0 to 11 under non-differentiation model has the result in Figure 3-3(a) since the normal one queue model achieves around zero *success* in visual compensation. The acoustic sensor 12 to 14 show better trajectory estimation in Figure 3-12(b) since they are assisted with more *success* of visual compensation.

3.3.7 Behavior Analysis of the Tracking System

From the observation of the developed tracking system, we can understand the successful visual compensation mainly depends on the sampling interval of the acoustic sensor and the image transmission delay. Therefore, in this section, we investigate the accuracy of a tracking system in terms of the two parameters.

Performance Metric

In order to represent the *success* and *fail* cases mentioned in Section 3.3.4 by a numeric value, we define a new performance metric applicable to the tracking system. We call it Successful Compensation Rate (SCR) and define it as:

$$\text{SCR} = \frac{n_s}{n_t}, \tag{3.4}$$

where n_s is the number of *success* in visual compensation that satisfies both *Conditions* in Section 3.3.4 and n_t is the total number of sampling of an object signal at an acoustic sensor. If total running time of the tracking system is T, $n_t = T/\Delta t_s$. If a tracking model achieves large SCR value, the PF algorithm is highly compensated

by localization algorithm, so that we can more accurately track the target object. Therefore, the SCR metric can be a gauge to determine the tracking accuracy of the established tracking system. Note the accomplished SCR reflects the *network synchronization problem*.

Simulation Setup

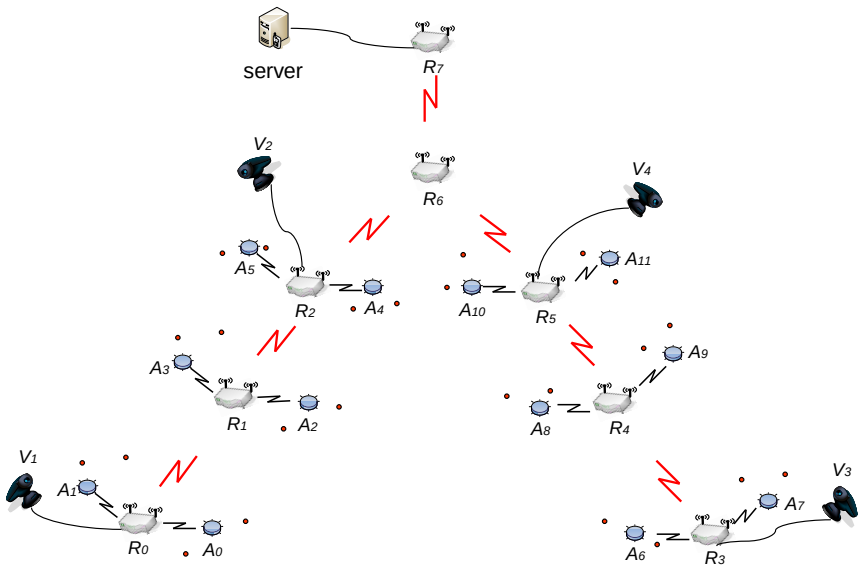


Figure 3-13: Tree scenario of the tracking system. Due to a line-of-sight characteristics of visual sensors, we install 4 visual sensors.

In order to observe the system behaviors, we perform the simulations based on the scenarios in Figure 3-8 and a more complex tree scenario as shown in Figure 3-13. Visual sensors generate image frame of 20, 40, and 60 KBytes size. We observe the behaviors based on the acoustic sampling interval of 0.1, 0.15, 0.2, 0.3, and 0.4. For the tree scenario, we set up the same communication link as Figure 3-8, and only two acoustic sensors work in the communication range of each Router. Each acoustic sensor tracks five objects. We assume that both branches of the tree can not

guarantee the line-of-sight characteristics of a visual sensor, so that we install two visual sensors for each branch.

Simulation Results

Figure 3-14 shows the simulation results achieved from daisy-chain scenario. We plot the SCR variations of 15 acoustic sensors. Let's assume the both tracking scenarios need SCR = 0.6 to detect the object trajectory. Then, in 20KBytes image size, the acoustic sensors do not need to capture the object signal with $\Delta t_s < 0.3$ since the network synchronization problem blocks the on-time transmission of the sensor data for the visual compensation. If the visual sensors generate 40KBytes image, only $t_s = 0.4$ supports the expected SCR value. In case of 60KBytes, all the simulated sampling times do not support the stable object tracking.

For the tree scenario, we plot only SCR result for 20KBytes image size in Figure 3-15 since the other cases support no visual compensation. This is because the complex network configuration leads to the large transmission delay of the visual image to a Server. Even in 20KBytes, only acoustic sensor 4 and 5 which are near the Server support the expected SCR = 0.6.

3.4 Statistical Estimation and Adaptation for Visual Compensation

3.4.1 Motivation

In the proposed tracking system, it is necessary to investigate the number of *success* of visual compensation to appear in the middle of tracking. This objective can be done by measuring the SCR achievement by simulation or system setup. However, the two measuring methods are time consuming task, but a mathematical estimation method could be a good tool for the investigation. Therefore, we formulate a SCR estimation algorithm in the following section.

Another question in the tracking system could be how to maintain the tracking system at a certain level of tracking accuracy. From the previous observation of the tracking system, we expect the *success* in visual compensation depends on the sampling interval of the acoustic sensor and the transmission delay, i.e., x^{v1}, x^{v2}, y, and z. However, the tracking system shows different behaviors in times since the delay factor has randomness property. Therefore, the maintaining of the accuracy could not be an easy part since the accurate tracking depends on the *success* in the visual compensation that subsequently depends on the sampling interval of the acoustic sensor and the transmission delay. Fortunately, the acoustic sampling interval is a predictable parameter since we can set it up in the acoustic sensor by a certain value. Thus, we can perform the maintenance by controlling the sampling interval. In order to do the adaptation of the parameter in real-time manner, the next section

proposes an adaptation algorithm by using the previously formulated SCR estimation algorithm.

3.4.2 Statistical Estimation Algorithm (SEA)

In this section, we explain a Statistical Estimation Algorithm (SEA) to predict the SCR variation in the multi-modal tracking system.

Before we illustrate the algorithmic details of the SEA, we first investigate a packet flowing example in Figure 3-16. Similar to Figure 3-6, this traffic pattern could happen in a situation that visual images are sent as many as possible since the acoustic signal of an object frequently disappears. The transmission delays x_i^{v1}, x_i^{v1}, y_j, and z_k take random values according to the status of the tracking network. From the figure, we can guess x_i^{v1} and x_i^{v2} are the key factors to make a visual compensation successful. The acoustic sensor samples the object signal every Δt_s interval. When we observe the time period between t_1 and t_7, it satisfies the both *Condition*s in Section 3.3.4. In this period, the acoustic sensor successfully receives the feedback from the Server, so that the SCR increases. In the time period between t_7 and t_{11}, the Server does not perform the localization algorithm since the visual sampling times t_4 and t_5 of x_2^{v2} and x_2^{v1} are earlier than the acoustic sampling time t_7, which violates the *Condition 1* in Section 3.3.4. At this point, we need to define another condition to simplify the compensation process.

- *Condition 3*: Only one association of acoustic sensor and visual sensor data is permitted in a Δt_s period.

For example, in time period t_7 to t_{11}, we do not perform the visual compensation process since z_2 has already used in the comparison with x_2^{v1} and x_2^{v2} even if x_3^{v1}, x_3^{v2}, and z_2 satisfy the *Condition 1*. Based on the *Condition 3*, we can match the current acoustic data with the recent visually sampled image in case there are many possible visual images that are feasible for the localization algorithm. We also perform quick decision of executing the localization algorithm without a complicated algorithm to match the sampling times of acoustic and visual data. According to the *Condition 1* to *Condition 3*, we can figure out why we do not perform the localization algorithm and increase the SCR in the third and fourth periods.

From this example, we can get an insight that SCR could be estimated by well-defined prediction model if packet transmission delays between multi-modal sensors and Server are correctly modeled and generated in the model.

Observation and Approximation of Packet Transmission Delay

We start deriving the SEA from the approximation of x_i^{v1}, x_i^{v2}, and y_j. For this work, we observe the transmission delay variations from NS-2 simulations which has the daisy-chain and tree scenarios like Figure 3-8 and 3-13, and obtain the Probability Density Functions (PDF) to generate random values to predict the transmission delays. Here, we do not measure z_k and find its PDF since Δt_s includes the z_k in the SEA operation.

Figure 3-17 shows the histogram of transmission delays, x_i^{v1} and y_j (i, j=1,2,3,...). For the measurement, we use daisy-chain and tree topologies with 20Kbytes visual

image size and $\Delta t_s = 0.1$. We plot the delays only for visual sensor 1 and acoustic sensor 0 on both topologies since other visual sensor and acoustic sensors have similar distribution patterns.

The histograms of the visual sensor in Figure 3-17(a) and 3-17(b) show Gaussian-like distributions with bi-modality and uni-modality in daisy-chain and tree topologies. In general, it is reasonable to assume that an internet traffic follows Exponential PDF for the transmission delay. However, our measurement does not follow the assumption. This is because the obtained histogram is not for a set of one packet delay but for the delay of one image file. For example, 20 KBytes image files are delivered by 20 separate packets each of which has 1000 Bytes size. Therefore, we can not say that the transmission delay of 20 KBytes image follows the Exponential PDF. To more clearly explain why we get the Gaussian-like PDF, let's consider a statistic example where we pick up 20 samples from a population having an arbitrary PDF. Then, the sum of 20 samples becomes a random variable, and the random variable approximately follows Gaussian distribution according to the Central Limit Theorem (CLT) [72]. Similar to this example, we can say that the transmission delay of 20 KBytes image file is a random variable summing transmission delays of 20 separate packets, and the random variable follows a Gaussian PDF according to the CLT. Even if the distribution with bi-modality is not the actual Gaussian PDF, we approximate the distribution as Gaussian in the SEA, which might give an estimation error in the SEA calculation.

Let X be the random variable for x_i^{v1} and x_i^{v2}. Then, X has PDF $f_X(x)$ as follows.

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{3.5}$$

where μ is the expected value and σ is the standard deviation.

For the acoustic sensor, the histogram is close to Gamma PDF with appropriate shape (k) and scale (θ) parameters. Let's define Y as the random variable for y_j. Then, Y has following PDF:

$$f_Y(y) = y^{k-1}\frac{e^{y/\theta}}{\theta^k \Gamma(k)}, y > 0, \tag{3.6}$$

where $k > 0$, $\theta > 0$. Note the Exponential PDF can be formulated from Gamma PDF with $k = 1, \theta = 1/\lambda$, where λ is a rate of the Exponential distribution. In order to simplify the random number generation for y_j in the SEA, we use Exponential PDF instead of Gamma PDF. The Exponential assumption is reasonable in that an internet traffic generally follows the Exponential PDF. In the validation section, this simplification is proven to be reasonable. Finally, we can determine the PDF for Y as:

$$f_Y(y) = \begin{cases} \lambda e^{-\lambda y} & , \ y \geq 0, \\ 0 & , \ y < 0. \end{cases} \tag{3.7}$$

Formulation of the SEA

Based on the previous PDF approximation, we explain the details of the SEA in this section. SEA is composed of four modes, Right (R), Left (L), Both (B), and Unreachable (U) and two states, success (s) and fail (f). Each mode except for U mode contains two states.

The meaning of the R, L, and B is well explained in Figure 3-16. Let's look at the red dashed lines that are vertically drawn at t_1, t_7, t_{11}, t_{15}, and t_{20}. The sampling times of x_i^{v1} and x_i^{v2} are placed along the lines with three different forms: (1) the sampling times are located at the right-hand side of the line such as (x_1^{v1}, x_1^{v2}), (2) they are at the left-hand side of the line like (x_2^{v1}, x_2^{v2}), (x_5^{v1}, x_5^{v2}), and (x_7^{v1}, x_7^{v2}), and (3) they are crossly placed on both sides such that the sampling time of x_4^{v1} is at the right-hand side and that of x_4^{v2} is at the left-hand side based on the line at t_{11}. We define the first case as R mode, second one as L mode, and third one as B mode. Once the mode is determined, we can find whether the compensation process is *success* or *fail* within the mode after refereing to the size of y_j. Note that the y_j can be created only if a server successfully performs the localization algorithm by using x_i^{v1}, x_i^{v2}, and z_k as parameters. If we have a *success* like y_1 that safely reaches an acoustic sensor, we call this case as s state; otherwise, f state. In s state, we can increase the SCR value. Note three modes in Figure 3-16 are unreachable to the next mode in some cases. For example, the s state in R mode between t_1 and t_7 can not yield R or B mode due to the traffic characteristics in the tracking network. Here, we define the unreachable case as U mode. We can guess the four modes and two states

arbitrarily happen in real situations. This uncertain mode and state occurrences are effectively modeled by state transition rules in the SEA.

Figure 3-18 shows the state transition rules between modes and states in the SEA. Each mode is represented by colored rectangle, and the s and f states are indicated by circle inside of the rectangle. For example, R_s and R_f stand for s and f states in R mode. In the figure, the solid arrow lines indicate possible mode-state transitions, and the dashed arrow lines stand for U mode. As drawn by the arrow lines, the transitions are allowed only from state-to-mode, and there are no mode-to-mode, state-to-state, and mode-to-state transitions. The transition rules in the state-to-mode are based on Δt_s and random values of transmission delay following the Gaussian and Exponential PDFs of random variables X and Y. X can take four different values: x_i^l, x_i^s, x_{i+1}^l, and x_{i+1}^s. The subscript i and $i+1$ represent the current and next image frame captured by a visual sensor. Whenever two visual sensors send the captured image frames, the transmission delays of the images depend on the current status of the tracking network. According to the size of the delay, we assign the superscript l and s. For example, if the transmission delay of visual sensor 1 is larger than that of visual sensor 2 for i_{th} image, that is, $x_i^{v1} > x_i^{v2}$, we determine l and s as follows:

$$x_i^l \leftarrow x_i^{v1}, \quad x_i^s \leftarrow x_i^{v2}. \tag{3.8}$$

The $(i+1)_{th}$ image also follows the l and s decision of i_{th} image. Therefore, in this

case, we have:

$$x_{i+1}^{l} \leftarrow x_{i+1}^{v1}, \quad x_{i+1}^{s} \leftarrow x_{i+1}^{v2}. \tag{3.9}$$

The reason why we need $(i+1)_{th}$ image frame is clarified when we explain the operation of the SEA in this section. The random variable Y takes a value y_j. In the transition rules, we use α to indicate what percent of x_i^{v1} and x_i^{v2} is slanted toward the right-hand side of the red dashed line explained in Figure 3-16. For example, at t_{15}, around 60% of x_5^{v1} is slanted to the right-hand side of the red line; but around 20% for x_5^{v2}. Then, $\alpha \approx 0.6$ for x_5^{v1} and $\alpha \approx 0.2$ for x_5^{v2}. Since it is difficult to determine α for each situation, we can fix it to a constant value obtained from heuristic mathematical calculation. From the transition rules of the SEA, we can approximate the next status of the tracking network based on the current status.

To help the understanding of the transition rules in the SEA, we explain the details by comparing Figure 3-16 with Figure 3-18. Figure 3-16 starts from R mode since x_1^{v1} and x_1^{v2} are right-hand side of the red dashed line at t_1. According to Equation (3.8), we have:

$$x_1^{l} \leftarrow x_1^{v1}, \quad x_1^{s} \leftarrow x_1^{v2}, \tag{3.10}$$

where $x_1^{v1} > x_1^{v2}$. Since y_1 occurs within Δt_s, we have a *success*, and current state becomes R_s. Here, we have a transition rule: R_s takes place if and only if $x_1^l + y_1 \leq \Delta t_s$ and $x_1^s + y_1 \leq \Delta t_s$, which subsequently results in L mode in the next Δt_s period.

In Figure 3-16, we can understand why the transition from R_s to B or R mode is unreachable. For R_s to B transition, the sampling of z_2 has to take place between t_4 and t_5. However, this case does not take place since current state is R_s, and the arrival point of y_1 should exist after t_5. Due to the same reason, R_s can not be reachable to R mode. Similar to this example, we can figure out why L_s is unreachable to R and B mode. We are now in L mode, and x_2^{v1} and x_2^{v2} are now used to determine the next status. Here, we require the next transmission delays x_3^{v1} and x_3^{v2} to determine a transition. Note we do not know the next delays in real situations. However, in the SEA, we can obtain the estimate values for the next delays by using the pre-determined Gaussian and Exponential PDFs under the assumption that we have known the PDF parameters: μ and σ of Gaussian PDF, and λ of Exponential PDF. We are now able to generate the next delays x_3^{v1} and x_3^{v2} from the PDFs, and determine the current state. If following rule is satisfied, current state becomes L_s.

$$\alpha x_2^l + x_3^l + y_2 \leq \Delta t_s \ \wedge \ \alpha x_2^s + x_3^s + y_2 \leq \Delta t_s. \tag{3.11}$$

Otherwise, the current state becomes L_f. Here, we assume $x_2^{v1} > x_2^{v2}$, and according to Equation (3.8) and (3.9), the superscript l and s have been determined as follows.

$$\begin{aligned} x_2^l \leftarrow x_2^{v1}, x_2^s \leftarrow x_2^{v2} \\ x_3^l \leftarrow x_3^{v1}, x_3^s \leftarrow x_3^{v2}. \end{aligned} \tag{3.12}$$

Unfortunately, time period (t_7, t_{11}) does not satisfy Equation (3.11). Therefore, L_f

becomes the current state. Now, we have to decide the next mode. We realize (t_7, t_{11}) period is under the following condition which is the third or-combined transition rule in C_{Lf3}.

$$\begin{aligned} &(\alpha x_2^l < \Delta t_s \wedge \alpha x_2^s < \Delta t_s) \wedge \\ &(\alpha x_2^l + x_3^l \leq \Delta t_s) \wedge \\ &(\alpha x_2^l + x_3^l + y_2 > \Delta t_s) \wedge \\ &(\alpha x_2^s + x_3^s < \Delta t_s). \end{aligned} \tag{3.13}$$

Equation (3.13) now results in B mode in the next Δt_s period. Once we enter the B mode, we heuristically define the mode always transits to f state. This is why the B_s yields only U mode. For example, if we assume that $x_4^l \leftarrow x_4^{v1}$ and $x_4^s \leftarrow x_4^{v2}$, the condition $\alpha x_2^s + x_3^s \leq \Delta t_s$ causes x_4^s to start at the left-hand side from t_{11} line. This leads to *fail* since it does not satisfy *Condition 1.* Even though we illustrate the mode and state transition from one typical example, we believe the other transition rules in Figure 3-18 cover almost all the transitions appeared in the tracking system, which will be verified in Section 3.4.4.

The Algorithm 2 describes the pseudo code of the SEA, which is based on the transition rules in Figure 3-18. In the algorithm, we assume the population parameters for X and Y have already known such that X has (μ_{v1}, σ_{v1}) and (μ_{v2}, σ_{v2}) for visual sensor 1 and 2, and Y has λ_y. Based on the parameters, we generate Gaussian random numbers from Equation (3.5) and Exponential numbers based on Equation (3.7). After we have five random numbers, we determine x_i^l, x_i^s, x_{i+1}^l, x_{i+1}^s by compar-

Algorithm 2: Pseudo code of the SEA

Input: μ_{v1}, σ_{v1}, μ_{v2}, σ_{v2}, λ_y, Δt_s
Output: SCR

SCR←0;
foreach Δt_s **do**
  // random number generation for X and Y
  $x_i^{v1} \leftarrow$ Gausrnd (μ_{v1}, σ_{v1}), $x_{i+1}^{v1} \leftarrow$ Gausrnd (μ_{v1}, σ_{v1}) ;
  $x_i^{v2} \leftarrow$ Gausrnd (μ_{v2}, σ_{v2}), $x_{i+1}^{v2} \leftarrow$ Gausrnd (μ_{v2}, σ_{v2}) ;
  $y_j \leftarrow$ Exprnd (λ_y);

  // determine x_i^l, x_{i+1}^l, x_i^s, x_{i+1}^s
  if $x_i^{v1} > x_i^{v2}$ **then**
    $x_i^l \leftarrow x_i^{v1}$, $x_{i+1}^l \leftarrow x_{i+1}^{v1}$, $x_i^s \leftarrow x_i^{v2}$, $x_{i+1}^s \leftarrow x_{i+1}^{v2}$;
  else
    $x_i^l \leftarrow x_i^{v2}$, $x_{i+1}^l \leftarrow x_{i+1}^{v2}$, $x_i^s \leftarrow x_i^{v1}$, $x_{i+1}^s \leftarrow x_{i+1}^{v1}$;
  end

  // determine next mode and update SCR
  switch *mode* **do**
    case R
      if $x_i^l + y_j < \Delta t_s$ *and* $x_i^s + y_j < \Delta t_s$ **then**
        SCR←SCR+1, $mode \leftarrow L$;
      else
        if $x_i^l > \Delta t_s$ *and* $x_i^s > \Delta t_s$ **then** $mode \leftarrow R$;
        else $mode \leftarrow B$;
      end
    case L
      if $\alpha x_i^l + x_{i+1}^l + y_j \leq \Delta t_s$ *and* $\alpha x_i^s + x_{i+1}^s + y_j \leq \Delta t_s$ **then**
        SCR←SCR+1, $mode \leftarrow L$;
      else
        if $\alpha x_i^l > \Delta t_s$ *and* $\alpha x_i^s > \Delta t_s$ **then**
          $mode \leftarrow R$;
        else if $\alpha x_i^l < \Delta t_s$ *and* $\alpha x_i^s < \Delta t_s$ **then**
          if $\alpha x_i^l + x_{i+1}^l > \Delta t_s$ **then**
            if $\alpha x_i^s + x_{i+1}^s > \Delta t_s$ **then** $mode \leftarrow L$;
            else $mode \leftarrow B$;
          else
            if $\alpha x_i^l + x_{i+1}^l + y_j > \Delta t_s$ **then**
              if $\alpha x_i^s + x_{i+1}^s < \Delta t_s$ **then** $mode \leftarrow B$;
              else $mode \leftarrow R$;
            end
          end
        else
          $mode \leftarrow B$
        end
      end
    case B
      if $x_i^l > \Delta t_s$ *and* $\alpha x_i^s > \Delta t_s$ **then** $mode \leftarrow R$;
      else if $x_i^l < \Delta t_s$ *and* $\alpha x_i^s < \Delta t_s$ **then** $mode \leftarrow L$;
      else
        $mode \leftarrow B$;
      end
    end
  end
end

ing x_i^{v1} with x_i^{v2}. Then, modes and states are exchanged according to the transition rules.

3.4.3 Statistical Estimation and Adaptation Algorithm (SEA^2)

In this section, we propose a Statistical Estimation and Adaptation Algorithm (SEA^2) to answer the system maintenance problem mentioned in Section 3.4.1. If we are given a target SCR (t_{scr}) that represents a level of tracking accuracy, the objective of this algorithm is to achieve the t_{scr} in a real-time manner. SEA^2 uses the SEA to get the initial acoustic sampling interval. Therefore, we need to predict the population parameters of the PDFs to start the algorithm.

Figure 3-19 shows the procedural illustration of the SEA^2 that is composed of two Phases. The prediction of population parameters is done in Phase 1, and the automatic adaptation of acoustic sampling interval is performed in Phase 2.

The SEA^2 is performed between acoustic sensors and a Router based on two message types: PT_TSRQ and PT_TSRP. When we remind the tracking system configuration, the acoustic sensors and a Router have n-to-1 mapping, where one Router needs to control n acoustic sensors to adjust the acoustic sampling interval. In Phase 1, the Router sends PT_TSRQ every Δt_a interval to synchronize the arrival of response message PT_TSRP from n acoustic sensors. Generally, we set up Δt_a to a larger value than the acoustic sampling interval, so that a number of compensation processes are performed within Δt_a. As soon as receiving the request packet, acoustic sensors set the sampling interval to $\frac{\Delta t_s^c}{2^i}$ (i=0,1,2...) which means an exponential de-

crease of a constant sampling interval Δt_s^c. The exponential factor i increases by 1 from 0 whenever acoustic sensors receive PT_TSRQ, and Δt_s^c is set to a reasonably large value. The main reason we use the exponential decrease is to fast advance to Phase 2, which is beneficial for finding the acoustic sampling interval to achieve t_{scr}. Actually, SEA is more sensitive not to the size of population parameters but to the acoustic sampling interval, so that we do not have to consume much time to obtain accurate population parameters in Phase 1. Note that we achieve 100% SCR when the acoustic sampling interval is sufficiently large. Therefore, as we exponentially reduce the sampling interval, the SCR also decreases in proportion to the interval. Since acoustic sensors have different size of sampling interval within Δt_a, they perform n_i number of sampling, which is denoted by $n_i \frac{\Delta t_s^c}{2^i}$. In Phase 1, acoustic sensors measure SCR value within $n_i \frac{\Delta t_s^c}{2^i}$, and send the information to the Router.

Let's define the estimated means and standard deviations of X in Phase 1 as ($\hat{\mu}_{v1}$, $\hat{\sigma}_{v1}$) and ($\hat{\mu}_{v2}$, $\hat{\sigma}_{v2}$), and the estimated mean of Y as $\hat{\lambda}_y$. To obtain ($\hat{\mu}_{v1}$, $\hat{\sigma}_{v1}$) and ($\hat{\mu}_{v2}$, $\hat{\sigma}_{v2}$), the Server piggybacks the transmission delays of visual images in the feedback packet. When a Router is relaying the feedback to acoustic sensors, it can obtain the transmission delay, and subsequently ($\hat{\mu}_{v1}$, $\hat{\sigma}_{v1}$) and ($\hat{\mu}_{v2}$, $\hat{\sigma}_{v2}$). The acoustic sensors measure the SCR and transmission delay from the Server to itself, and send them to the Router by using PT_TSRP. Then, the Router uses the received delay to obtain the $\hat{\lambda}_y$. This procedure continues until we have SCR$< \gamma$, where γ is a confidence level and set by a constant value. The estimated parameters for X and Y are obtained from averaging the measured delays so far. Now, we can run the SEA to obtain an

initial acoustic sampling interval Δt_s^0. For the input parameters of the SEA, we use the estimated population parameters as follows.

$$\begin{aligned} &\mu_{v1} \leftarrow \hat{\mu}_{v1}, \sigma_{v1} \leftarrow \hat{\sigma}_{v1}, \\ &\mu_{v2} \leftarrow \hat{\mu}_{v2}, \sigma_{v2} \leftarrow \hat{\sigma}_{v2}, \\ &\lambda \leftarrow \hat{\lambda}_y. \end{aligned} \tag{3.14}$$

Algorithm 3: Adjustment algorithm of the acoustic sampling interval Δt_s^i

Data: $e_1 \leftarrow 0, e_2 \leftarrow 0$

if $SCR < t_{scr} - \epsilon$ **then**
  $\Delta t_s^i \leftarrow \Delta t_s^i + \delta \times 2^{e_1}$ and $e_1 \leftarrow 0, e_2 \leftarrow e_2 + 1$;
else if $SCR > t_{scr} + \epsilon$ **then**
  $\Delta t_s^i \leftarrow \Delta t_s^i - \delta \times 2^{e_2}$ and $e_1 \leftarrow e_1 + 1, e_2 \leftarrow 0$;
else
  $e_1 \leftarrow 0, e_2 \leftarrow 0$;
end

Phase 2 starts with sending Δt_s^0 from a Router to acoustic sensors. Different from Phase 1, the Router receives the SCR from the acoustic sensors for $\kappa \Delta t_a$ and controls the sensors to adapt the sampling interval Δt_s^i $(i = 1, 2, ...)$ based on Algorithm 3, where κ is a constant positive integer. We maintain Δt_s^i for a longer period $\kappa \Delta t_a$ since the tracking network fluctuates in times and it is not sufficient to determine the object sampling interval only by observing one Δt_a period. In Algorithm 3, in order to do fast adjustment of the sampling interval from the continuous miss from t_{scr}, we use exponential increase and decrease of the acoustic sampling interval by $\delta 2^{e_1}$ or $\delta 2^{e_2}$, where e_1 and e_2 alternately increase according to their conditions, and δ should be set to a proper value. If δ is too small, we do not get the advantage of fast increase

and decrease; otherwise, the measured SCR keeps deviated from the t_{scr}. If the SCR measured for $\kappa\Delta t_a$ is less than $t_{scr} - \epsilon$, we perform increment; otherwise, decrement is done. The ϵ is a statistical tolerance error which reflects the fluctuation of tracking network traffic. We recommend 5% tolerance error to sufficiently include the network fluctuation.

3.4.4 Algorithm Validation and Discussion

In order to conduct the validation of the proposed algorithms, we use NS-2 simulator. We configure daisy-chain and tree-based tracking system like Figure 3-8 and 3-13 except for the multiple channel for Router-Router communication and the single channel for acoustic sensor-Router communication. From the use of different channel, we can minimize the channel interference among the Routers. In the scenarios, each acoustic sensor is sampling 5 objects. 20 Kbytes or 40 KBytes visual image is generated by visual sensors and delivered by 1000 Bytes TCP packets.

Validation of the SEA

We verify that the SEA accurately estimates the SCR by comparing simulation results with mathematical calculation of the SEA. To do this work, we obtain SCR results (%) for five different acoustic sampling intervals (Δt_s): 0.1, 0.15, 0.2, 0.3, 0.4. In order to observe precise variation in short Δt_s cases, we include the result of $\Delta t_s = 0.15$. It is possible to track high speedy objects and obtain more accurate tracking information on the same objects if Δt_s gets shorter. As visual image size increases, we also get

more information from the image, so that we can increase the accuracy of visual localization algorithm at a Server.

Figure 3-20 and Figure 3-21 plot SCR results from the simulation and numerical calculation. We first observe the SCR from simulations drawn by solid lines with black points. The plots indicate that small SCRs are achieved in acoustic sensors far from the Server, short Δt_s, and small visual image size. These results are obvious since the *Condition 1* to *Condition 3* in Section 3.3.4 and Section 3.4.2 are well satisfied under the above tracking system environment. Especially, Figure 3-21(b) shows that all the SCRs are very small since the tree scenario and larger image size cause non-synchronization between multi-modal sensors. The SCR estimates from the SEA are drawn by dotted lines with white points in the figures. For the input population parameters (μ_{v1}, σ_{v1}), (μ_{v2}, σ_{v2}), and λ_y for the SEA, we use the measured values from simulations. We can observe that almost all SEA calculation results are close to the simulation results except for some cases like $\Delta t_s = 0.15$ in Figure 3-20(a), and $\Delta t_s = 0.3$ in Figure 3-20(b) and 3-21(a), which have around 20% deviation from the simulation results.

Validation of the SEA^2

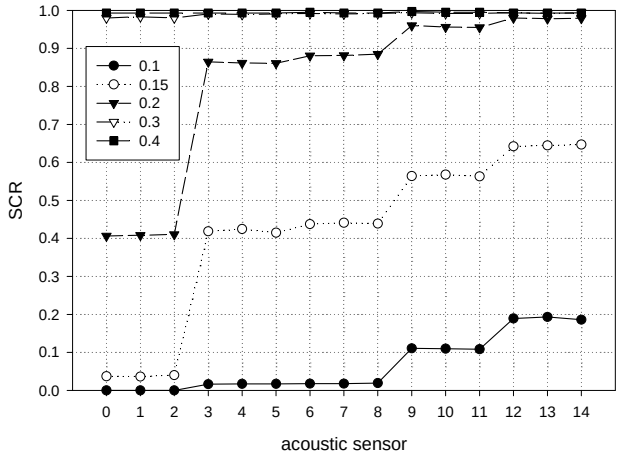
For the validation of SEA^2, we observe how the SEA^2 automatically adapts the acoustic sampling interval to accomplish a given t_{scr}. To do this work, we run NS-2 simulations for 2400 seconds, and measure the sampling interval variations every 10 seconds interval. In Phase 1, the confidence level γ is set to 90%, and SEA^2 starts

its running by setting the initial sampling interval Δt_s^c as 1.2 second. In Phase 2, the statistical tolerance error ϵ is set to 5%, and δ for controlling the exponential increase and decrease level is set to 0.01. We fix $\kappa = 4$ for SCR observation interval.

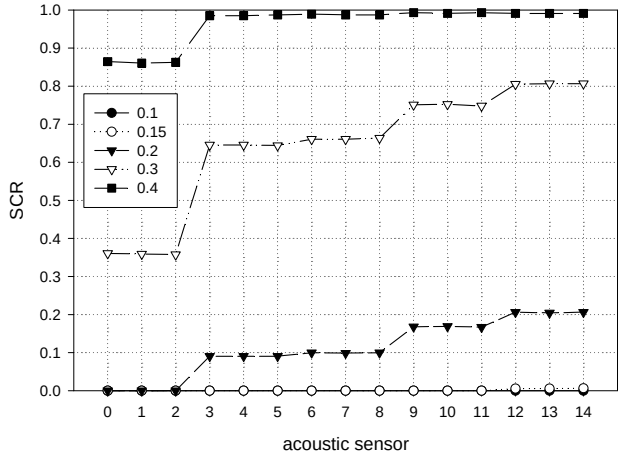
Figure 3-22 shows the automatic adjustment of the acoustic sampling interval to achieve $t_{scr} = 90\%$. To obtain the plots, we run simulations ten times and take average values. The results are observed at each Router to prevent the plotting all the acoustic sensors' sampling intervals, which is reasonable since the router controls the acoustic sensor behaviors within its communication range. In Figure 3-22(a), we understand $R0$ to $R3$ achieve 90% SCR at around $\Delta t_s = 0.19$. However, $R4$ accomplishes it at shorter value, $\Delta t_s = 0.16$ since $R4$ directly connects to the Server and fast exchanges sensor data with the Server. Note we achieve 60% SCR at $\Delta t_s = 0.15$ and around 100% SCR at $\Delta t_s = 0.2$ in Figure 3-20(a). This indicates that SEA^2 correctly adapts the acoustic sampling interval to achieve 90% target SCR. When the visual image size increases to 40Kbytes in daisy-chain topology as shown in Figure 3-22(b), Δt_s saturates to between 0.30 and 0.35 after fluctuating for initial simulation time duration. The correctness of this adaptation is also verified in Figure 3-20(b). As a tracking scenario is complicated to tree and the file size is large, the Δt_s shows a larger variations as shown in Figure 3-22(c) and 3-22(d). However, the adaptation to achieve 90% SCR is verified to be correct if we compare the results with Figure 3-21(a) and 3-21(b). In the worst scenario like tree and 40Kbytes image size, 90% SCR is achieved when $\Delta t_s > 0.65$. From these results, we have knowledge that we should reduce the visual image size to increase successful compensation rate in case

we have to construct a complicated tracking environment.

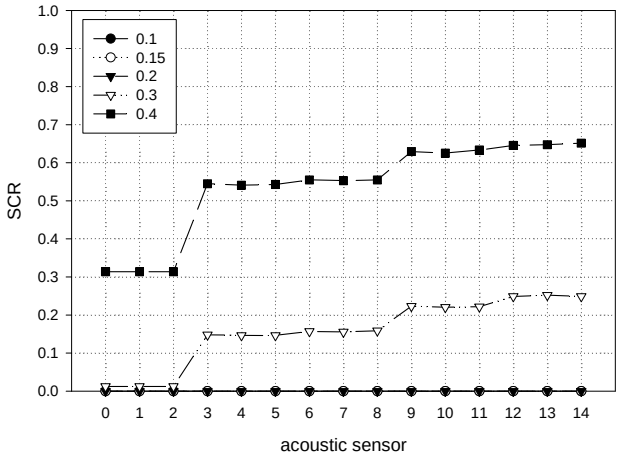
In order to observe real-timing adaptation capability of the SEA^2, we change t_{scr} during simulations, and observe the acoustic sampling interval variations as shown in Figure 3-23, where we include only 20Bytes image cases for daisy-chain and tree topologies. When simulations start, we fix t_{scr} to 30%. After simulation time reaches 800 seconds, we change the t_{scr} to 60%. Finally, $t_{scr} = 90\%$ when the time becomes 1600. In Figure 3-23(a), we observe SEA^2 is efficiently adaptive to the changes of t_{scr}. The Δt_s variations at 30% and 60% t_{scr} are around 0.02 in worst case, which is expect to be tolerable when we compare it with the exponential increase and decrease level (δ) which is set to 0.01. Figure 3-23(b) indicates SEA^2 provides confidential acoustic sampling interval adaptation mechanism even if the acoustic sampling interval variation in tree is larger than that of daisy-chain scenario.



(a) image size: 20KBytes



(b) image size: 40KBytes



(c) image size: 60KBytes

Figure 3-14: SCR results achieved from daisy-chain scenario.

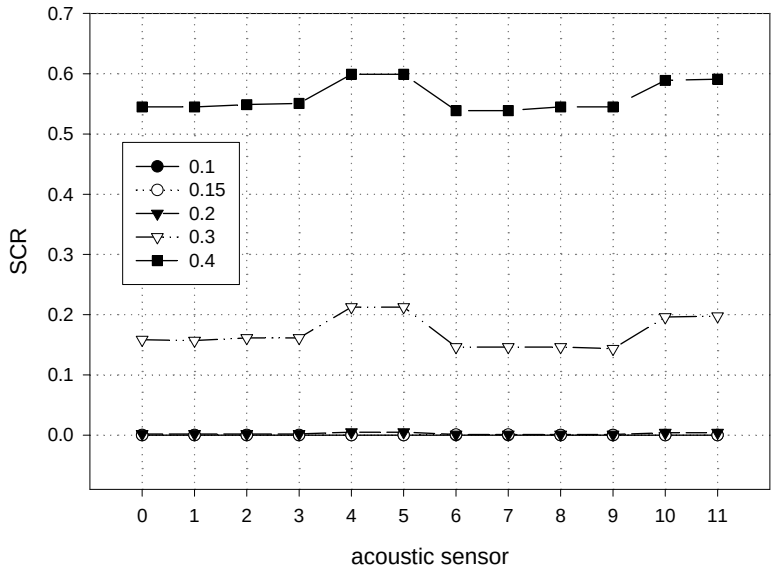


Figure 3-15: SCR results achieved from tree scenario when the image size is 20Kbytes.

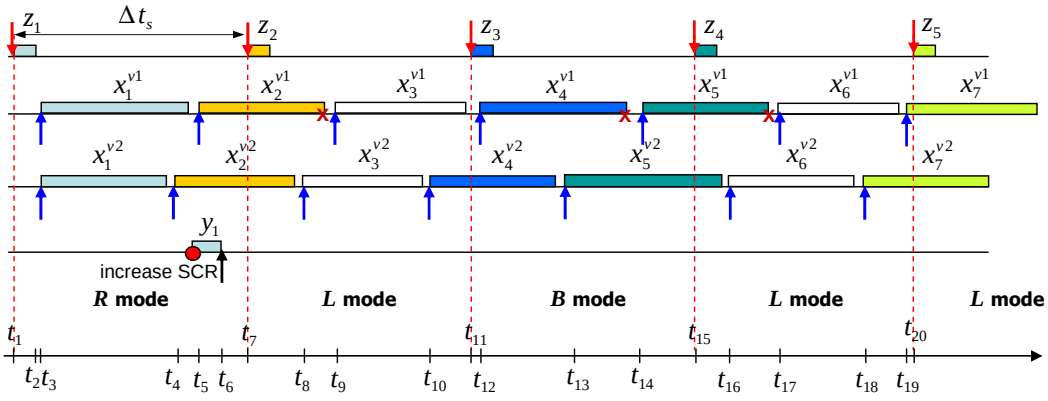
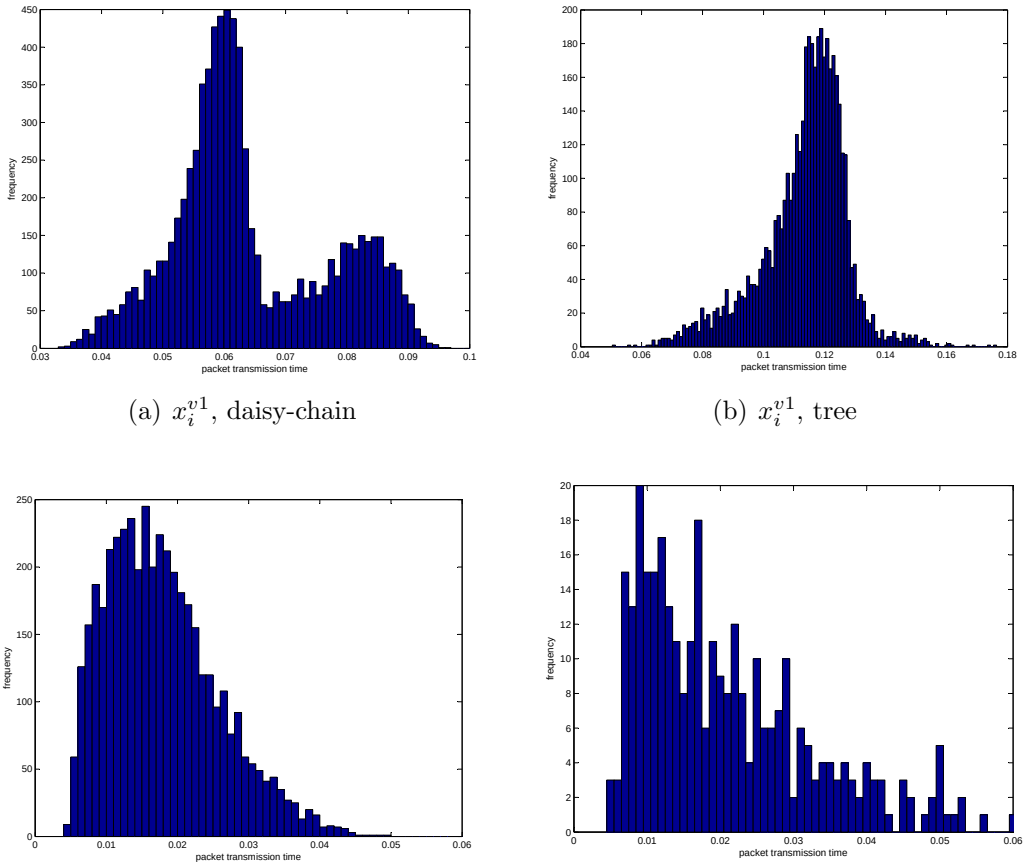


Figure 3-16: Packet flowing example possibly appearing in the tracking system.



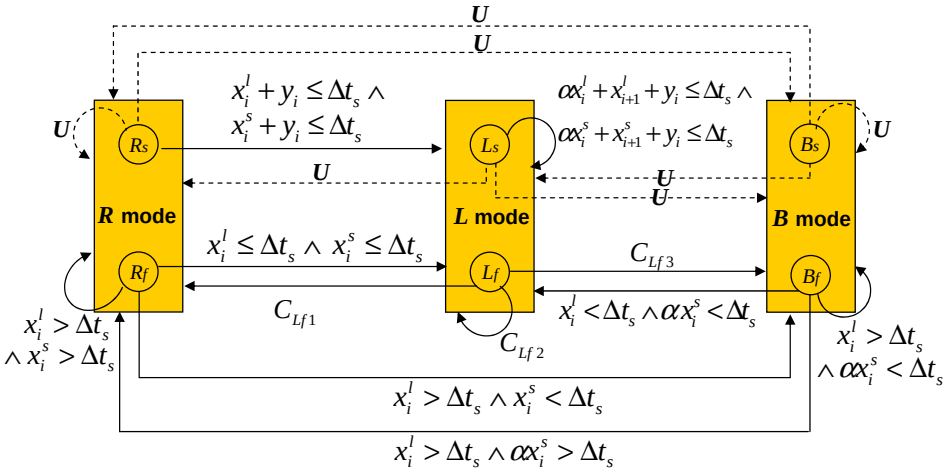
(a) x_i^{v1}, daisy-chain

(b) x_i^{v1}, tree

(c) y_j at acoustic sensor 0, daisy-chain

(d) y_j at acoustic sensor 0, tree

Figure 3-17: Histogram of transmission delay of multi-modal sensor data in daisy-chain and tree scenarios.



$$C_{Lf1} = \left\{ \alpha x_i^l > \Delta t_s \wedge \alpha x_i^s > \Delta t_s \right\}$$

$$\vee \left\{ \begin{array}{l} \left(\alpha x_i^l < \Delta t_s \wedge \alpha x_i^s < \Delta t_s \right) \\ \wedge \left(\alpha x_i^l + x_{i+1}^l + y_j > \Delta t_s \right) \\ \wedge \left(\alpha x_i^l + x_{i+1}^l \le \Delta t_s \right) \\ \wedge \left(\alpha x_i^s + x_{i+1}^s \ge \Delta t_s \right) \end{array} \right\}$$

$$C_{Lf2} = \left(\alpha x_i^l < \Delta t_s \wedge \alpha x_i^s < \Delta t_s \right) \wedge \left(\alpha x_i^l + x_{i+1}^l > \Delta t_s \right) \wedge \left(\alpha x_i^s + x_{i+1}^s > \Delta t_s \right)$$

$$C_{Lf3} = \left\{ \alpha x_i^l > \Delta t_s \wedge \alpha x_i^s < \Delta t_s \right\}$$
$$\vee \left\{ \left(\alpha x_i^l < \Delta t_s \wedge \alpha x_i^s < \Delta t_s \right) \wedge \left(\alpha x_i^l + x_{i+1}^l > \Delta t_s \right) \wedge \left(\alpha x_i^s + x_{i+1}^s \le \Delta t_s \right) \right\}$$
$$\vee \left\{ \begin{array}{l} \left(\alpha x_i^l < \Delta t_s \wedge \alpha x_i^s < \Delta t_s \right) \wedge \left(\alpha x_i^l + x_{i+1}^l \le \Delta t_s \right) \wedge \left(\alpha x_i^l + x_{i+1}^l + y_j > \Delta t_s \right) \\ \wedge \left(\alpha x_i^s + x_{i+1}^s < \Delta t_s \right) \end{array} \right\}$$
$$\vee \left\{ \alpha x_i^l > \Delta t_s \wedge \alpha x_i^s < \Delta t_s \right\}$$
$$\vee \left\{ \alpha x_i^l > \Delta t_s \wedge \alpha x_i^s < \Delta t_s \right\}$$

Figure 3-18: Transition rules of the SEA.

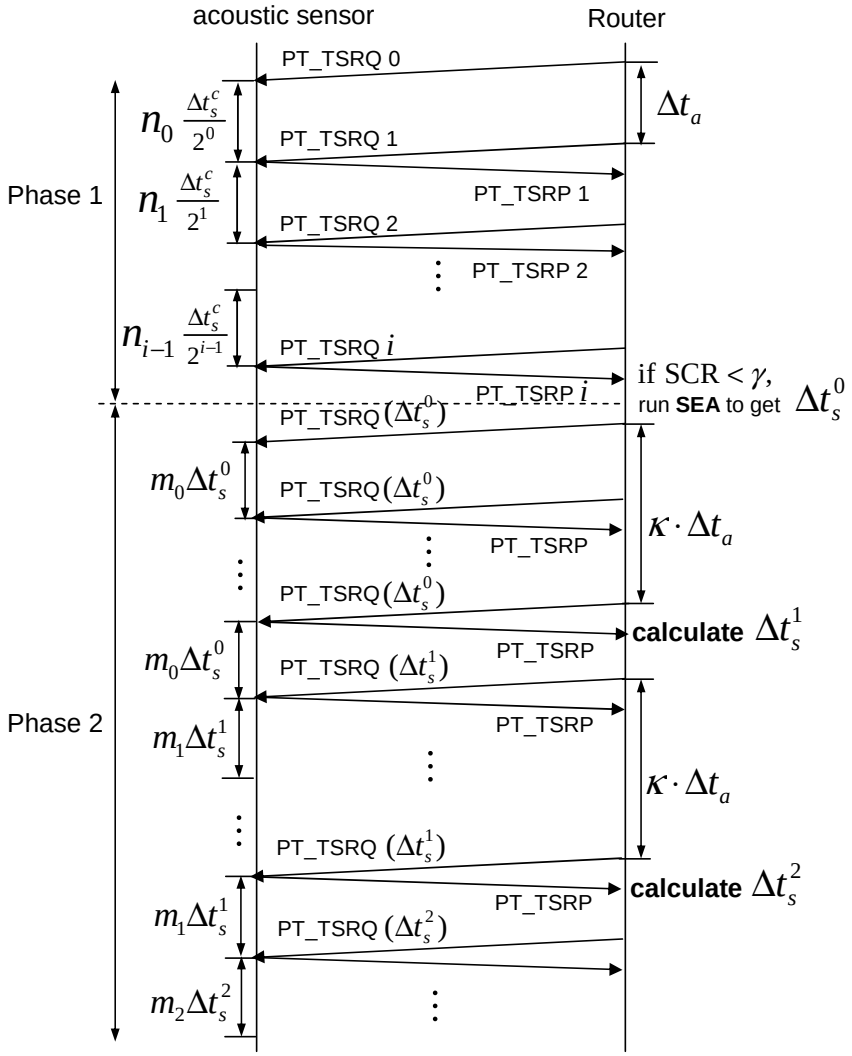
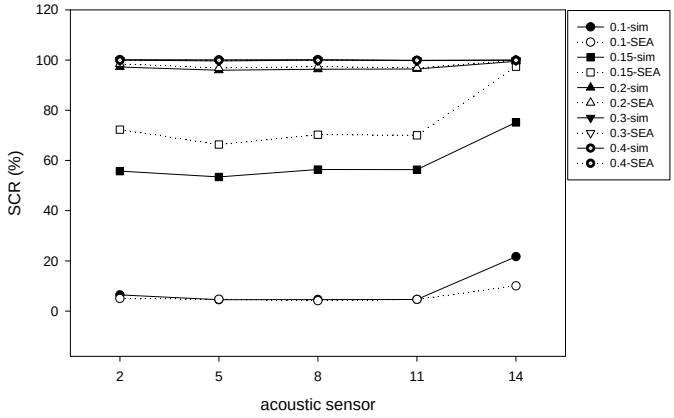
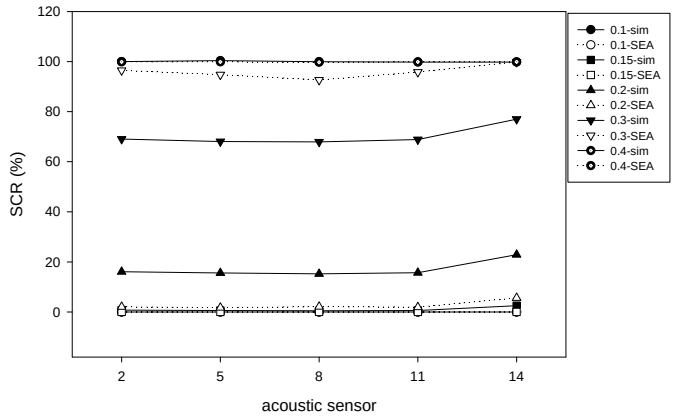


Figure 3-19: Procedural illustration of the SEA^2.

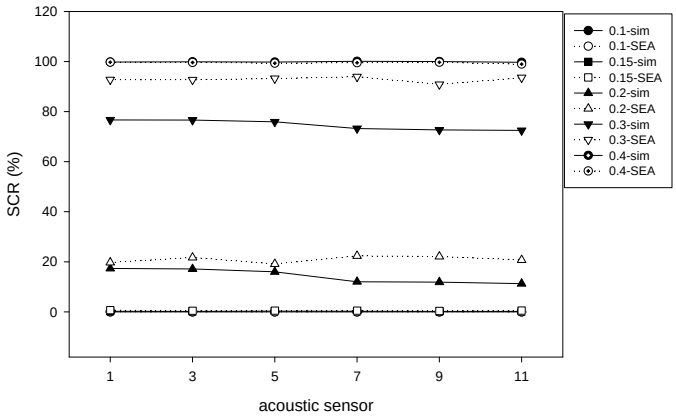


(a) Daisy-chain, 20 KBytes

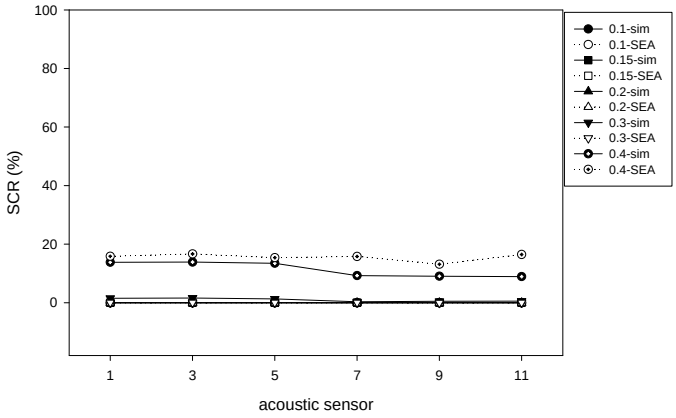


(b) Daisy-chain, 40 KBytes

Figure 3-20: SCR results achieved by simulations and mathematical calculation of the SEA: daisy-chain scenario.

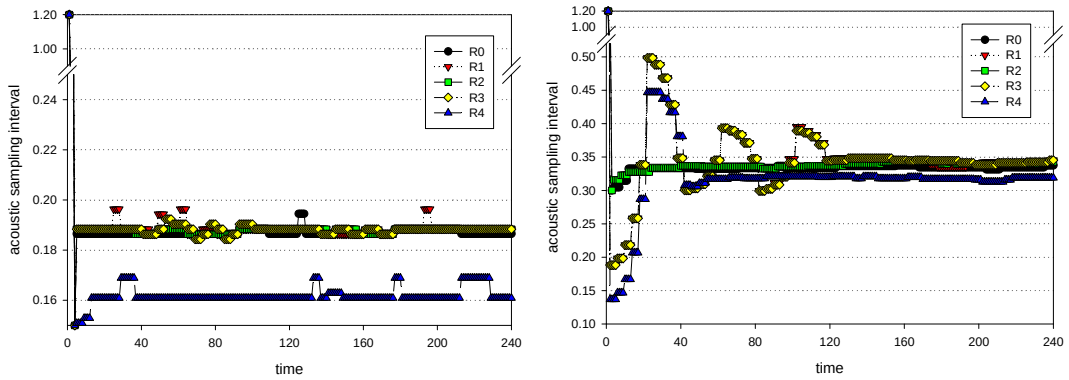


(a) Tree, 20 KBytes



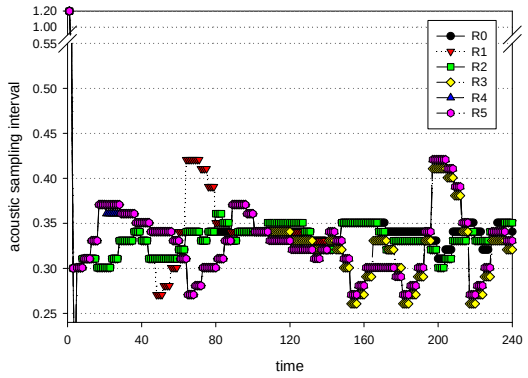
(b) Tree, 40 KBytes

Figure 3-21: SCR results achieved by simulations and mathematical calculation of the SEA: tree scenario.

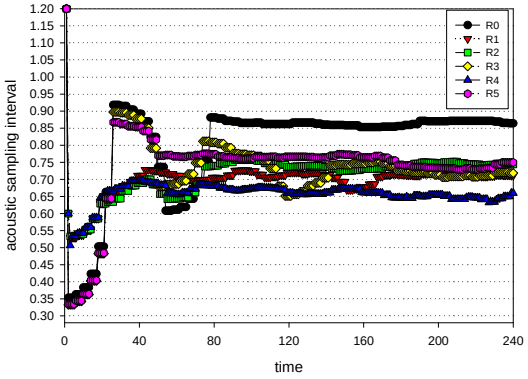


(a) Daisy-chain, 20 KBytes

(b) Daisy-chain, 40 KBytes

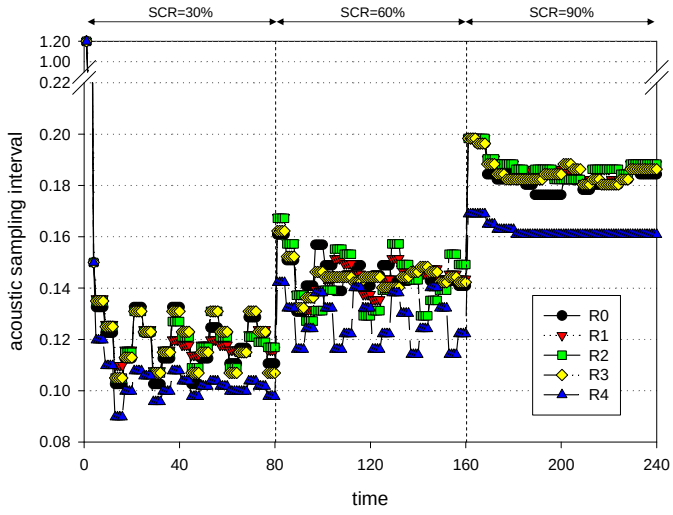


(c) Tree, 20 KBytes

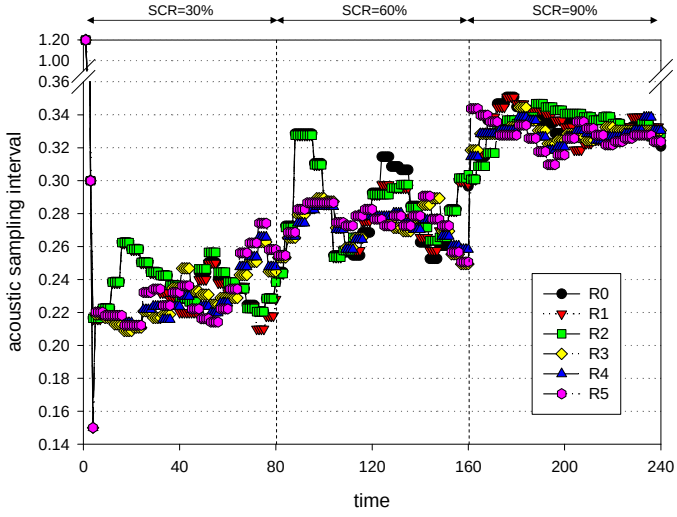


(d) Tree, 40 KBytes

Figure 3-22: Acoustic sampling interval variation in the SEA^2, where t_{scr}=90%.



(a) Daisy-chain, 20 KBytes



(b) Tree, 20 KBytes

Figure 3-23: Acoustic sampling interval variation in the SEA^2. t_{scr} is changed to 30, 60 and 90% during the simulations.

Chapter 4

Conclusion and Future Works

4.1 Conclusion

This dissertation has tackled the design and optimization problems appearing in the node (hardware) architecture design of RObust Header Compression (ROHC) and packet aggregation over Wireless Mesh Networks (WMNs), and the object tracking applications supported by multi-modal sensors.

To design an appropriate node architecture and investigate the impact of single node on the global WMN performance, we have proposed network/hardware co-simulation methodology that integrates the network level simulator, NS-2 and the node level simulator, SystemC. Based on the co-simulation method, we have verified that the hardware realization of the ROHC and packet aggregation algorithms sufficiently provides the high processing power, and resolves the processing overhead problems taking place in the general purpose processors of Pentium 4 and RouterBOARD.

From the evaluation of the co-simulation method, we have shown the hardware encoding and decoding times scatter from a few hundreds nanoseconds to a few tens of thousands nanoseconds, which is different from the estimated constant hardware processing time. We also have shown the sensor throughput and voice quality in the co-simulation method present different results compared with the simulations applied by constant hardware delays. This result tells us how overall WMN performance can be changed by the hardware operations in each mesh router. The numerical analysis model to predict the hardware capacity also has been proposed based on the open Jackson queueing network. In the analysis model, we have derived average system waiting time of compression and decompression paths after obtaining the average system size and average system waiting time for the individual queue. By using the network/hardware co-simulation method, we have shown the proposed numerical model accurately predicts the processing delay of the hardware architecture.

In the object tracking application where an acoustic sensor has a role in the main modality and two visual sensors perform the error correction, we have first discovered the *network synchronization problem* caused by the sampling time difference and randomness property of transmission delay of multi-modal sensor data. For the possible solution of the *network synchronization problem*, we have proposed a traffic differentiation model to differently serve the sensor traffic and non-sensor traffic. The differentiation model serves the traffic types by Weight Round Robin (WRR). The weighting factors of the scheduling mechanism are determined by the proposed Delay-based Weight Allocation (DWA) algorithm. We have shown that the differentiation

model efficiently mitigates the unbalance in transmission delay and supports high level of visual compensation. In the tracking system, the estimation of the *success* in visual compensation has been conducted by the Statistical Estimation Algorithm (SEA). We have shown that the SEA properly approximates the number of *success* by hand calculation without simulation or system setup. Another question in the tracking system could be how to maintain the tracking system at a certain level of tracking accuracy. This question has been answered by the proposed algorithm, Statistical and Estimation and Adaptation Algorithm (SEA^2). From the simulations based on the daisy-chain and tree tracking scenario, we have shown the SEA^2 properly adapts the sampling interval of acoustic sensors to maintain the tracking system at the target accuracy level.

4.2 Future Works

Even if the proposed network/hardware co-simulation method makes it possible to reassess the global wireless network performance in terms of the hardware behaviors in a single node, the method follows the off-line based evaluation procedures. The 3 profiling steps should be manually performed, so that it is not applicable to the large scale network analysis. Moreover, the method only suggests the processing delay as the profiling parameter. Therefore, in the future work, we can build up the on-line version of the co-simulation method where automatic profiling procedures are performed. The possible works could be the integration of the kernels of NS-2 and SystemC, and the developments of the hardware mapping function and hardware module library.

We can also add the trace file management modules to record the various profiling parameters such as power and delay. Since the automatic co-simulation method may require a high processing power to be executed, we can adopt the parallel processing of the co-simulation for the high speed simulations by porting the co-simulation tools to a multi-processor computing environment.

In case of tracking system, we can migrate the simulation works into the real implementation of the tracking network and verify the impact of the network synchronization problem and the correct operations of the SEA and SEA^2. For the possible Router system, we can use the pre-installed RouterBOARD and the Pentium-based Linux could be a good candidate for the Server system.

.

Bibliography

[1] I. F. Akyildiz. "A Survey on Wireless Mesh Networks". *IEEE Radio Communications*, September 2005.

[2] *Network Simulator-2 (NS-2).* http://www.isi.edu/nsnam/ns.

[3] Jinseok Lee, Sangjin Hong, We-Duke Cho. "Enhancing Particle Filtering Performance in Tracking through Visual Information Association". *submitted to EURASIP Journal on Advances in Signal Processing.*

[4] Sangkil Jung, Jinseok Lee, Sangjin Hong, and We-Duke Cho. "On addressing Network Synchronization in Object Tracking with Multi-modal Sensors". *submitted to EURASIP Journal on Advances in SIgnal Processing.*

[5] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. "A Tutorial on Particle Filters for Online Non-linear/Non-gaussian Bayesian Tracking". *IEEE Transactions on Signal Processing*, 50(2):174 – 188, February 2002.

[6] Sangjin Hong, Jinseok Lee, Akshay Athalye, and Petar M. Djuric. "Design Methodology for Domain-Specific Parameterizable Particle Filter Realizations". *IEEE Transactions on Circuits and Systems-I*, 2007.

[7] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, H. Zheng. "Robust header compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3059, July 2001.

[8] *SystemC.* http://www.systemc.org.

[9] *RouterBOARD.* http://www.routerboard.com.

[10] Donald Gross and Carl M. Harris. *"Fundamentals of Queueing Theory"*. John Wiley & Sons.

[11] Sangkil Jung, Sangjin Hong, and Kyungtae Kim. "Network/Hardware Cross-Layer Evaluation for ROHC and Packet Aggregation on Wireless Mesh Network". *submitted to ACM/Springer Wireless Networks.*

[12] Sangkil Jung and Sangjin Hong. “Numerical Analysis of Hardware Architecture for High-Speed and Reliable Data Transmission on Wireless MESH Networks”. *submitted to ACM/Springer Mobile Networks and Applications.*

[13] Sangkil Jung, Sangjin Hong, Peom Park. “Effect of RObust Header Compression (ROHC) and Packet Aggregation on Multi-hop Wireless Mesh Networks”. In *IEEE International Conference on Computer and Information Technology*, September 2006.

[14] Sangkil Jung, Sangjin Hong, and Kyungtae Kim. “Voice Transmission Enhancing Model on Wireless Mesh Networks”. In *IEEE International Conference on Communications*, June 2007.

[15] Sangkil Jung, Sangjin Hong, Kyungtae Kim, Junghoon Jee and Eunah Kim. “On Achieving High Performance Wireless Mesh Network with Data Fusion”. In *IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*, June 2007.

[16] *Powerimpact.* http://ee.ucla.edu/PowerImpact.

[17] David Brooks, Vivek Tiwari, Margaret Martonosi, and Wattch. “A Framework for Architectural-level Power Analysis and Optimizations”. In *International Sypmosium on Computer Architecture*, June 2006.

[18] A. Sinha and A. P. Chandrakasan. “JouleTrack: A Web based Tool for Software Energy Profiling”. In *Design Automation Conference*, 2001.

[19] *GloMoSim.* http://pcl.cs.ucla.edu/projects/glomosim.

[20] *MaRS.* http://www.cs.umd.edu/projects/netcalliper/software.html.

[21] A. Chatelain, Y. Mathys, G. Placido, A. La Rosa, L. Lavagno. “High-Level Architectural co-Simulation Using Esterel and C”. In *International Symposium on Hardware/Software Codesign*, pages 189–194, 2001.

[22] G. Nicolescu, S. Yoo, A. A. Jerraya. “Mixed-Level Cosimulation for Fine Gradual Refinement of Communication in SoC Design”. In *Design, Automation and Test in Europe (DATE)*, pages 754–759, 2001.

[23] H. Park, W. Liao, K.H. Tam, M.B. Srivastava, L. He. “A Unified Network and Node Level Simulation Framework for Wireless Sensor Networks”. Technical Report TR-UCLA-NESL-200309-02, CENS, http://nesl.ee.ucla.edu/document/show/26, September 2003.

[24] V. Aue, J. Kneip, M. Weiss, M. Bolle, and G. Fettweis. “MATLAB based codesign framework for wireless broadbandcommunication DSPs”. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1253–1256, 2001.

[25] *Hardware/Software/Network Co-Design Tool (HSN).* http://eda.sci.univr.it/.

[26] Gallo P. Poncino M. Martini S. Ricciato F. Fummi F., Perbellini G. A. "Timing-accurate Modeling and Simulation Environment for Networked Embedded Systems". In *Design Automation Conference*, pages 42–47, 2003.

[27] V. Jacobson. "Compressing TCP/IP Headers for Low-speed Serial Links", RFC 1144, February 1990.

[28] M. Arnau, A. Calveras, and J. Paradells. "A Controlled Overhead for TCP/IP Header Compression Algorithm over Wireless Links". In *International Conference on Wireless Communications*, 1999.

[29] S. Casner and V. Jacobson. "Compressing IP/UDP/RTP Headers for Low-speed Serial Links", RFC 2508, February 1999.

[30] L. Jonsson, M. Degermark, H. Hannu, and K. Svanbro. "Evaluation of CRTP Performance over Cellular Radio Links". In *IEEE Personal Communications*, 2000.

[31] Stephan Rein, and Frank H.P. Fitzek. "Voice Quality Evaluation for Wireless Transmission with ROHC". In *IASTED conference on Internet and Multimedia Systems and Applications*, 2003.

[32] Ranjani Sridharan. "A Robust Header Compression Technique for Wireless Ad Hoc Networks". In *MobiHoc*, Annapolis, Maryland, USA, June 2003.

[33] A.C. Minaburo, K.D. Singh, L. Toutain, Nuaymi, L. "Proposed Behavior for Robust Header Compression over a radio link". In *IEEE International Conference on Communications*, June 2004.

[34] Kyungtae Kim, Sangjin Hong. "VoMESH: Voice over wireless MESH networksVoMESH: Voice over wireless MESH networks". In *IEEE Wireless Communications and Networking Conference*, 2006.

[35] M. N. D. G. Ashish Jain, Marco Gruteser. "Benefits of packet aggregation in ad-hoc wireless network". Technical Report CU-CS-960-03, Department of Computer Science, University of Colorado at Boulder, 2003.

[36] N. H. Hideaki Yamada. "Voice Quality Evaluation of IP-based Voice Stream Multiplexing Schemes". In *Local Computer Networks*, 2001.

[37] R. Komolafe, O. Gardner. "Aggregation of VoIP Streams in a 3G Mobile Network: A Teletraffic Perspective". In *European Personal Mobile Communications Conference*, 2003.

[38] Y. Lin, Y. Lin, S. Yang, and Y. Lin. "DiffServ over Network Processors: Implementation and Evaluation". In *IEEE Symposium on High Performance Interconnects Hot Interconnects*, 2002.

[39] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss. "An Architecture for Differentiated Services", RFC 2475, December 1998.

[40] E. J. Johnson and A. R. Kunze. *"IXP1200 Programming"*. Intel Press, 2002.

[41] R. Persaud, D. Sabath, G. Berghoff, and R. Schanko. "Performance Evaluation of MPLS and IP on an IXP1200 Network Processor". In *IEEE International Conference on Advanced Information Networking and Applications*, 2006.

[42] David E. Taylor, Andreas Herkersdorf, Andreas Doering, and Gero Dittmann. "Robust Header Compression (ROHC) in Next-Generation Network Processors". *IEEE/ACM Transactions on Networking*, 13(4):755–768, August 2005.

[43] R. Cle and J. Rosenbluth. "Voice over IP performance Monitoring". *ACM Computer Communication Review*, 31, April 2001.

[44] D. Niculescu, S. Ganguly, K. Kim, R. Izmailov. "Performance of VoIP in a 802.11-based Wireless Mesh Network". In *IEEE Infocom*, April 2006.

[45] *ROHC-Robust Header Compression*. http://rohc.sourceforge.net.

[46] *Mica Motes*. http://www.xbow.com.

[47] *Telos Motes*. http://www.moteiv.com.

[48] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. "The click modular router". *ACM Transactions on Computer Systems*, 18, August 2000.

[49] C. H. Knapp and G. C Carter. "The Generalized Correlation Method of Estimation of Time Delay". *submitted to IEEE Trans. on Acoustic, Speech, and Signal Processing*, 4:320–327, 1976.

[50] J. H. Dibiase, H. F. Silverman, and M. S. Brandstein. "Robust Localization in Reverberant Rooms". In *Microphone Arrays : Signal Processing Techiniques and Applications*.

[51] N. Strobel, T. Meier, and R. Rabenstein. "Speaker localization using steered filtered-and-sum beamforemrs". In *Erlangen Workshop on Vision, Modeling, and Visualization*, pages 195–202, 1999.

[52] J. Vermaak and A. Blake. "Nonlinear Filtering for Speaker Tracking in Noisy and Reverberant Environments". In *IEEE International Conference on Acoustic, Speech, Signal Processing*, May 2001.

[53] D. B. Ward and R. C. Williamson. "Particle Filter Beamforming for Acoustic Source Localization in a Reverberant Environment". In *IEEE International Conference on Acoustic, Speech, Signal Processing*, May 2002.

[54] C. Jue, J. P. Le Cadre, and P. Perez. "Sequential Monte carlo Methods for Multiple Target Tracking and Data Fusion". *IEEE Trans. Signal Processing*, 50:309–325, February 2002.

[55] Jinseok Lee, Jaechan Lim, Sangjin Hong, Peom Park. "Tracking an Object in 3-D Space using Particle Filtering based on Sensor Array". In *IEEE International Conference on Computer and Information Technology*, September 2007.

[56] P. M. Djurić and J. H. Kotecha and J. Zhang and Y. Huang and T. Ghirmai and M. F. Bugallo and J. Miguez. "Particle Filtering". *IEEE Signal Processing Magazine*, 20:19–38, September 2003.

[57] A. Doucet, N. de Freitas, and N. Gordon, Eds. *"Sequential Monte Carlo Methods in Practice"*. Springer Verlag, 2001.

[58] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. "A novel approach to nonlinear and non-Gaussian Bayesian state estimation". *IEE Proceedings-F Radar and Signal Processing*, 140(2):107–113, April 1993.

[59] J. Carpenter, P. Clifford, and P. Fearnhead. "An improved particle filter for non-linear problems". *IEE Proceedings-F Radar and Signal Processing*, 146:2–7, 1999.

[60] Ryuzo Okada, Yoshiaki Shirai and Jun Miura. "Object Tracking Based on Optical Flow and Depth". In *IEEE/SICE/RSJ International Conference*, pages 565–571, December 1996.

[61] S. Khan and M. Shah. "Consistent Labeling of Tracked Objects in Multiple Cameras with Overlapping Fields of View". *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(10):1355–1360, October 2003.

[62] A. Bakhtari, M. D. Naish, M. Eskandari, E. A. Croft and B. Benhabib. "Active-Vision based Multisensor Surveillance - An Implementation". *IEEE Trans. on Systems, Man and Cybernetic - Part C : Application and Riviews*, 36(5):668–680, September 2006.

[63] D. N. Zotkin, R. Duraiswami and L. S. Davis. "Joint Audio-Visual Tracking Using Particle Filters". *EURASIP Journal on Applied Signal Processing*, pages 549–552, 2003.

[64] Sangkil Jung, Sangjin Hong, and We-Duke Cho. "Statistical Estimation and Adaptation for VIsual Compensation in Object Tracking". *submitted to International Journal of Distributed Sensor Networks.*

[65] M. S. Arulampalam, B. Ristic, N. Gordon and T. Mansell. "Bearings-only Tracking of Manoeuvring Targets Using Particle Filters". *EURASIP Journal on Applied Signal Processing*, 2004.

[66] Y. Boers and J. N. Driessen. "Interacting Multiple Model Particle Filter". *IEE Proceedings - Radar, Sonar and Navigation*, 150:344–349, October 2003.

[67] A. S. Chhetri, D. Morrell and A. P. Suppappla. "Scheduling Multiple Sensors using Particle Filters in Target Tracking". In *IEEE Workshop on Statistical Signal Processing*, pages 549–552, September 2003.

[68] J. Lee, S. Hong, P. Park, W. D. Cho. "Object Tracking Based on RFID Coverage and Visual Compensation in Wireless Sensor Network". In *IEEE International Symposium on Circuits and System*, 2007.

[69] M. Stanacevic, G. Cauwenberghs. "Micropower Gradient Flow acoustic Localizer". In *Solid-State Circuits Conference*, pages 69–72, 2003.

[70] Jinseok Lee, Sangkil Jung, Yuntai Kyong, Xi Deng, Sangjin Hong, and We-Duke Cho. "Data Traffic Analysis in Wireless Fusion Network with Multiple Sensors". In *IEEE International Midwest Symposium on Circuits and Systems*, August 2007.

[71] *4XEM PTZ Pan/Tilt/Zoom IP Network Camera.* http://www.4xem.com/products/wired/IPCAMWPTZ/index.html.

[72] A. Leon-Garcia. *"Probability and Random Processes for Electrical Engineering"*. Addision Wesley, 2001.