Advanced scheduling in data transmission and mobile media cloud computing

Yang, Ming

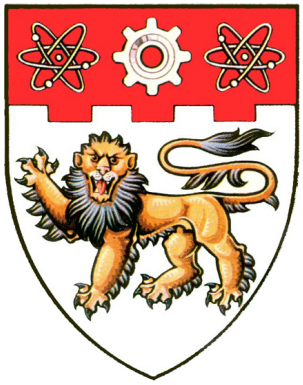
2015

Yang, M. (2015). Advanced scheduling in data transmission and mobile media cloud computing. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/62259

https://doi.org/10.32657/10356/62259

Advanced Scheduling in Data Transmission and Mobile Media Cloud Computing



YANG Ming

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2015

Acknowledgments

I would like to give my sincere acknowledgement to my supervisor Assoc. Prof. Cai Jianfei and my co-supervisor Dr. Foh Chuan Heng, for dedicating their knowledge, encouragement and support in guidance of my research work.

I would also like to thank the members of my PhD dissertation examination committee for their valuable time and advice.

Finally, I would express my wholehearted gratitude towards my parents and my friends for their dedication and love throughout my life.

Contents

Acknowledgments ii

List of Figures vii

List of Tables ix

List of Abbreviations x

Abstract xi

1 Introduction **1**

1.1 Background 1

1.2 Motivation and Scope 3

1.3 Major Contributions 5

1.4 Thesis Organization 7

2 Literature Review **8**

2.1 Efficient Data Transmission Scheduling in Long-RTT Low-bandwidth Wireless Network 8

2.1.1 Design Problems in Underwater Acoustic Networks 8

2.1.2 Related MAC Protocol Design for UANs 10

2.2 Energy Efficient Media Computing on Mobile Devices 12

2.2.1 Feasible Solutions to Energy Saving 13

2.2.2 Dynamic CPU Speed Scheduling 14

2.3 Media Computing Scheduling in Cloud 15
2.3.1 Media Cloud Frameworks 15
2.3.2 Video Transcoding Job Dispatch 17

3 Hybrid Collision-Free Medium Access Protocol for UANs 18

3.1 Overview of The JSW Transmission Scheme 19
3.2 The HCFMA Protocol 22
3.2.1 Multi-channel Setup 22
3.2.2 The Channel Reservation Procedure 24
3.2.3 The Data Transmission Procedure 28
3.2.4 The Joining and Leaving Procedures 29
3.2.5 The Error Recovery Procedure 30
3.2.6 The Initialization Procedure 31
3.3 Throughput Performance Analysis 31
3.4 Simulation Results and Discussions 33
3.4.1 Simulation Setting 34
3.4.2 Worst-Case Performance Comparisons 35
3.4.3 Performance Comparisons in General Network Scenarios 36
3.4.4 HCFMA Throughput and Delay Tradeoff 39
3.4.5 Performance Comparisons in Light Load Scenario 40
3.5 Summary 42

4 Energy Minimization via DVS for Real-Time Mobile Video Encoding 44

4.1 Model and Formulation 45
4.1.1 Energy Consumption Model for Mobile Devices 45

4.1.2 Probabilistic Workload Model for Video Encoding 46
4.1.3 Problem Formulation 46
4.2 Optimal DVS scheduling 48
4.2.1 Derivation of Optimal DVS policy 48
4.2.2 Optimal DVS Scheduling Characteristics 50
4.3 DVS Application to Real-Time Video Encoding 54
4.3.1 Empirical Workload Distribution for H.264/AVC Coding 54
4.3.2 Optimal Clock Frequency Configuration and Energy Consumption 56
4.4 Summary 59

5 Adaptive Job Scheduling for Cloud-based Video Transcoding 60

5.1 System Models 61
5.1.1 System Overview 61
5.1.2 Job Arrival and Dispatch 62
5.1.3 Job Adaptive Configuration 62
5.1.4 Transcoding QoS and Resource Consumption 64
5.2 Problem Formulation and Solution 64
5.2.1 Optimization Problem 65
5.2.2 Job Scheduling Solution 66
5.3 Simulation 67
5.3.1 Real Trace 69
5.3.2 Simulation Setup 70
5.3.3 Result Analysis 70
5.4 Summary 73

6 Conclusion and Future Works **74**

6.1 Conclusion 74

6.2 Future Directions 75

6.2.1 Extension of Scheduling in Data Transmission 75

6.2.2 Extension of Scheduling in Mobile Media Cloud Computing . . . 76

References **80**

List of Figures

3.1 The packet transmission process of JSW scheme. [1] 20

3.2 Timing diagram of multi-channel setup. 24

3.3 The finite state machine of the HCFMA protocol. 24

3.4 An example of channel reservation on CCH. 26

3.5 Throughput versus offered load in the worst case for $N_{dch=3}$. 34

3.6 Throughput versus offered load for $N_{dch=3}$. 36

3.7 Throughput versus offered load for different N_{dch} for T_{jsw}. 37

3.8 Delay versus offered load for different N_{dch}. 38

3.9 Throughput versus offered load for different N_g. 39

3.10 Delay versus offered load for different N_g. 40

3.11 Throughput versus offered load for light load scenario. 41

3.12 Delay versus offered load for light load scenario. 42

4.1 The energy saving improvement compared to flat scheduling. $\eta = \sigma/\mu$, and η_1, η_2, η_3 are respectively $0.1, 0.12, 0.14$. 51

4.2 The optimal clock frequency scheduling. The mean of each workload are respectively $\mu_1 = 3Bc$, $\mu_2 = 3.3Bc$, $\mu_3 = 3.6Bc$, while standard derivation is fixed to $\sigma = 0.3Bc$, and deadline is $T = 0.5s$. (Bc denotes Billion cycle) 51

4.3 The optimal clock frequency scheduling. The standard derivation of each workload are respectively $\sigma_1 = 0.3Bc$, $\sigma_2 = 0.36Bc$, $\sigma_3 = 0.42Bc$, while mean of cycles is fixed to $\mu = 3Bc$, and deadline is $T = 0.5s$. 53

4.4 Per-GOP CPU cycle consumption histogram and normal distribution CDF fitting for four different videos 55

4.5 Cumulative energy consumption for three alterative DVS policies 57

5.1 The cloud transcoding system framework 61

5.2 Transcoding performance statistics of computing time and generated bit-rate under three presets (superfast, faster and medium). The box is the range of estimation, where the horizontal line within the box is the median value. The horizontal lines of upper and lower outside the box are the boundary value, while the 'plus' symbol denotes the out of range result values. 68

5.3 Performance comparison of static scheduling and adaptive scheduling . . 71

5.4 Backlog-Bitrate performance for parameter V 71

5.5 Performance with different system loads 72

List of Tables

3.1 Notations used in the JSW scheme 21

3.2 Notations used in the HCFMA operation 23

4.1 Energy Saving Over Brute-Force Algorithm 57

4.2 Energy consumption gap compared to Gene-Aided DVS 58

5.1 The statistics of transcoding performance 68

List of Abbreviations

ARQ	Automatic Repeat reQuest
$CCDF$	Complementary Cumulative Distribution Function
CDF	Cumulative Distribution Function
CTS	Clear To Send
$CSMA$	Carrier Sensing Multiple Access
$EC2$	Elastic Compute Cloud
$FAMA$	Floor Acquisition Multiple Access
GOP	Group of Pictures
$HCFMA$	Hybrid Collision-Free Media Access
JSW	Juggling-like Stop-and-Wait
MAC	Media Access Control
QoS	Quality of Service
RTS	Request To Send
RTT	Round Trip Time
$SFAMA$	Slotted Floor Acquisition Multiple Access
SVC	Scalable Video Coding
$UANs$	Underwater Acoustic Networks
VM	Virtual Machine

Abstract

In general, scheduling can be considered as the way to efficiently arrange the usage of some shared resource (e.g. communications bandwidth and computing time) to complete multiple tasks or sub-tasks. In this thesis, three types of scheduling are investigated, including the data transmission scheduling in wireless MAC protocol, the CPU scheduling for media computing on mobile devices and the dynamic job scheduling in cloud system.

In the data transmission scheduling, we study how to efficiently schedule a large amount of data transmission jobs in the resource constrained scenarios, particularly in the long-RTT (round-trip time) and low-bandwidth networks scenario. The MAC protocol of underwater networks is chosen as a study case to investigate how to overcome the limited resource. A new underwater MAC protocol is proposed where the design tackles the costly protocol handshakes by reducing the protocol handshake overhead and increasing the data transmission opportunities. An efficient dynamic polling-based handshake operation is designed which offers time-bounded collision-free channel assignment. Through extensive simulations, the results show the proposed HCFMA protocol outperforms its counterparts, in terms of throughput and delay, such as ALOHA with carrier sensing (ALOHA-CS) and slotted floor acquisition multiple access (SFAMA).

Then we study the energy efficient scheduling for media computing on mobile devices. Specifically we take video encoding, a CPU intensive application, as the study case. The CPU running frequency on video encoding process is scheduled to minimize the total expected energy consumption, while meeting the requirements of quality-of-service (QoS). A probabilistic QoS model is adopted, in which the encoding process should complete with a target probability within a specified delay deadline for each GOP (Group of Pictures). The optimization problem is solved analytically and closed-form solutions are obtained

for both the optimal clock frequency scheduling and the minimum energy consumption. The numerical results suggest that significant amount of energy can be saved by using our optimal solution.

Finally, we focus on the job scheduling for media computing in cloud system. Taking video transcoding as the study case, a dynamical video configuration scheduler is proposed to minimize QoS degradation while satisfying the criteria of job completion delay. The trade-off between job compeletion delay and QoS of video transcoding is explored in cloud system. The Lyapunov optimization framework is applied as the problem solver for the scheduler to adaptively set up the video transcoding configurations. Compared to the static configuration strategy, the proposed framework obtains smooth transcoding QoS degradation when system load becomes heavier and also achieves better delay performance.

In conclusion, in this thesis we study the advanced scheduling for wireless data transmission, particularly in efficient MAC protocol design of underwater acoustic network. We also study the scheduling for cloud media computing, particularly in energy efficient video coding on mobile device and optimized video transcoding job scheduling on cloud system.

Chapter 1

Introduction

1.1 Background

Scheduling in general can be considered as the way to efficiently arrange the usage of some shared resource (e.g. communications bandwidth, computing time or storage space) to complete multiple tasks or sub-tasks [2]. In the past decades, to achieve various target quality of service (QoS) in the computing, many types of scheduling have been investigated, such as the scheduling of data transmission in wireless mobile networks, the scheduling of threads or running processes in operating systems and the distributed resource and workload scheduling in cloud systems, etc.

In data transmission field, the scheduling in different networks, e.g. wireless mobile networks, sensor networks or content delivery networks, plays a critical role in improving the system performance. Furthermore, the scheduling strategy can be applied at each of the network protocol layers, which provides the flexibility for system design and optimization. Although high bandwidth data transmissions services [3] have become more popular with the development of Internet and the emerging of new generation of wireless technologies such as 3GPP Long Term Evolution (LTE) [4], scheduling is still needed due to the increase of bandwidth-demanding applications such as multimedia streaming.

Moreover, there still exist some resource-limited networks with long-RTT (round-trip time) and low-bandwidth such as satellite networks and underwater networks. How to efficiently schedule data transmission jobs in these networks is extremely challenging. Many existing works have been proposed to optimize the transmission scheduling strategy, e.g. using network coding to avoid data retransmission in long-RTT networks and reducing the wireless channel competition cost to improve the bandwidth utilization, etc.

Besides the data transmission, the computing scheduling on mobile devices is also important, which not only improves the computing performance but also reduces the resource consumption. Considering the trend of rapid increase of mobile CPU performance and the rich media related applications, it is urgent to enhance the computing scheduling for the high energy consumption applications on mobile devices, due to the battery limitation. The traditional solution is to restrict the application running complexity or the computing speed. On the other hand, without QoS degradation, some existing works have tried to develop the advanced scheduling strategy for rich media computing application, through the configuration adaptation in video coding process and the dynamic CPU frequency scheduling on mobile devices, etc.

By leveraging the prevailing cloud computing technologies [5], which offer massive high performance and scalable computing capability, the experience of rich media computing on mobile devices can also be improved. The provided services include elastic networking, computing and storage over the Internet, which will enhance the service provisioning on mobile multimedia applications [6, 7]. The services can be grouped into three categories, software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) from top to down [8]. IaaS refers to the on-demand infrastructual resources provisioning, such as the famous Amazon Elastic Compute Cloud (Amazon EC2) [9]. PaaS mostly focuses on providing a development platforms to application developers, such as Google App Engine [10] and Heroku [11]. SaaS refers to provide practice

applications to end users, such as the Amazon Elastic Transcoder and Zencoder [12] both of which support elastic video transcoding service. Some existing works have focused on the mobile media cloud system development, including cloud video streaming and coding frameworks and cloud-based online game system, etc. The efficient job scheduling in a cloud system is critical which helps improve system performance and reduces resource cost. Some scheduling related topics for cloud computing have recently been studied, such as resource allocation for job dispatching, job adaptation, auto-scaling computing capacity, etc.

1.2 Motivation and Scope

To improve the QoS in data transmission and mobile media cloud computing, we particularly study the scheduling strategy on three topics: wireless data transmission scheduling, media computing scheduling on mobile devices and distributed media computing scheduling in a cloud system.

First, it is challenging to provide robust and fast wireless communications in the long-round trip time (RTT) and bandwidth limited wireless networks, such as satellite networks and underwater acoustic networks (UANs) [13, 14], where most of traditional wireless network protocols are not applicable [15]. To significantly improve the communication performance, it is worth to design new scheduling technologies for the data transmission in long-RTT and low-bandwidth wireless networks. The improved scheduling strategy can be designed at different layers of the protocol stack [16]. For example, by using forward error correction (FEC) in the application layers and applying network coding technologies in the transport layer, the probability of packet retransmission could be reduced significantly in some scenarios. In this thesis, we focus on the data transmission scheduling in the medium access control (MAC) layer which is a sub-layer of the

link layer. Furthermore, we take the underwater acoustic networks as a study case of the long-RTT and low-bandwidth networks. By exploring the characteristics of UANs communication environment, a MAC protocol for single hop UANs is designed. The performance of the proposed MAC protocol will be evaluated with a discrete event simulator built for UANs.

For media computing scheduling on mobile devices, considering the high computing complexity of media processing, it is challenging to overcome the inherent resource constraint on mobile devices [17]. Especially, the energy supply on mobile devices is limited by the physical size of the battery that does not have much room to grow despite high demand [18]. Except for the screen, the major energy consumption of a mobile device is on two parts, CPU and wireless modules. For CPU intensive applications such as video encoding and 3D game processing, energy-efficient computing scheduling will lead to a longer battery life-time [19]. However, different from the general computing task scheduling, media computing scheduling needs to analyze the energy consumption pattern for various media applications. The model derivation on media QoS and its energy consumption is not a trivial work. In this thesis, we design an energy efficient scheduling strategy on mobile media computing, which takes video encoding as the study application. Specifically we aim to schedule the CPU running frequency on video encoding process to minimize the expected energy consumption under the QoS constraint.

In addition to scheduling the media computing process on mobile device, to reduce the energy consumption, with the assistant of cloud computing, some CPU intensive and big data based workloads can be offloaded to the cloud system. The emerging cloud computing provides a promising energy-efficient and capability-scalable solution for media-rich mobile applications [20]. By offloading the computing jobs to cloud, the mobile applications could enjoy the improved QoS while the battery life is prolonged simultaneously.

By renting more virtual machine (VM) from the cloud infrastructure providers, the system computing capacity can scale up quickly to sustain more media processing requests. However, considering the media dynamics on resource consumption and QoS criteria, scheduling the huge number of computing jobs in cloud-based distributed environment is not easy, especially in cloud resource constrained scenario. Taking video transcoding as a study case, due to the job arrival dynamics and video content inherent dynamics, it is challenging to design an optimal scheduling algorithm which can maximize the QoS under constraints of limited computing capacity and job completion delay. In this thesis, we study the trade-off between job completion delay and transcoding QoS in cloud system, and design an efficient scheduling algorithm to dynamically set up the video transcoding configurations.

1.3 Major Contributions

According to the major problems mentioned in Section 1.2, this thesis makes the following major contributions:

- *Hybrid Collision-Free Medium Access Protocol for UANs:* An underwater MAC protocol is proposed where the design tackles the costly protocol handshakes by reducing the protocol handshake overhead and increasing the data transmission opportunities. An efficient dynamic polling-based handshake operation is designed which offers time-bounded collision-free channel assignment. Multi-channel technique is applied to separate the channel assignment protocol handshake and data transmission on different channels, where the channels bandwidth allocation is optimized. The data packets are transmitted under packet train with juggling-like stop-and-wait (JSW) ARQ scheme to deal with channel errors while maximizing channel utilization. Finnally we analyze throughput performance of HCFMA under

both *unsaturated* and *saturated* situations. Extensive simulations are also carried out for different system parameters. The performance of the HCFMA protocol outperforms its counterparts, in terms of throughput and delay, such as ALOHA with carrier sensing (ALOHA-CS) [21] and slotted floor acquisition multiple access (SFAMA) [22].

- *Energy Minimization via Dynamic Voltage Scaling for Real-Time Video Encoding:* A systematic approach is taken to investigate the problem of how to dynamically reconfigure the clock frequency in the mobile device to minimize the energy consumption, while respecting the QoS requirement. The optimal clock-scheduling problem is formulated as a constrained optimization problem. The optimization problem is analytically solved and the closed-form solutions are obtained for both the optimal clock frequency schedule and the minimum energy consumption. The lightweight clock scheduling algorithm is applied to real-time H.264/AVC encoding application on mobile devices. The numerical results suggest that significant amount of energy can be saved by using our optimal solution.

- *Adaptive Job scheduling for Cloud-based Video Transcoding:* A comprehensive study on the trade-off between transcoding delay and video QoS is conducted. The Lyapunov optimization framework [23] is applied as the problem solver to adaptively set up the transcoding job to guarantee the queue backlog bounded. The trade-off of job completion delay and QoS of video transcoding is studied. Compared to the static configuration strategy, the proposed framework obtains smooth coding QoS degradation when system load becomes heavy and reports better delay performance.

In this thesis, the three major contributions address different issues, however they are connected within a mobile media cloud computing ecosystem. There are energy efficient

media computing on mobile side and optimized media transcoding job scheduling on cloud side, while data transmission scheduling acts as a bridge to connect the mobile client and cloud service provider.

1.4 Thesis Organization

The rest of the thesis is organized as follows:

- **Chapter 2** gives a literature review on the related topics. Three major research problems and related works are investigated.
- **Chapter 3** presents underwater networks MAC protocol design as a study case for reliable and efficient data transmission in long-RTT low bandwidth scenario.
- **Chapter 4** presents an energy efficient CPU executing scheduling on mobile device. The dynamic voltage scheduling algorithm is studied to minimize the expected energy consumption.
- **Chapter 5** presents a dynamic cloud video transcoding job scheduling algorithm which trades off between job delay and job QoS. The algorithm minimizes the time-average bit-rate of the generated video stream, while keeping delay requirement.
- **Chapter 6** concludes the thesis and discuss our future research directions.

Chapter 2

Literature Review

2.1 Efficient Data Transmission Scheduling in Long-RTT Low-bandwidth Wireless Network

Here we take underwater networks MAC protocol as a study case to investigate how to overcome the drawbacks of long-RTT and low bandwidth wireless communication scenario. In the following, the key problems for the communication and MAC layer design in underwater environment will be discussed.

In the underwater acoustic networks field, wireless network protocol research has received much attention, due to many potential applications ranging from oceanographic information gathering, environmental monitoring, tsunami alarm and navigation assistance to coastal defense [24]. To make these applications practical, efficient network protocols are needed.

2.1.1 Design Problems in Underwater Acoustic Networks

In the long RTT and bandwidth limited wireless networks, (satellite networks and underwater acoustic networks), rather than signal interference, absorption, reflection and multipath fading, the problem of hidden terminal [25, 15] and channel competition [26]

becomes the major issues in the protocol design [27]. To ensure the reliable and fast data transmission, innovative network protocol design is required to overcome the disadvantage of long RTT and bandwidth limited channel.

The objective of a MAC protocol is to properly allocate channels which are shared by many nodes so as to avoid collisions while maintaining reliable transmission. The MAC protocols for terrestrial wireless environment have been studied extensively over the past decades [28, 29, 30, 31, 32]. While underwater environment shares several similarity to the terrestrial wireless environment, some key differences in the medium characteristic have made MAC protocols for terrestrial wireless networks unsuitable to operate in underwater environment.

In underwater environments, the propagation speed of acoustic waves is approximately 1500 m/s, which is five orders of magnitude lower than the speed of radio waves. The low propagation speed results in a significantly large propagation delay. The available bandwidth for an underwater acoustic channel is very limited, which is typically on the order of 10 kHz around a center frequency with the order of 20 kHz for a system operating over few kilometers. Besides, the nature of multipath propagation and rapid time variations of underwater acoustic channels also induce high bit error rates. As a result, these unfavorable conditions make the design of the MAC protocols for UANs a challenging proposition.

The previously proposed underwater MAC protocols found in the literature can be typically divided into two categories, namely, centralized and distributed protocols. For centralized underwater MAC protocols (see [33, 34] for examples), a special device is required to serve as a central controller to manage channel assignment. This requirement often suffers from single point of failure, network scalability limit, and high overhead, especially under low offered loads. As opposed to centralized underwater MAC protocols,

the distributed approach appears to be attractive given its robustness and network scalability. Currently, distributed underwater MAC protocols often employ random access strategies. However, a pure random access strategy such as the ALOHA protocol [14] leads to a significantly high number of transmission collisions due to the lengthy signal propagation time with respect to the transmission time, which results in long vulnerability periods.

It was shown that introducing channel reservation into the pure random access strategy helps to reduce collisions and thus improves the channel utilization [35]. With this approach, additional protocol handshakes such as Ready-To-Send/Clear-To-Send (RTS/CTS) control message exchange are used to achieve channel reservation for collision-free transmissions of data packets. Even though data transmissions are free of collision, the protocol handshakes in these reservation-based MAC protocols still remain contention-based which poses challenges in achieving efficient operation. Specifically, due to the lengthy signal propagation delay in the underwater environment, frequent collisions occurring during the handshake period significantly affect the throughput performance [36, 22]. Besides, allowing only one data packet to be transmitted for every successful handshake is costly. With a data transmission overwhelmed by protocol handshakes, the channel utilization remains low. Moreover, the effect of high error rate further degrades the performances. To design efficient underwater MAC protocols, a scheme that explicitly deals with the lengthy signal propagation delays must be sought.

2.1.2 Related MAC Protocol Design for UANs

The earliest underwater MAC proposals often follow basic methods such as ALOHA and time division multiple access (TDMA) with certain customizations for the underwater environment. A typical example that evolves from ALOHA is the propagation-delay-tolerant collision avoidance protocol (PCAP) proposed in [35]. PCAP improves ALOHA

by introducing protocol handshakes for channel reservation with precise timing control, so as to keep the handshaking duration constant. Although this scheme can offer 20% more throughput than the conventional handshaking protocols for UANs, it has limited space for further throughput improvement. In [34], a TDMA based multi-cluster MAC protocol was proposed for autonomous underwater vehicles (AUVs) using a fixed allocation mechanism, where within each cluster, TDMA is used with long guard time to overcome the effects of propagation delay. Interference among different clusters is minimized by assigning different spreading codes to different clusters. However, large overheads are needed for cluster control and management, which significantly impacts the throughput performance and scalability.

In order to enhance channel utilization and network energy efficiency, Kredo *et al.* [37] developed a hybrid MAC protocol by combining TDMA with random access. In particular, each frame consists of a TDMA portion followed by a portion that allows nodes to contend for channel access with the aid of overheard network state distribution. It is shown that the combination of TDMA and random access leads to better throughput performance compared to earlier designs. However, the use of a fixed TDMA allocation in the scheme limits the achievable throughput.

Following the earlier works which indicate the potential of random access for underwater MAC protocol design, most recent works focus on improving the protocol handshakes for channel reservation with consideration of transmission synchronization. For example, Molins *et al.* [22] introduced the SFAMA protocol using a 4-way (RTS/CTS/DATA/ACK) handshake with synchronization. However, frequent collisions still take place during the handshaking period, which results in low throughput. The collisions are particularly serious for high traffic loads.

To eliminate potential collisions during data transmission, Tone Lohi (T-Lohi, "Lohi" means slow in Hawaiian) [38] introduces a tone-based contention resolution mechanism

which uses additional hardware for busy tone transmission and detection to avoid collisions. With additional busy tone devices and the requirement of channel synchronization, throughput of around 0.35 is reported in a network with 8 nodes [38].

By exploiting the multi-channel technique, Zhou *et al.* [39] proposed a cooperative multi-channel MAC (CUMAC) scheme for multi-hop UANs with the help of additional busy tone devices. CUMAC attempts to solve triple hidden terminal problems, i.e., multi-channel, long-delay and multi-hop hidden terminal problems, by employing a channel reservation phase which is fulfilled by a 3-way (RTS/Beacon/CTS) handshake on the control channel, and a data transmission phase on the data channel. However, costly and frequent collisions remain, which limits the throughput performance.

It has been shown that collisions are highly costly in UANs. All the existing methods except pure TDMA cannot guarantee collision-free operations in either channel reservation or data transmission. In the following, we shall describe our proposed HCFMA protocol, which can achieve collision-free operations in both channel reservation and data transmission in normal operation.

2.2 Energy Efficient Media Computing on Mobile Devices

Due the inherent resource-constrained drawback, the energy supply on mobile devices is limited by the physical size of the battery. As a result, the energy efficient design on operation system of smart-phone and applications have been shown to be the most important factor affecting the user experience [18]. Except for the back-light of screen, most of energy on smart-phone is consumed by chipsets (e.g. CPU and GPU) and wireless communication modules (e.g. WiFi, 3G and 4G) [40].

2.2.1 Feasible Solutions to Energy Saving

Previous researchers have investigated the problem of energy-aware media computing from a perspective of application interior algorithm design [41, 42]. Considering video coding related applications, in [43], a comprehensive power-rate-distortion model was developed to describe the relationship of different encoding modules for the general video coding structure. In [44], a joint complexity-distortion optimization approach was proposed for real-time H.264 video encoding under power-constraint, in which computational resource is dynamically allocated to frames and Macro-Blocks. The proposed system needs to dynamically allocate resource to motion estimation and mode decision modules and configure the two modules to utilize the resource. However, these encoder-centric approaches often resulted in algorithms that are complicated, rendering its applicability to resource-poor mobile devices to a limited level.

To reduce the energy consumption on mobile devices, two potential ways will be studied in this section. One is to limit the workload executed and application executing speed. For media computing, the workload is associated with media content and computing algorithm. Due to the natural dynamics of media content, workload controlling is achieved by adjusting the applied media computing algorithms or media quality setups. Controlling application executing speed is commonly implemented through chipsets speed adjustment or the downgrading of QoS. The dynamics voltage scheduling technology will be investigated as the guideline of energy efficient algorithm design on CPU executing scheduling. An alternative way is to offload some works to the advance cloud infrastructures to achieve energy saving gains and potential QoS improvement. Such computing workloads can be offloaded [45, 46] to nearby resource-rich cloud servers (Cloudlets [47], Clonecloud [48]). However existing research shows that not all workload is suitable for offloading to save energy, since mobile cloud computing would take energy cost on com-

munication into consideration, which is significantly different from cloud computing for broadband desktop PC [49].

2.2.2 Dynamic CPU Speed Scheduling

In CMOS circuits [50], the energy per operation $\mathcal{E}_{op}$ is proportional to V^2, where V is the supply voltage to the chip. Moreover, it has been observed that the clock frequency of the chip, f, is approximately linearly proportional to the voltage supply of V [50]. Therefore, the energy per operation can be expressed as, $\mathcal{E}_{op} = \kappa f^2$, where κ is the energy coefficient depending on the chip architecture. Note that CPU can reduce its energy consumption substantially by running more slowly [51]. However, for real-time video encoding, the encoder has to meet a delay deadline for each group of pictures (GOP), which suggests that the clock frequency cannot be constantly small. To minimize the energy consumed for video encoding on mobile devices, dynamically reconfiguring the clock frequency of the chip is feasible. The dynamic voltage scaling technology (DVS) is a common mechanism to save CPU energy [52, 53, 54]. The DVS is used to reduce the executing speed as much as possible while still guaranteeing the application performance. Therefore, the feasibility of DVS depends on the workload prediction accuracy and QoS control.

An approach named PACE (Processor Acceleration to Conserve Energy) [55] was proposed to schedule the voltage scaling to keep performance the same but minimize expected energy consumption. The PACE scheduling depends on the probability distribution estimation of the workload required. The real workloads simulation shows an average 20.6% energy saving gains over previously published algorithms.

In [19], the authors present an energy-efficient soft real-time CPU scheduling for mobile multimedia computing systems (GRACE-OS). DVS is integrated into soft real-time scheduling on executing speed decision. The DVS scheduler allocates CPU cycles

to the multimedia applications based on the statistical performance required and the probability distribution of CPU cycles demand. The algorithm is implemented in the Linux kernel and evaluated on real laptop environment.

2.3 Media Computing Scheduling in Cloud

In this section, the related work on media cloud frameworks will be presented. Then we review some previous works about the scheduling of video transcoding workload in cloud.

2.3.1 Media Cloud Frameworks

In cloud platform [56], the computational resource is commonly organized as groups of virtual machines (VMs), where each VM is assumed to be able to perform computing independently. In a video processing application scenario, the system can dispatch each original single video file to a standalone VM and do the whole encoding process within it. However, to utilize the large scale distributed computing resource, we can speed up the video stream processing process, which required the job dispatched within cloud is represented in a parallel-friendly way, i.e. being decomposed into individual parallel processing units [57]. A common way is to decompose a raw video into non-overlapping coding-independent groups of pictures (GOPs) and dispatch GOPs to individual VMs. For shorter encoding delay and the match between individual coding units and VMs, a finer parallelization granularity might be needed, which can be obtained through macro-block (MB) level parallelization.

Many researcher and software engineers have focused on building cloud computing frameworks [8, 58]. There are lots of critical challenging problems including storage management, computing job dispatching, etc.

A Cloud-assisted live media streaming system (CALMS), a generic framework was proposed to migration existing live media streaming services to a cloud-assisted solution [59]. Even if the system is only designed for live media streaming, video encoding/transcoding applications are also applicable. CALMS system is divided into two layers, including Cloud Layer and User Layer. The Cloud Layer can dynamically leased cloud servers to satisfy the coming live media encoding requests. Upon receiving a location-dependent users request, the Cloud Layer will transparently redirect this user to a properly selected cloud server.

In [60], the authors take video transcoding as a cloud-based video proxy framework, which utilizes cloud computing to transcode non-scalable videos into H.264 SVC in real time so as to facilitate streaming adaptation to network dynamics. The framework relies on map-reduce framework and multi-level transcoding parallelization to speed up the transcoding process with plenty of cloud resource. However, the framework is designed for only SVC transcoding and streaming, and the QoS of video is not considered.

In [61], the author proposed a framework to improve the quality of video service for mobile users. A cloud-assisted adaptive video streaming and social-aware video encoding/prefetching system was proposed. It adapts the transcoding quality according to a link condition based on scalable video coding (SVC) technology.

In [46, 62], the authors proposed a transcoding as a service framework, media computing workload on mobile devices can be offloaded to a mobile cloud system that consists of a workload dispatcher at the front end and a set of media processing service engines at the back end. The optimized offloading can reduce energy consumption on both mobile devices and the cloud jointly, which provides guidelines for the design of green mobile cloud.

2.3.2 Video Transcoding Job Dispatch

As presented works above, in the literature, a few of cloud-assisted video coding/transcoding systems have been developed, including map-reduce framework based multimedia transcoding platform [63, 64, 62] and social-aware video transcoding/prefetching system [61]. When these video coding requests arrive into system, the cloud needs to carefully store the video content and dispatch these jobs to working VM. On the video transcoding job scheduling problem, one direction of research is on the admission control under limited computing resource to guarantee QoS for the running jobs. In [65], a stream-based admission control and scheduling approach (SBACS) was designed. It uses queue waiting time of transcoding servers to make admission control decisions, including admit, defer, or reject.

Another direction is to schedule the computing resource provisioning to maximise the revenue for the transcoding service provder. In [66] Y. Song et al. proposed a profit aware dynamic biding algorithm to maximize the time-average profit of the cloud service broker. The algorithm is self-adaptive and requires no a priori statistical knowledge of the job distribution and arrival.

Different from the previous two direction, Some work focused on the transcoding job adaptation. H. Ma et al. [67] proposed a dynamic scheduling methodology on video transcoding for MPEG-DASH in a cloud scenario. The designed scheduler monitors the workload on each machine and dispatch high-priority job to the faster processor machine while low-priority job is dispatched to slower processor machine. The algorithm also adjusts the video transcoding mode (VTM) according to the system load. In [62], the authors investigate the energy-efficient job-dispatching algorithm for transcoding as a service (TaaS) in a multimedia cloud. It aims to minimize the energy consumption of service engines in the cloud while achieving low delay for TaaS. However, the approach does not consider the video QoS, such as video Quality, compression ratio.

Chapter 3

Hybrid Collision-Free Medium Access Protocol for UANs

In this chapter, focusing on a fully connected single hop network where each node can directly communicate to any other nodes in the network, an underwater MAC protocol is proposed where the design tackles the costly protocol handshakes by reducing the protocol handshake overhead and increasing the data transmission opportunities. In brief, firstly, an efficient dynamic polling-based handshake operation is designed which offers time-bounded collision-free channel assignment. Secondly, the protocol applies out-of-band signalling that separates the channel assignment protocol handshake and data transmission on different channels, where the separation of the control channel (CCH) and the data channels (DCHs) permits flexible bandwidth allocation which offers optimization of protocol designs based on a given network environment. The use of multiple DCHs also leads to a lower data rate on each DCH. This effectively reduces the normalized overhead due to the long signal propagation delay in underwater environment (a ratio of idle time to a unit of data packet transmission time), which helps improve automatic repeat request (ARQ) performance in the data link layer.

In addition, the protocol employs packet train transmissions[1] with ARQ to deal with

[1]Packet bursting transmission, like the linked carriages in a train.

channel errors while maximizing channel utilization. In particular, we apply the earlier developed juggling-like stop-and-wait (JSW) ARQ scheme [1] which integrates continuous error-control mechanism into data transmission in the underwater environment. All these components together compose an efficient comprehensive MAC protocol, which is called Hybrid Collision-Free Media Access (HCFMA), for UANs.

Finally the throughput performance of HCFMA under both *unsaturated* and *saturated* situations is analyzed. Extensive simulations are also carried out by setting up different system parameters. The results show that the performance of the HCFMA protocol outperforms its counterparts, in terms of throughput and delay, such as ALOHA with carrier sensing (ALOHA-CS) [21] and slotted floor acquisition multiple access (SFAMA) [22].

3.1 Overview of The JSW Transmission Scheme

Due to the noisy channel in the underwater environment, an error control is often necessary in protocol design. While one may rely on protocols above data link layer in the protocol stack to feature error control, we recognize the benefit of implementing an error control at the data link layer to achieve timely recovery of unsuccessful transmissions. The implementation of error control at the data link layer helps reduce the packet transmission delay. In protocol design, the JSW transmission scheme [1] is incorporated. The JSW transmission scheme is designed to achieve efficient sliding-window flow control on an underwater data link between two nodes. The operation assumes that a half-duplex data link has been established between a *transmitter* and a *receiver* for the transmission of a series of data packets.

The JSW transmission scheme assumes that the transmitter holds an estimation of the initial propagation delay (denoted by p_0) to its receiver, as well as upper bound on the relative radial velocity (denoted by v_{r}) between the two nodes. A positive sign in

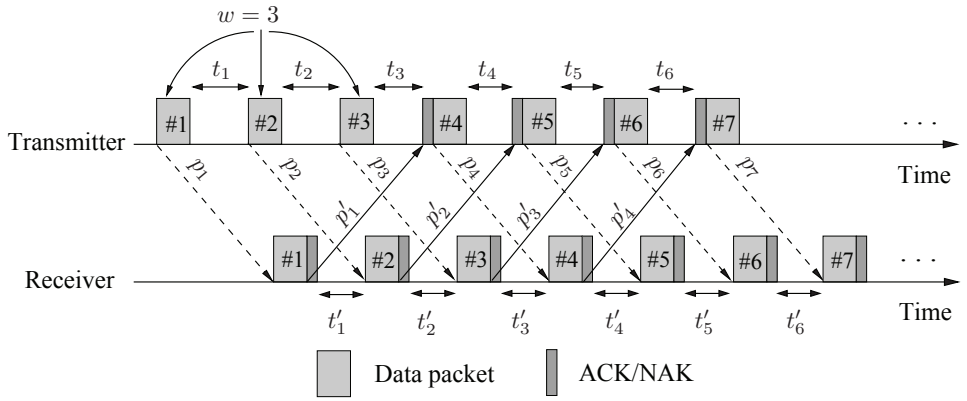


Figure 3.1: The packet transmission process of JSW scheme. [1]

v_r indicates that the two nodes are approaching each other. Fig. 3.1 shows the flow of the JSW transmission scheme when the window size (denoted by w), is three. The used notations are listed in Table 3.1.

The transmitter first sends w data packets, and then switches to a "stop-and-wait" (SW) mode, in which it waits until it has received an ACK/NAK before transmitting the next data packet. This eliminates the need for explicit time-slot synchronization between the transmitter and the receiver. The key difference from the conventional SW-ARQ protocol is that, in the conventional SW-ARQ, an ACK/NAK corresponds to the data packet that was recently transmitted by the sender, which implies that no packet will be sent unless the recently sent ACK/NAK for the data packet has been received.

In contrast, in the JSW scheme, the ACK/NAK which the transmitter expects is not the response for the most recently transmitted data packet, but for an earlier data packet. As shown in Fig. 3.1, for example, after transmitting packet #3, the transmitter waits to receive the ACK/NAK for packet #1. For the receiver, it transmits the ACK/NAK immediately after receiving a data packet. Except for the time it spends on transmitting the ACK/NAK, the receiver listens for data packets all the time.

Note that, in a static case where the relative radial velocity is zero, the window

Table 3.1: Notations used in the JSW scheme

Notation	Description
δ	Transmission time of a data packet
γ	Transmission time of an ACK/NAK packet
p_i	Actual propagation time of the i^{th} data packet from the transmitter to the receiver
p'_i	Actual propagation time of the ACK/NAK control packet for the i^{th} data packet from the receiver to the transmitter
t_i	Actual idle period at the transmitter after transmitting the i^{th} data packet
t'_i	Actual idle period at the receiver after acknowledging the i^{th} data packet with either an ACK or a NAK
w	Maximum permissible number of transmitted, but not yet acknowledged, data packets; this is also referred to as the "window size"
t_0	Computed inter-packet spacing between successively transmitted data packets
T_{jsw}	Maximum allowable duration for data transmission in each round
N_{max}	Maximum number of data packets transmitted between a pair of nodes in a cycle

size (w) and the inter-packet spacing (t_0) [68] can be, respectively, obtained as

$$w = \left\lfloor \frac{2p_0}{\delta + \gamma} \right\rfloor + 1 \qquad \text{(Eq. 3.1)}$$

and

$$t_0 = \frac{2p_0 - (w-1)\delta + \gamma}{w}. \qquad \text{(Eq. 3.2)}$$

The maximum number of data packets transmitted in a cycle (N_{max}) [68], is given by

$$N_{max} = \left\lfloor \frac{T_{jsw} - 2p_0}{\delta + t_0 + \gamma} \right\rfloor. \qquad \text{(Eq. 3.3)}$$

The overhead of the JSW transmission scheme depends on the network setup. With a proper design under some common network setup, it may offer no more than 10% of transmission overhead. More details about the JSW transmission scheme can be found in [1].

3.2 The HCFMA Protocol

In this section, we explain in detail the operation of the HCFMA protocol. The notations used are defined in Table 3.2. Assuming that a single hop network where all nodes are able to communicate with each other directly, and the signal propagation time from a node to any other node is not longer than the maximum end-to-end signal propagation delay.

3.2.1 Multi-channel Setup

The HCFMA protocol operation consists of a channel reservation (CR) period with duration of $T_{po}^{max} + \tau$, followed by a data transmission (DT) period with duration of T_{jsw}. The channel reservation operation appears on the CCH where a collection of nodes participate in requesting for channel access right of a DCH, and at the end of the period,

Table 3.2: Notations used in the HCFMA operation

Notation	Description
(φ, i)	The i^{th} one-hop neighbor of group head φ
$T_{(\varphi,i)}$	Time at which node (φ, i) starts transmitting its REQ packet
$d_{(\varphi,i)}$	Propagation delay between group head φ and node (φ, i), ordered from smallest to largest
N_{dch}	Total number of DCHs
T_{po}^{max}	Maximum allowable duration for channel reservation
T_{enq}	Transmission time of an ENQ packet
T_{req}	Transmission time of a REQ packet
T_{noti}	Transmission time of a NOTI packet
t_g	Guard time to protect against any inaccurate estimation of the inter-node propagation delays
λ	Poisson arrival rate at each node
R	Total available bandwidth
N	Total number of nodes
N_g^{max}	Maximum allowable number of nodes in each group
N_{dch}	Total number of DCHs
p_e	Packet error rate
L	Length of each data packet
T_c	Transmission cycle duration for each group
G	Total offered load
τ	Maximum end-to-end propagation delay
t_σ	Interval between two succeeding CRs

only a selected node, which shall be called the *winner*, acquires the channel access right. The winner then switches to the DCH for its data transmission at the end of the channel reservation period.

To achieve high efficiency in channel reservation operation when dealing with a large number of nodes, the protocol is designed to divide nodes into different groups as evenly as possible[2]. Different groups of nodes take turn to participate in the channel reservation.

[2]One simple way to achieve this is to enforce detection of the number of members in each group during the joining procedure. In other words, when a new node wishes to join the network, it must first observe the channel reservation periods of all groups and pick the group with the lowest number of group members to join by overhearing the "enquire" (ENQ) messages, as depicted in Section 3.2.4.

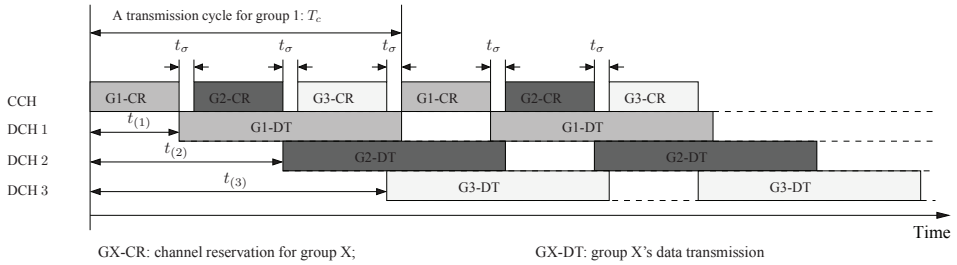


Figure 3.2: Timing diagram of multi-channel setup.

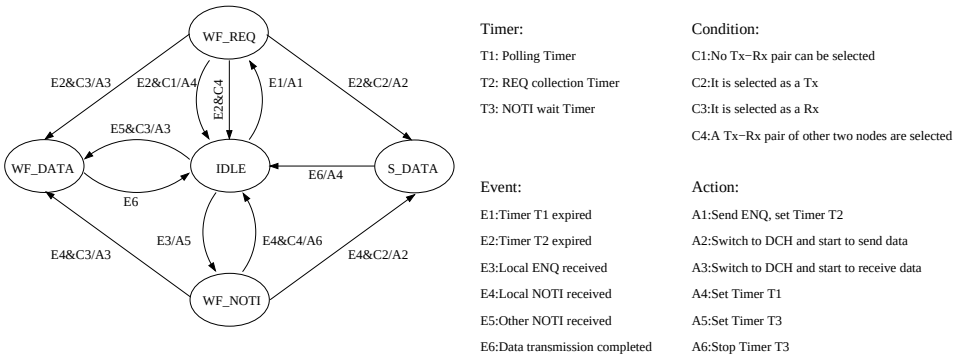


Figure 3.3: The finite state machine of the HCFMA protocol.

Fig. 3.2 shows the timing diagram in a multi-channel setup where $N_{dch} = 3$. Accordingly, the three groups, say G1, G2 and G3, are served by DCH 1, DCH 2 and DCH 3, respectively. The CCH repeats three channel reservation rounds during a transmission cycle. After each reservation procedure for a particular group, the assigned sender-receiver pair proceeds with data transmission in their corresponding DCH immediately. The data transmission ends just before the next channel reservation for the group occurs where the sender must return back to the CCH to coordinate the reservation. Inevitably, a transmission gap on DCH is resulted which causes some loss of bandwidth.

3.2.2 The Channel Reservation Procedure

The main purpose of channel reservation is to allocate the corresponding DCH to a node that is ready for transmission within its group. The channel reservation procedure is

executed repeatedly on the CCH, and each group of nodes takes turn to participate in its corresponding channel reservation round.

The channel reservation design is a classical problem. Typically, there are three strategies, namely, random access, token passing, and polling. Considering the long signal propagation delay and the high error rate in the underwater environment, the use of a distributed polling mechanism appears to be a good candidate for these unique channel characteristics. The channel reservation procedure is depicted by the finite state machine (FSM) of the HCFMA protocol, as shown in Fig. 3.3[3].

A node implementing the HCFMA protocol resides in one of the following five states: IDLE, WF_REQ, WF_NOTI, S_DATA, and WF_DATA. The default state for a node to operate is IDLE. Without loss of generality, assuming that in a group, a particular node is elected as a *group head*. This node has its timer T1 turned on during the earlier election for the group head.

The expiration of T1 in the group head marks the beginning of a channel reservation round for its group. The elected group head broadcasts a short control packet called "enquire" (ENQ) to initiate channel reservation. Accordingly, it transits from state IDLE to state WF_REQ. Since the distances between group members and the group head are different, each group member may detect the ENQ message at a different time. All idle group members react to the ENQ message by replying with a "require" (REQ) control packet specifying the amount of data ready for transmission to a particular receiver, or a zero amount indicating no ready data. Note, in this case, that their states are changed from IDLE to WF_NOTI.

Each group member may transmit its REQ immediately after receiving the ENQ message. This may lead to a collision of REQ packets when two nodes are located at the

[3]In this figure, the statement before "/" is the condition of the transition while the statement after "/" is the action of the transition.

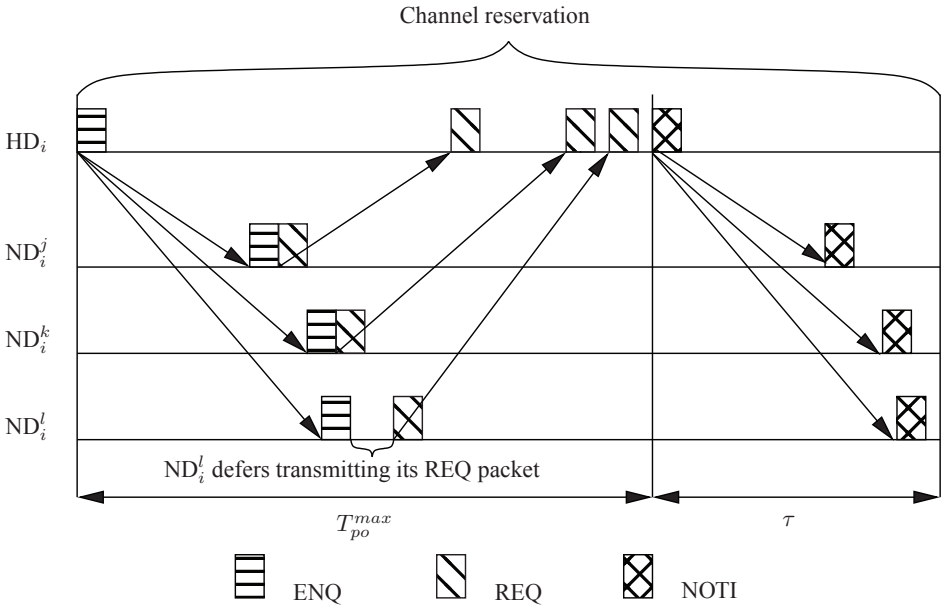


Figure 3.4: An example of channel reservation on CCH.

same distance from the group head. This collision can be avoided if inter-node propagation delay information is available. This information can be acquired as follows. In static scenarios, such inter-node propagation delays can be estimated by exchanging some control messages during network initialization; alternatively, if each node has positioning capability, messages could be exchanged between neighboring nodes to update each other about their locations [69], which can then be used for computing the inter-node propagation delays. With this information, the group head can compute the needed delay for each REQ reply such that all REQ replies arrive at the group head with no collision. The ENQ packet carries a list of delays for each group member to wait before replying its REQ message. The instant $T_{(\varphi,i)}$ at which node (φ, i) starts transmitting its REQ packet, can be obtained as:

$$T_{(\varphi,i)} = 2d_{(\varphi,i-1)} + T_{(\varphi,i-1)} + T_{req} - 2d_{(\varphi,i)} + t_g,$$

where $i \geq 2$ and $T_{(\varphi,1)} = 0$. This ensures collision-free reception of REQ replies at the group head.

After broadcasting the ENQ packet, the group head listens to the CCH and collects the REQ messages replied by its group members. The collection time shall last for at least twice the propagation delay between itself and its farthest group member plus the transmission time of an ENQ packet and an REQ packet. After the collection, the group head orders the REQ messages, selects a winner, and broadcasts a "notify" (NOTI) control packet where it specifies the winner and its intended receiving peer. For performance consideration, the node with the largest amount of ready data is chosen as the winner.

The winner in this round of channel reservation will automatically become the group head of the next round. This assignment is achieved by starting its timer T1 that schedules the next ENQ transmission as illustrated in Fig. 3.3. Note that a poll may result in no request for channel reservation when all nodes in the group have no packet to transmit. In this case, a NOTI packet is broadcast to indicate so and the group head continues to serve as the group head for the coming round. Fig. 3.4 illustrates a channel reservation round showing the three-way polling handshake of ENQ-REQ-NOTI message exchanges.

To request for a data transmission, a node must have ready data to transmit and also ensure that its intended receiver is available to participate in the data transmission. It is possible that the intended receiver is unable to participate in the data transmission. This happens when the intended receiver belongs to another group and is participating in an ongoing data transmission on another DCH when the ENQ-REQ-NOTI message exchange occurs. In the design, the responsibility for detecting such unavailability of intended receiver falls on the node requesting for data transmission. Referring to Fig. 3.2, concurrently with a channel reservation round, there may exist two pairs of nodes performing data transmission on other DCHs, which are unable to monitor the current CCH and participate in the next data transmission. Any data transmission addressed to them

should not be made in the current channel reservation round. The information about node availabilities can be easily obtained by monitoring the previous channel reservation rounds on the CCH performed by other groups with the assistance from other nodes.

After collecting all the REQ packets associated with the DCH, the group head has the following possibilities: 1) if no pair of nodes can be selected as the transmitter-receiver pair, then the group head returns its state to IDLE; 2) if a pair of nodes (excluding the group head itself) are selected as the transmitter-receiver pair, then the group head also returns its state to IDLE; 3) if the group head itself is selected as the transmitter, then the group head switches to the corresponding DCH to send data in the JSW manner, going into the S_DATA state; 4) if the group head itself is selected as the receiver, then the group head switches to the corresponding DCH to receive data in the JSW manner, going into the WF_DATA state.

For an idle group member (other than the group head), when it receives the local NOTI packet in the WF_NOTI, there exist three possibilities: 1) if it is selected as neither the intended transmitter nor the intended receiver, then it returns to the IDLE state; 2) if it is selected as the transmitter, then it switches to the corresponding DCH to send data with the JSW manner, and it transits into S_DATA state; 3) if it is selected as the intended receiver, then it switches to the corresponding DCH to receive data with the JSW manner, and it transits into WF_DATA state.

3.2.3 The Data Transmission Procedure

Upon hearing the NOTI packet, the transmitter and the receiver switch to the corresponding DCH to start data transmission, which may take place right after detecting the NOTI message.

The data transmission shall last for a predefined time period of T_{jsw}. The transmitter uses the JSW transmission scheme to send data packets to the receiver. Any new data

packet that is generated by the transmitter during data transmission and destined for the current receiver will also be transmitted if time permits. After data transmission, both the transmitter and the receiver switch back to the CCH, returning to the IDLE state. The transmitter will become the group head and initiate an ENQ packet when its channel reservation round arrives.

3.2.4 The Joining and Leaving Procedures

It is expected that some nodes may join and leave the network at any time of the protocol operation. A node leaving event can be easily detected by lack of REQ reply. The group head may update all group members about this event by indicating it in the NOTI message. To avoid unnecessary false alarms, a member may be considered leaving the network only after a number of consecutive absences in REQ replies.

The joining of nodes can be achieved during the ENQ-REQ handshake. A new node may first observe the channel reservation periods of all groups and pick the group with the lowest number of group members to join. This is to achieve even distribution among all the groups. After picking a group to join, based on monitoring of a completed ENQ-REQ handshake procedure, the node can detect the last appearance of the REQ message in the procedure. The node may take the opportunity to transmit a new REQ message right after the last REQ message. To incorporate this operation, the group head must always wait for a longer time to detect not only all expected REQ messages but also some new REQ messages to allow members to join. This new REQ message also contains some time information for the group head to estimate one-way signal propagation delay between the group head and the new node.

Note that the new REQ message may suffer from collisions if there are more than two new nodes wanting to join at the same channel reservation round. Persistent collision

can be easily avoided by allowing more timeslots for new nodes to choose for their REQ transmissions and enforcing random selection of a timeslot. This operation only slightly lengthens the ENQ-REQ handshake duration as T_{req} is much smaller than T_{po}^{max}.

3.2.5 The Error Recovery Procedure

Since the three-way polling handshake of ENQ-REQ-NOTI is triggered by a group head, it is important to ensure the existence of a group head throughout the protocol operation. Due to, for example, hardware failure, the group head may be disconnected resulting in no ENQ packet appearing at the beginning of the channel reservation round. A procedure is propose for election of a new group head as the error recover procedure. A lack of an ENQ packet in a particular channel reservation round triggers the election procedure. The election procedure will take place in the next corresponding channel reservation round, and it may last over several corresponding channel reservation rounds. Each corresponding channel reservation round is seen as an election round.

In each NOTI message, the corresponding group head also specifies several candidate nodes in an ordered list. These nodes will prepare to serve as the next group head if no ENQ packet is received at the beginning of the next channel reservation round. These candidate nodes are the ones that belong to the same group and are currently detected to be in idle states. In the case where no ENQ is detected at the beginning of the next channel reservation round, the first candidate node in the list automatically becomes the group head in the next subsequent round. The second candidate node may become the group head if the first candidate node does not respond to the corresponding channel reservation round. The same rule applies to the third candidate node, the fourth, and so on.

3.2.6 The Initialization Procedure

The network requires the formation of groups and the election of a group head to start its operation. A lack of group formation can be detected when a node fails to detect any ENQ packet on the CCH after a certain period of sensing, and this triggers the initialization procedure. The initialization procedure is similar to the election procedure for error recovery, except that in the latter there is no formation of groups. In other words, all nodes are assumed to be in the same group. A different control packet called INIT is used in this procedure. Similar to the error recovery procedure, this procedure eventually produces a group head. Other group heads can be formed by iterating the same procedure. Then each node uses the joining procedure to add itself into an individual group.

3.3 Throughput Performance Analysis

In this section, the throughput of the proposed HCFMA protocol is analyzed in both *unsaturated* and *saturated* situations. Here *unsaturated* refers to the case where all the data packets generated at each node will be correctly transmitted due to the use of the error-control mechanism in HCFMA, while *saturated* refers to the case where there is one or more nodes with at least N_{max} to be sent to one destination when responding to an ENQ packet.

For simplicity, making the following assumptions: 1) each group consists of $\frac{N}{N_{dch}}$ nodes; 2) each node has a Poisson data source with an average arrival rate of λ; 3) the propagation delay between any pair nodes is τ; 4) CCH has the same bandwidth as one DCH. Next, calculating the HCFMA's throughput in both *unsaturated* and *saturated* situations, respectively.

1) In unsaturated situations: Let m be a sufficiently large positive integer. For each node, the average arrival rate during the first m transmission cycles is then $\lambda \times mT_c$, where $T_c \triangleq T_{po}^{max} + T_{noti} + \tau + T_{jsw}$ and $T_{po}^{max} = 2\tau + T_{enq} + (N_g^{max} - 1)(T_{req} + t_g)$. Thus, the average number of data packets generated by the nodes belonging to the first group can be expressed as $\frac{N}{N_{dch}} \times m\lambda T_c$. After the m transmission cycles, the number of packets left in the nodes corresponding to the first group, on the average, should be less than $\frac{N}{N_{dch}} \times (N-1) \times [(N_{max} - 1) + \lambda T_{jsw}]$, where $\frac{N}{N_{dch}}$ is to account for the number nodes in one group, $N-1$ is to account for $N-1$ possible queues in one node corresponding to different possible receivers, and $(N_{max} - 1) + \lambda T_{jsw}$ is to account for the maximum possible number of packets left in one queue, i.e. $N_{max} - 1$, and the number of packets just generated in the current transmission. This is because, if the number of packets left in one queue for one node is more than $N_{max} - 1$, the node will be selected as the winner[4]. Therefore, the average throughput $\eta_1(m)$ for the m transmission cycle for group 1 is given in (Eq. 3.4).

$$\eta_1(m) \geq \frac{\{\frac{N}{N_{dch}} \times m\lambda T_c - \frac{N}{N_{dch}} \times (N-1) \times [(N_{max} - 1) + \lambda T_{jsw}]\} \times L}{mT_c \times (R/(N_{dch}+1))}. \quad \text{(Eq. 3.4)}$$

As $m \to \infty$, the average throughput for DCH 1 (denoted by η_1) satisfies:

$$\begin{aligned} \eta_1 &= \lim_{m\to\infty} \eta_1(m) \\ &\geq \frac{N}{N_{dch}} \times \lambda \times \frac{L}{R/(N_{dch}+1)} \\ &= \frac{N}{N_{dch}} \times \lambda \times \delta = \frac{G}{N_{dch}}. \end{aligned} \quad \text{(Eq. 3.5)}$$

Then, the total throughput in an unsaturated situation (denoted by η_{us}) can be approximately calculated as

$$\eta_{us} = N_{dch} \times \eta_1 \geq G. \quad \text{(Eq. 3.6)}$$

[4]There exists a special case where a node with more than $(N_{max} - 1)$ data packets destined for one node is currently receiving data on other DCH. Clearly, it will be soon selected as the winner in the upcoming transmission cycle.

Since η_{us} should not be larger than G, we have

$$\eta_{us} = G. \tag{Eq. 3.7}$$

2) In saturated situations: As can be seen from Section 3.1,

The maximum number of data packets correctly transmitted in a transmission cycle duration (T_f) is $N_{max}(1 - p_e)$. Clearly, for saturated situations, the throughput of one DCH in a transmission cycle duration is $\frac{N_{max}(1-p_e)\delta}{T_c}$.

Recall that there are totally $N_{dch} + 1$ subchannels including N_{dch} DCHs. As a result, the total average throughput in a saturated situation (denoted by η_s) is given by

$$\eta_s = \frac{N_{dch}}{N_{dch} + 1} \times \frac{N_{max}(1 - p_e)\delta}{T_c}, \tag{Eq. 3.8}$$

where N_{max} is given by (3). Note that p_0 there is τ.

From ((Eq. 3.7)) and ((Eq. 3.8)), the throughput of the HCFMA protocol, η, can be approximately written as

$$\eta = \begin{cases} G, & 0 \leq G < \eta_s \\ \eta_s, & \text{otherwise.} \end{cases} \tag{Eq. 3.9}$$

3.4 Simulation Results and Discussions

In this section simulations are conducted to evaluate the performance of the HCFMA protocol in terms of throughput and delay. Throughput measures the ratio of the sum of the time taken to correctly receive data packets at the destination nodes to the total simulation time, and delay measures the duration from the instant when a data packet is generated to the time when it is correctly received at the destination.

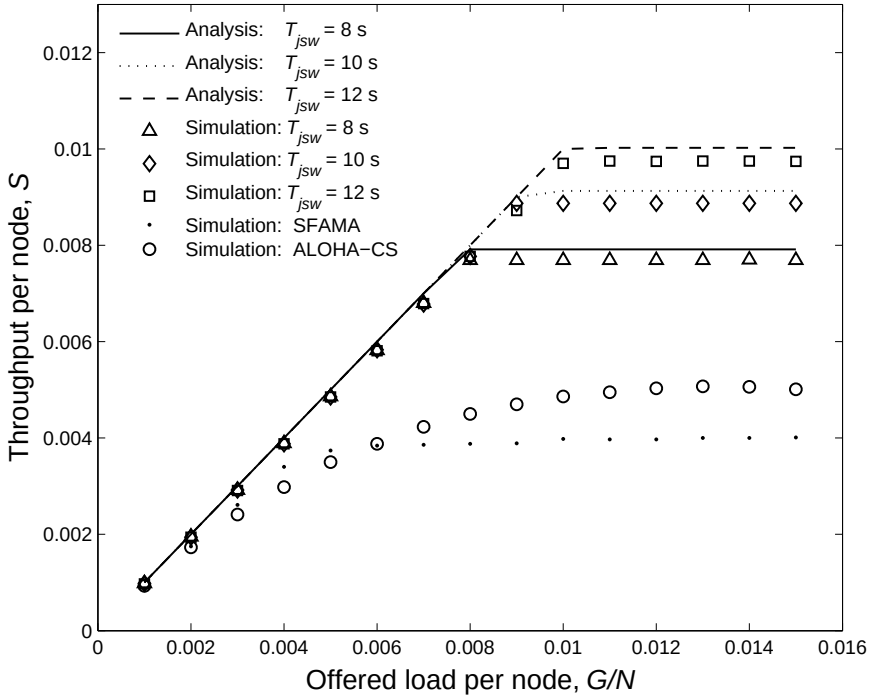


Figure 3.5: Throughput versus offered load in the worst case for $N_{dch=3}$.

3.4.1 Simulation Setting

In the simulation, the overall data rate is set to 12 kbps; each subchannel (including CCH and DCH) is given the same transmission rate. Each data packet transmitted has a size of 1024 bits. The ENQ packet is set to 200 bits while both the REQ and NOTI packets are set to 64 bits, respectively; other control packets (i.e., ACK/NAK) are set to 40-bit long. The guard time (t_g) is set to 0.01s. Each data packet transmitted on DCH experiences a packet error rate of 0.1, unless specified otherwise. All the results presented here are the average of 100 simulation runs, each lasting for 100000 seconds.

3.4.2 Worst-Case Performance Comparisons

First considering the worst cases where all pairs of nodes are τ units of time apart. Fig. 3.5 is depicted to compare the throughput of HCFMA obtained by analysis based on (Eq. 3.9) with that of simulation for different values of T_{jsw} for $N_{dch} = 3$. HCFMA utilizes four channels, each of which has a transmission rate of 3 kbps. It shows good agreement between the two results indicating the accuracy of the analysis. Note that when the value of T_{jsw} is set higher which allows a longer period in the data transmission, the throughput becomes higher accordingly. This is because the per data transmission overhead due to polling is reduced as T_{jsw} increases.

In Fig. 3.5, let us also compare HCFMA with two other protocols, namely, ALOHA-CS which is its predecessor, and SFAMA which is one of its closest peers. Both the ALOHA-CS and SFAMA protocols utilize a single common channel of 12 kbps. In SFAMA, assuming that every successful reservation permits a transmission up to three time-slot or 36 packets. From the results, it can be seen that HCFMA achieves a throughput twice as high as that of ALOHA-CS and SFAMA. This can be attributed to the fact that 1) both ALOHA-CS and SFAMA would render a large amount of collisions as the offered load becomes large, which drastically degrades their throughput; 2) by adopting the multi-channel technique and the channel reservation mechanism, HCFMA can avoid collisions during either in channel reservation or data transmission, thus resulting in much higher throughput.

It is also observed in Fig. 3.5 that based on the current setup, the performance of SFAMA is similar to that of ALOHA-CS. This is because the performance of SFAMA depends on the duration of data transmission time-slot. With the current setting, its operation is not optimal where it operates at the level closed to that of ALOHA-CS. Since the performance of SFAMA depends on the duration of data transmission timeslot,

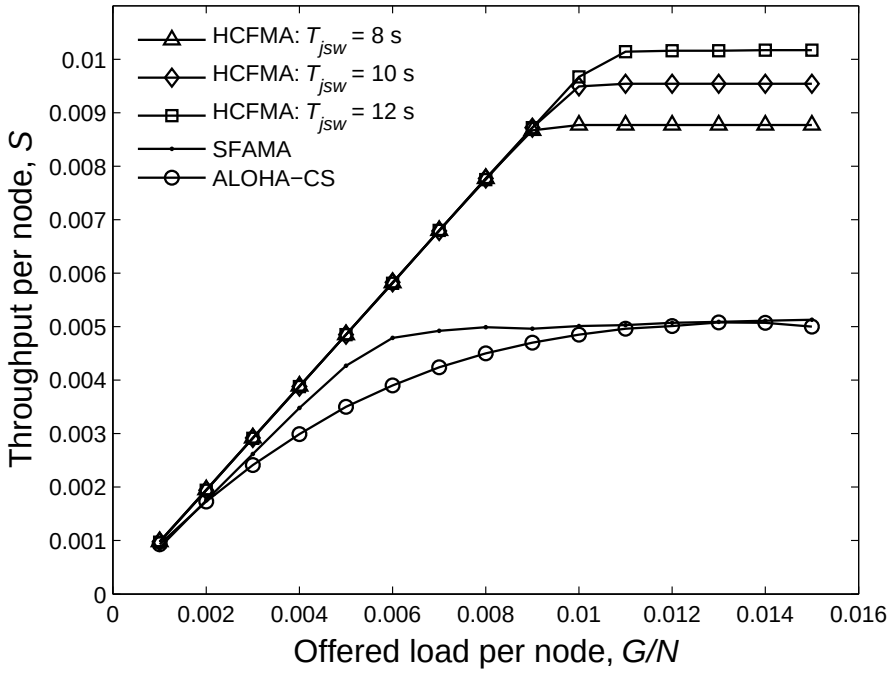


Figure 3.6: Throughput versus offered load for $N_{dch=3}$.

its performance may surpass that of HCFMA by introducing a longer timeslot, which is studied and presented in Section 3.4.5. However, this setting may have adverse effect in delay performance which may not be desirable.

3.4.3 Performance Comparisons in General Network Scenarios

In this subsection simulations with a general setup are conducted, where the network topology comprises 36 nodes with a grid spacing of 200 m. Accounting for the impact due to the movement of ocean current, the simulation assumes that each node is uniformly distributed in a circular region centered at the grid intersection point, with a radius of 20 m. The maximum transmission range for each node is 1500 m, implying the single-hop scenario with the maximum propagation delay of approximately one second. For every packet generated by a node, the node randomly selects one of the other nodes as

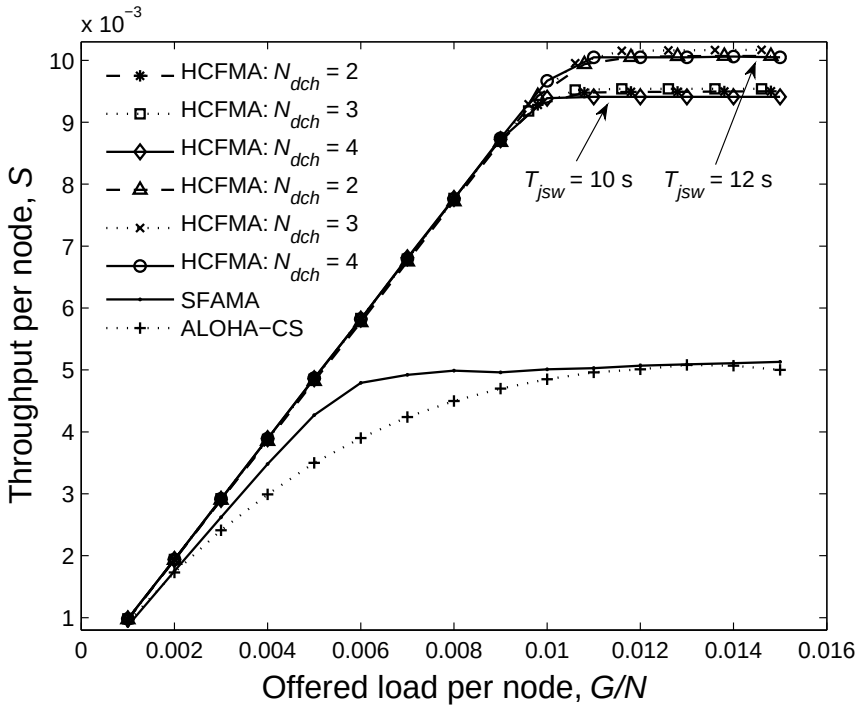


Figure 3.7: Throughput versus offered load for different N_{dch} for T_{jsw}.

the destination with equal probability. Fig. 3.6 shows the throughput performance of HCFMA, ALOHA-CS and SFAMA. The same conclusions as those for Fig. 3.5 can be drawn for Fig. 3.6.

Fig. 3.7 illustrates the impact of the number of DCHs used in HCFMA on throughput for $T_{jsw} = 10$ and 12 s, respectively. It is interesting to note that $N_{dch} = 3$ yields the highest throughput performance compared to the settings of $N_{dch} = 2$ and $N_{dch} = 4$ for $T_{jsw} = 10$ and 12 s, respectively. This implies that $N_{dch} = 3$ is an adequate choice for the performance consideration.

It is important to highlight that, although a better throughput can be obtained in SFAMA by using packet trains, the inefficiency of SFAMA cannot be fundamentally eliminated due to the large number of collisions introduced during the channel reservation. This is particularly the case when traffic load is high.

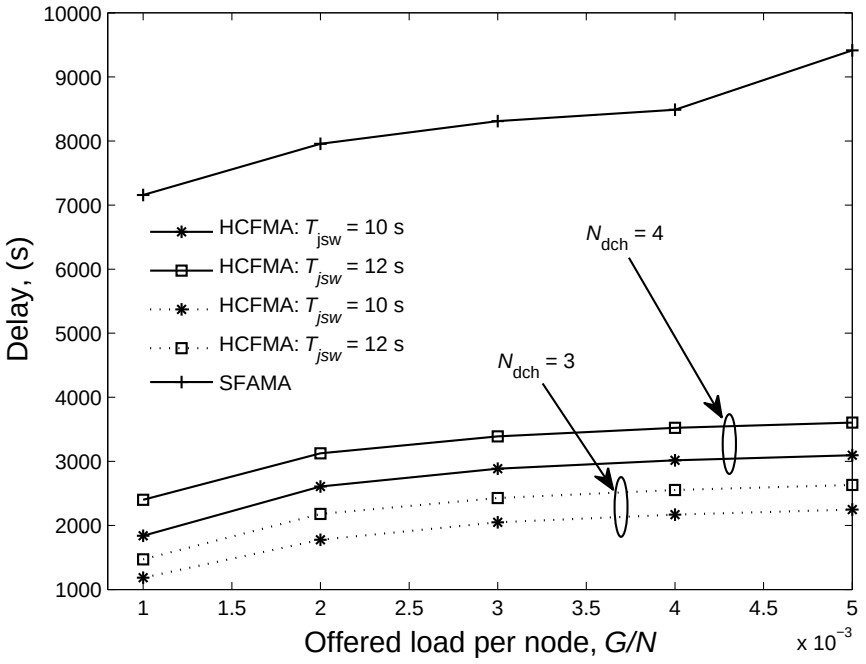


Figure 3.8: Delay versus offered load for different N_{dch}.

The delay performance of HCFMA and SFAMA is depicted in Fig. 3.8. It can be seen that, the delay of HCFMA gently increases with the increase of offered load, and it is typically less than 3200 s. By contrast, the delay performance of SFAMA is consistently above 7000 s, which is much larger than that of HCFMA. The reason is that the use of multiple channels and the channel reservation mechanism not only avoids collisions during channel reservation, but also provides chances for transmitting more data packets concurrently, which leads to lower delay, while the use of a single channel with contention-based handshake for channel reservation in SFAMA introduces long channel access delay, especially under high offered loads. It is also observed that a large T_{jsw} would decrease the delay performance. For example, when the offered load per node is less than 0.005, the end-to-end delay of HCFMA for $T_{jsw} = 12s$ is much larger than that for $T_{jsw} = 10s$, as shown in Fig. 3.8, while achieving the same throughput, as shown in Fig. 3.5. This suggests that the parameter T_{jsw} may be optimized so as to offer timely transmission.

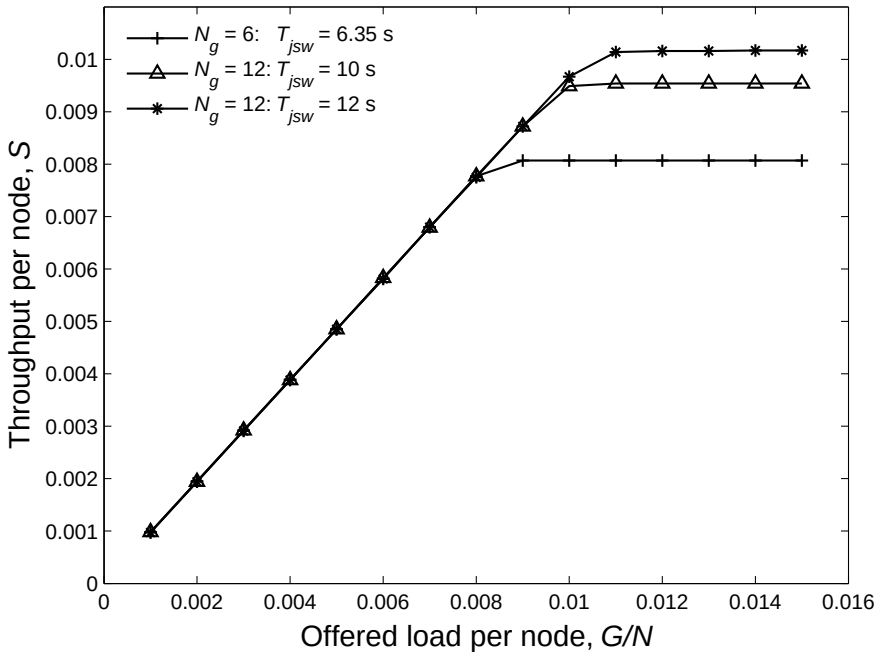


Figure 3.9: Throughput versus offered load for different N_g.

3.4.4 HCFMA Throughput and Delay Tradeoff

In Figs. 3.9-3.10, the results show the impacts of different values of the number of nodes per group, N_g and T_{jsw}, on the throughput and delay performance of the HCFMA protocol with $N_{dch} = 3$. It can be seen from Fig. 3.9 that, when offered load per node is less than 0.008, the throughputs achieved under different setting are the same. As G grows beyond 0.008, a larger value of T_{jsw} offers a higher level of throughput. Nonetheless, from Fig. 3.10 it can be seen that smaller N_g and T_{jsw} values lead to a smaller delay. This is attributed to the fact that, when G is small, a large T_{jsw} causes the under-utilization of channel resource, which leads to a longer delay. Thus, in practical deployment, an appropriate choice of parameter sets should be sought to achieve a good tradeoff between throughput and delay.

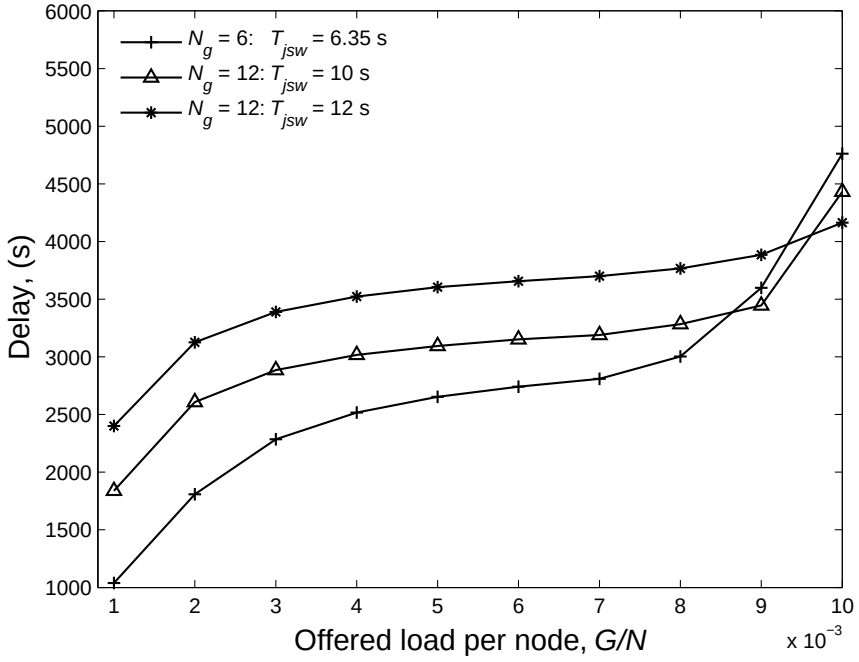


Figure 3.10: Delay versus offered load for different N_g.

3.4.5 Performance Comparisons in Light Load Scenario

In the previous subsections 3.4.2 through 3.4.3 , the proposed HCFMA based on polling has shown the outperformance of throughput and delay under moderate to heavy load scenario. In this subsection, the simulations compare the performances of HCFMA, ALOHA-CS and SFAMA under a light load scenario using simulation.

In the new simulation setup, there are only 6 nodes fully connected in the single hop network. For SFAMA, each successful handshake permits a continuous transmission of up to D_{slot} data transmission slots (the setup of $D_{slot} = 3, 4, 6, 12$ are evaluated, each slot permits transmission of up to 12 data packets). For HCFMA, three sub-channels are used where one is used for polling control and two others are used for data transmission. In other words, nodes are split into two groups for polling purposes. Each reservation permits transmission of up to 27 data packets.

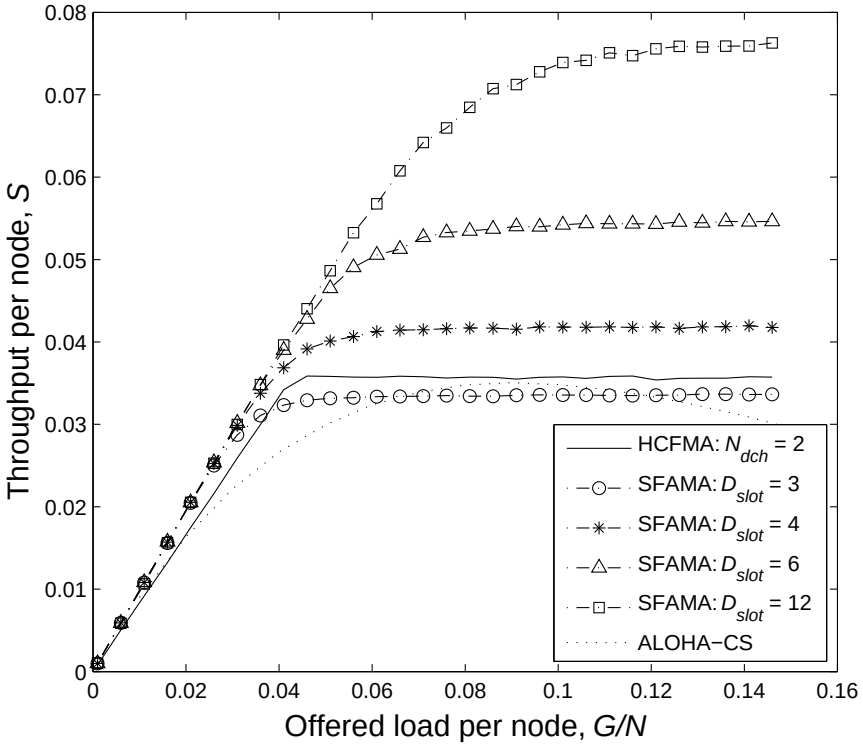


Figure 3.11: Throughput versus offered load for light load scenario.

From the results shown in Figs. 3.11-3.12, it is shown that the throughput performances of HCFMA and SFAMA are similar for $D_{slot} = 3, 4$. As the performance of SFAMA depends on D_{slot}, higher D_{slot} values further promote the throughput of SFAMA. The outperformance of SFAMA in these cases is mainly attributed by a small number of nodes in this scenario and large D_{slot}.

In delay performance, as a reservation based protocol, HCFMA offers much lower and stable transmission delay than that of SFAMA. Transmission delay in SFAMA increases as offered load increases. For SFAMA, higher D_{slot} settings records lower delay than that of a lower D_{slot} settings when the offered load is high. This is because more data transmission can be performed after a successful channel assignment in SFAMA which reduces the time overhead for each data packet transmission. However, short term un-

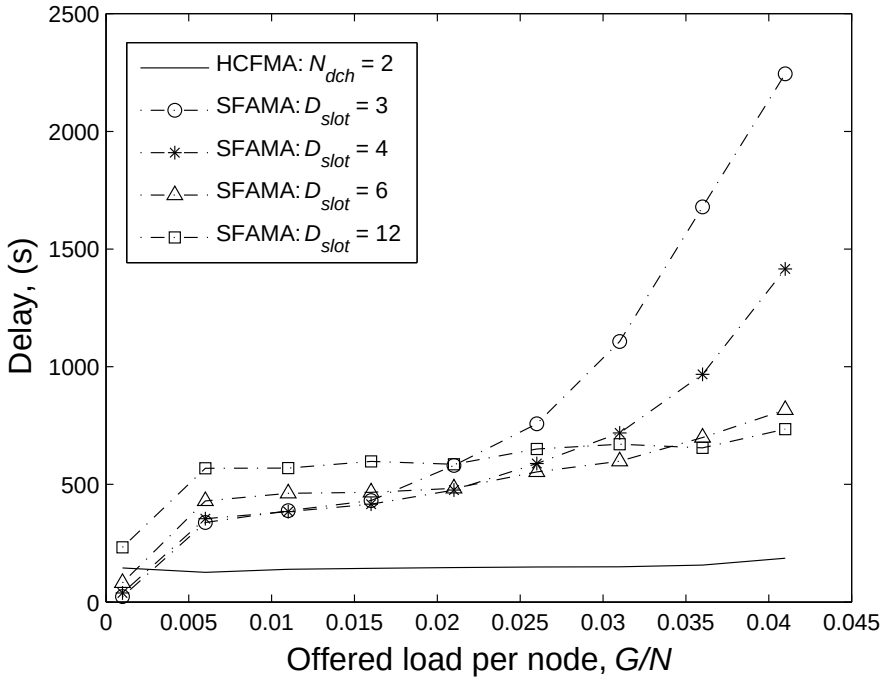


Figure 3.12: Delay versus offered load for light load scenario.

fairness may arise. On the other hand, HCFMA offers adequately high throughput with relatively low delay, showing its attractiveness under a more realistic scenario.

3.5 Summary

In this chapter, a dynamic polling based protocol called HCFMA is proposed for underwater acoustic networks. Noting that the polling mechanism is relatively efficient for the underwater environment that has lengthy signal propagation delay, a dynamic polling scheme is designed operating in an out-of-band multi-channel setup. The results have shown encouraging throughput and delay performance compared with that of ALOHA-CS and SFAMA, especially under heavy load conditions.

The chapter provides detailed discussions on error recovery, joining and leaving of nodes, and initialization procedures for the protocol illustrating its capability to handle

critical and exceptional events while such discussions are often omitted in many existing underwater MAC protocol designs. Inclusion of these discussions also shows the practicability of the proposed MAC protocol for implementations.

Chapter 4

Energy Minimization via DVS for Real-Time Mobile Video Encoding

In this chapter, a systematic approach is developed to investigate the problem of how to dynamically reconfigure the clock frequency in the mobile device to minimize the energy consumption, while respecting the QoS requirement. A probabilistic QoS model is adopted, in which the encoding process should complete with a target probability within a specified delay deadline for each GOP. Such a requirement is translated into the number of CPU cycles required before the encoding deadline. Under this model, the optimal clock-scheduling problem is formulated as a constrained optimization problem, in which the objective is to minimize the total energy consumption with a constraint of delay deadline. The optimization problem is solved analytically and obtain closed-form solutions for both the optimal clock frequency schedule and the minimum energy consumption. Then the lightweight clock scheduling algorithm is applied to real-time H.264/AVC encoding application on mobile devices. The numerical results suggest that significant amount of energy can be saved by using our optimal solution.

4.1 Model and Formulation

In this section, a mathematical model is first presented for energy consumption in mobile devices and a probabilistic model for encoder workload. Under this model, the problem of optimal clock-scheduling mechanism is formulated as a constrained optimization problem.

4.1.1 Energy Consumption Model for Mobile Devices

The energy consumed on mobile devices, for a special computing task, depends on the number of CPU cycles and the clock frequency. First, in CMOS circuits, the clock frequency f, is approximately linearly proportional to the voltage supply V, and the energy per cycle E_c is proportional to V^2 [50]. Therefore, the energy consumption per cycle can be expressed as

$$E_c = \kappa f^2, \qquad \text{(Eq. 4.1)}$$

where κ is a coefficient depending on the chip architecture. The energy per CPU cyle, as denoted in (Eq. 4.1), has the following properties including:

- $E_c(f)$ is an increasing function of the clock frequency of f;
- $E_c(f)$ is a convex function of the clock frequency of f.

Given these properties, it can be seen that CPU can conserve energy substantially by running more slowly. However, for real-time video encoding, the encoder has to meet a specified deadline for each GOP, which suggests that the clock frequency cannot be constantly small. Therefore properly scheduling of CPU clock frequency can conserve energy while meeting the required deadline simultaneously.

4.1.2 Probabilistic Workload Model for Video Encoding

The workload of an encoding task is characterized by the number of CPU cycles, denoted as W. It is normally modeled as a random variable. As shown in [55] [70], the (truncated) normal distribution can be used to model the workload. The probability density function (PDF) of normal distribution is given by

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad \text{for} \quad x > 0. \tag{Eq. 4.2}$$

Here we assume a probabilistic workload model that an encoding task should be completed with probability ρ by allocating W_ρ cycles. This requirement can be expressed as the cumulative distribution function (CDF)

$$F(x) = \Pr[x \leq W_\rho] \geq \rho. \tag{Eq. 4.3}$$

As such, the required number of CPU cycles W_ρ, for an empirical normal distribution and completion probability ρ, is given by

$$W_\rho = F_W^{-1}(\rho). \tag{Eq. 4.4}$$

In this model, a CPU frequency scheduling consists of two parts, including the pre-deadline part and the post-deadline part. The maximum number of cycles executed in the pre-deadline part equals to W_ρ. The post-deadline part describes the scheduling when task has missed its deadline. In this chapter, we only focus on the scheduling policy for the pre-deadline part. If the encoding process misses its required deadline, it is assumed that the post-deadline part is executed with a maximum clock frequency.

4.1.3 Problem Formulation

In the real-time video encoding system, we focus on the encoding task of each GOP. Specifically, the encoding task of each GOP is required to meet the deadline at a specified

probability, which can be expressed as a probabilistic QoS model,

$$\Pr[t \leq T] \geq \rho, \qquad \text{(Eq. 4.5)}$$

where t is the encoding time for an individual GOP-encoding task and T is the required video encoding completion deadline for a GOP-encoding task.

The encoding requirement specified in (Eq. 4.5), under the probabilistic workload model, can be translated into a requirement of the number of CPU cycles, i.e. W_ρ defined in (Eq. 4.4). Therefore, the total energy consumption can be derived as

$$\begin{aligned} \varepsilon_c &= \kappa \int_0^{W_\rho} p(x) \int_0^{x} [f(w)]^2 dw dx \\ &\overset{(a)}{=} \kappa \int_0^{W_\rho} [f(w)]^2 \int_w^{W_\rho} p(x) dx dw \\ &\overset{(b)}{=} \kappa \int_0^{W_\rho} [f(w)]^2 (1 - F(w)) dw, \end{aligned}$$

where $f(w)$ is the clock frequency defined as a function of w, which is the number of CPU cycles that has been completed for the current task, (a) results from the exchange of integral order, and (b) is from the definition of the CDF.

Due to the CPU cycle is discrete, letting $dw = 1$, a discrete version of the energy consumption can be written as

$$\varepsilon_c = \kappa \sum_{w=1}^{W_\rho} F^c(w)[f(w)]^2, \qquad \text{(Eq. 4.6)}$$

where $F^c(w)$ is the complementary cumulative distribution function (CCDF) of workload. Notice that $F^c(w)$ is the probability in which the encoding task has not finished after executing w CPU cycles.

Using the above definition, we can formulate the optimal clock frequency allocation

problem as the following constrained optimization problem,

$$\min_{f(w)} \quad \varepsilon_c = \kappa \sum_{w=1}^{W_\rho} F^c(w)[f(w)]^2, \tag{Eq. 4.7}$$

$$s.t. \quad \sum_{w=1}^{W_\rho} \frac{1}{f(w)} \leq T, \tag{Eq. 4.8}$$
$$f(w) > 0,$$

where the constraint of (Eq. 4.8) corresponds to the task deadline requirement.

4.2 Optimal DVS scheduling

In this section, we solve the optimization problem via a Lagrangian method and obtain the closed-form solutions for the optimal clock-scheduling policy and the corresponding minimum energy consumption. We then evaluate the characteristics of the clock-scheduling policy.

4.2.1 Derivation of Optimal DVS policy

In this subsection, we first show the existence of a unique solution to the aforementioned optimization problem and then use a Lagrangian multiplier method to solve the optimization problem in (Eq. 4.7).

First, we show the existence of a unique solution for the optimization problem. In (Eq. 4.7), the energy expression is a linear combination of $[f(w)]^2$. Since the form of x^2 is a convex function, our objective function (Eq. 4.7) is also a convex function. It is clear that the two constrains in (Eq. 4.8) are both convex sets [71]. Therefore, we can conclude that there exists a unique solution for the convex optimization problem.

Second, we use the Lagrangian multiplier method to solve the optimization problem

in (Eq. 4.7). The Lagrangian function is given by

$$\begin{aligned} L(f(w),\lambda) &= \sum_{w=1}^{W_\rho} F^c(w)[f(w)]^2 + \lambda(\sum_{w=1}^{W_\rho} \frac{1}{f(w)} - T) \\ &= \sum_{w=1}^{W_\rho} \{F^c(w)[f(w)]^2 + \frac{\lambda}{f(w)}\} - \lambda T. \end{aligned}$$

Using KKT condition, the optimization problem must satisfy the following conditions,

$$\frac{\partial L(f(w),\lambda)}{\partial f(w)} = 2F^c(w)f(w) - \frac{\lambda}{[f(w)]^2} = 0 \quad \text{(Eq. 4.9)}$$

$$\frac{\partial L(f(w),\lambda)}{\partial \lambda} = \sum_{w=1}^{W_\rho} \frac{1}{f(w)} - T = 0. \quad \text{(Eq. 4.10)}$$

From (Eq. 4.9) we can obtain

$$f^*(w) = \{\frac{\lambda}{2F^c(w)}\}^{1/3}. \quad \text{(Eq. 4.11)}$$

Substituting (Eq. 4.11) into (Eq. 4.10), we can obtain

$$(\frac{\lambda}{2})^{1/3} = \frac{\sum_{w=1}^{W_\rho}[F^c(w)]^{1/3}}{T}. \quad \text{(Eq. 4.12)}$$

Therefore, substituting (Eq. 4.12) into (Eq. 4.11), the optimal CPU frequency scheduling policy is given by

$$f^*(w) = \frac{\theta}{T[F^c(w)]^{1/3}}, \quad \text{(Eq. 4.13)}$$

where

$$\theta = \sum_{i=1}^{W_\rho}[F^c(i)]^{1/3}. \quad \text{(Eq. 4.14)}$$

Substituting (Eq. 4.13) into (Eq. 4.6), we obtain the expected optimal energy consumption as

$$\begin{aligned} \varepsilon_c^* &= \frac{\kappa}{T^2}\{\sum_{i=1}^{W_\rho}[F^c(i)]^{1/3}\}^3 \\ &= \frac{\kappa}{T^2}\theta^3. \end{aligned} \quad \text{(Eq. 4.15)}$$

In this research, we also consider a benchmark scheduling policy, i.e. a brute force approach that adopts a flat frequency scheduling. The brute force approach has the same amount of pre-deadline workload with our proposed optimal DVS scheduling. The lowest frequency for the flat frequency scheduling scheme is

$$f^F(w) = W_\rho/T. \tag{Eq. 4.16}$$

In this case, the minimum energy consumption is

$$\mathcal{E}_c^F = \kappa W_\rho^3/T^2. \tag{Eq. 4.17}$$

It will be shown in the next section, for a probabilistic workload, our optimal DVS frequency allocation can conserve considerable energy most time, compared to the flat frequency scheduling.

4.2.2 Optimal DVS Scheduling Characteristics

In this subsection, we investigate the characteristics of the optimization solution, including the energy saving for different workloads and the optimal DVS scheduling relationship between different workloads.

First, let us consider the energy saving performance. We need to know the potential improvement capability of energy saving under different workload (e.g., μ, σ combinations). Compared to the flat frequency scheduling policy, the energy saving of our proposed optimal DVS scheduling policy, denoted as

$$\delta = \frac{\mathcal{E}_c^F - \mathcal{E}_c^*}{\mathcal{E}_c^F}, \tag{Eq. 4.18}$$

is plotted in Fig. 4.1, as a function of the task completion probability, for different variance-to-mean ratios ($\eta = \sigma/\mu$). We can see that the energy saving increases with the increasing of the task completion probability. Moreover for larger variance-to-mean

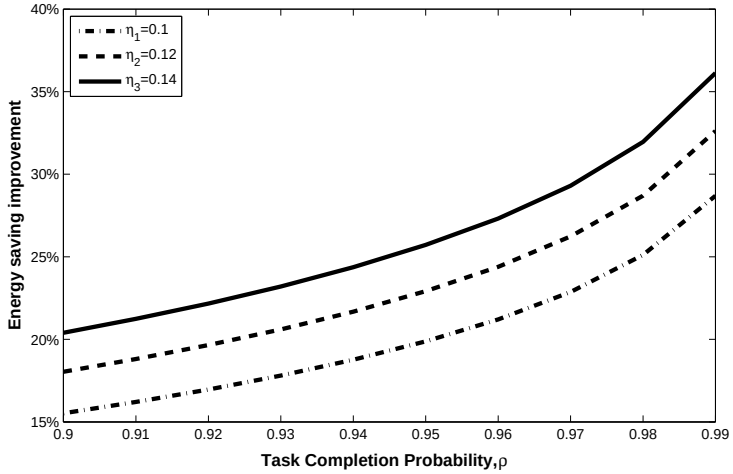


Figure 4.1: The energy saving improvement compared to flat scheduling. $\eta = \sigma/\mu$, and η_1, η_2, η_3 are respectively $0.1, 0.12, 0.14$.

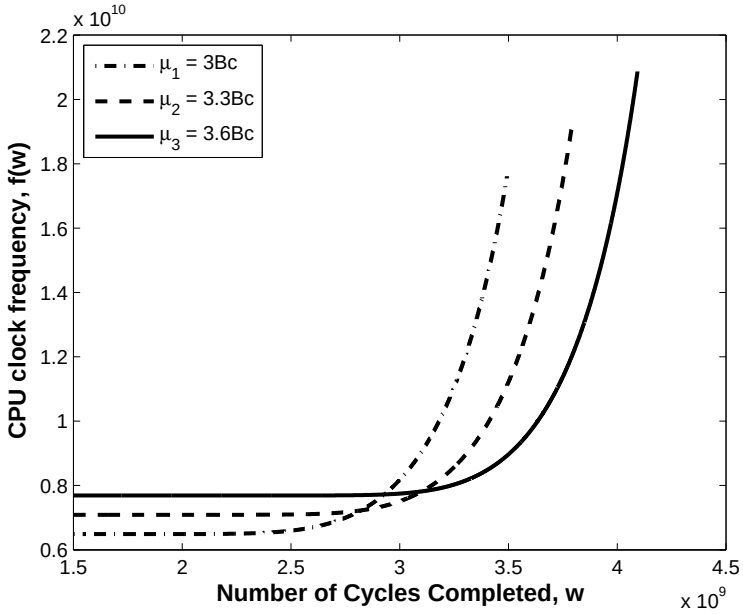


Figure 4.2: The optimal clock frequency scheduling. The mean of each workload are respectively $\mu_1 = 3Bc$, $\mu_2 = 3.3Bc$, $\mu_3 = 3.6Bc$, while standard derivation is fixed to $\sigma = 0.3Bc$, and deadline is $T = 0.5s$. (Bc denotes Billion cycle)

ratio, we can obtain more energy saving, which means higher variance of workload can potentially result in more energy saving.

Second, let us consider the optimal clock-frequency policy. In Fig. 4.2, we compare the optimal clock-frequency policies for three different means of the workload cycles (μ), with the same variance of the workload cycles (σ^2). We observe that the shape of frequency scheduling curves for different μ is similar to each other, as shown analytically next.

Let us consider two workloads of WL_1, WL_2, with $\mu_1 > \mu_2$ and $\sigma_1 = \sigma_2$. In normal distributions, the PDF curve of WL_1 can be obtained by right shifting the WL_2's curve with a distance of $\Delta\mu = \mu_1 - \mu_2$. The CDF and CCDF curves of them follow the same shifting rule. As a result, we can get the relationship,

$$F_2^c(w) = F_1^c(w + \Delta\mu). \qquad \text{(Eq. 4.19)}$$

Since the shapes of CCDF for two distribution have a shifting relationship, the difference between θ can be derived as follows,

$$\begin{aligned} \theta_1 - \theta_2 &= \sum_{i=1}^{W_{\rho_1}} [F_1^c(i)]^{1/3} - \sum_{i=1}^{W_{\rho_2}} [F_2^c(i)]^{1/3} \\ &= \Delta\mu = \mu_1 - \mu_2. \end{aligned} \qquad \text{(Eq. 4.20)}$$

Using this result, we can obtain the following relationship between their corresponding clock-frequency policies,

$$\begin{aligned} f_2^*(w) &= \frac{\theta_2}{T[F_2^c(w)]^{1/3}} \\ &= \frac{\theta_2}{T[F_1^c(w + \Delta\mu)]^{1/3}} \\ &= \frac{\theta_1 - \Delta\mu}{\theta_1} f_1^*(w + \Delta\mu). \end{aligned} \qquad \text{(Eq. 4.21)}$$

Therefore, the optimal frequency scheduling vector of WL_2 can be calculated as a left shift $\Delta\mu$ with a scale of $\frac{\theta_1 - \Delta\mu}{\theta_1}$ from the scheduling vector of WL_1.

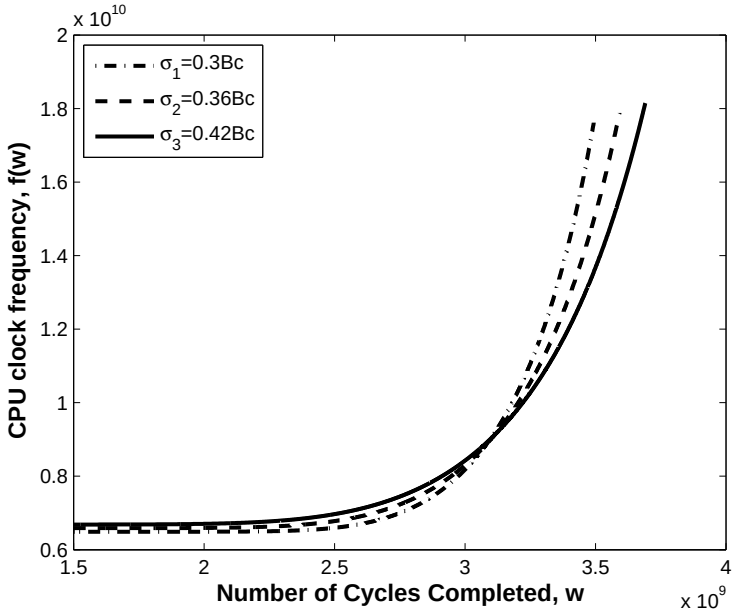


Figure 4.3: The optimal clock frequency scheduling. The standard derivation of each workload are respectively $\sigma_1 = 0.3Bc$, $\sigma_2 = 0.36Bc$, $\sigma_3 = 0.42Bc$, while mean of cycles is fixed to $\mu = 3Bc$, and deadline is $T = 0.5s$.

Using the shift-and-scale property, we can reduce the complexity of our proposed optimal DVS scheduling algorithm significantly. Specifically, for a given workload distribution, the optimal DVS scheduling policy, as denoted in (Eq. 4.13), can be stored in a table. For practical encoding applications, the practical clock frequency scheduling vector can be obtained by the shift-and-scale approach. Therefore, the shift-and-scale relationship could be used to simplify the frequency optimization algorithm on practical platforms.

Finally, let us investigate the impact of variance on the optimal scheduling policy. In Fig. 4.3, three frequency scheduling curves are plotted for different variance σ^2 while the same mean μ. It can be seen that, the shape varies for different variances, and larger variance leads to earlier frequency acceleration.

In summary, owing to these characteristics, our optimal DVS solution can be implemented with a low complexity. The frequency expression in (Eq. 4.13) is a light weight

computation process, since it can be tabulated. In addition, the curve shifting relationship between different workload can reduce the complexity. Therefore, our proposed algorithm is suitable for mobile devices with a limited energy budget.

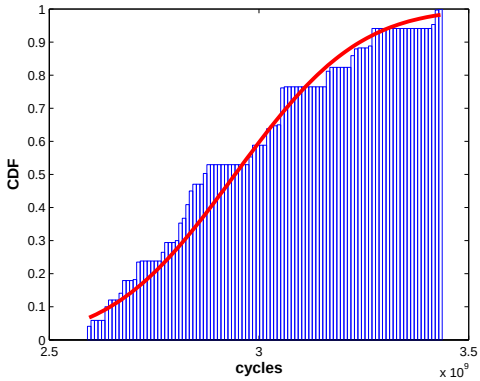
4.3 DVS Application to Real-Time Video Encoding

In this section, we apply our proposed optimal DVS scheduling algorithm to real-time video encoding. Firstly, we evaluate the workload characteristics of H.264/AVC video encoding, and then apply the optimal DVS scheduling policy for it and compare the energy consumption with different distribution estimation configurations.

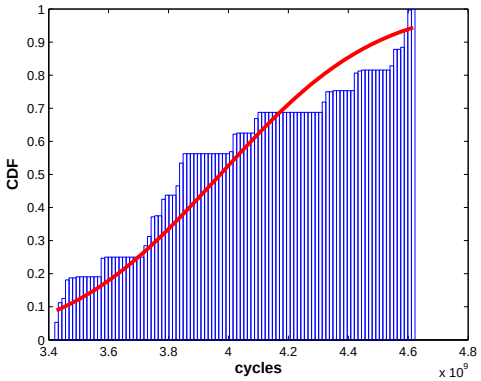
4.3.1 Empirical Workload Distribution for H.264/AVC Coding

In a real-time video encoding system, the real-time management unit can be selected as a single frame or a batch of frames. Considering the high complexity of encoding management, we set a batch of frames encoding process (i.e. encoding of a GOP) to be an individual task as the minimum real-time encoding unit. Several CIF size sample videos are used as test sequences and encoded with x264 software [72], which is a high performance open source H.264/AVC encoder. The raw video sequences are compressed with frame structure of I-B-P-B-P-B...(GOP-16), a frame rate of 25 fps and quantization parameter of 30. The encoding experiments run on a 3GHz Intel Core Duo CPU. The number of cycles consumed is collected via Oprofile tools [73]. The x264 platform-specific assembly optimization is disabled for platform-independent results.

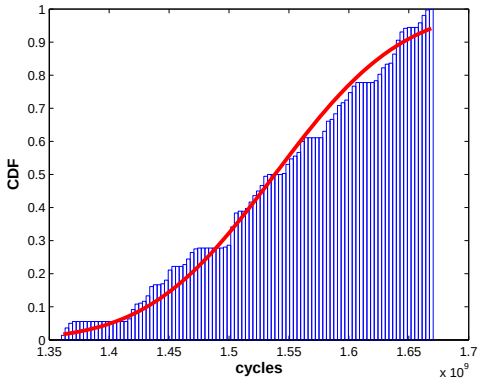
In Fig. 4.4, we plot the histograms of the per-GOP CPU cycles consumed for encoding four different video sources. The resulted histograms are curve-fitted with a (truncated) normal distribution CDF. It can be seen that video encoding workload can be well modeled with a normal distribution.



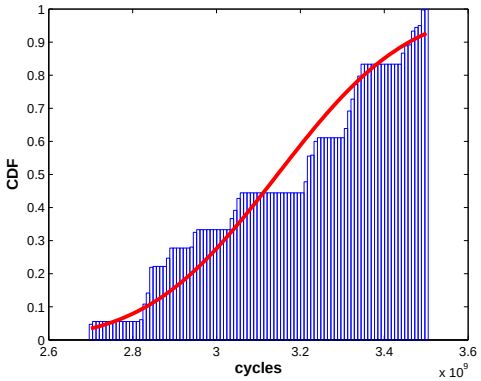
4.4.a: Foreman $\mu = 3Bc, \sigma = 0.23Bc$



4.4.b: Football $\mu = 4Bc, \sigma = 0.41Bc$



4.4.c: Akiyo $\mu = 1.5Bc, \sigma = 0.08Bc$



4.4.d: Mobile $\mu = 3.1Bc, \sigma = 0.24Bc$

Figure 4.4: Per-GOP CPU cycle consumption histogram and normal distribution CDF fitting for four different videos

4.3.2 Optimal Clock Frequency Configuration and Energy Consumption

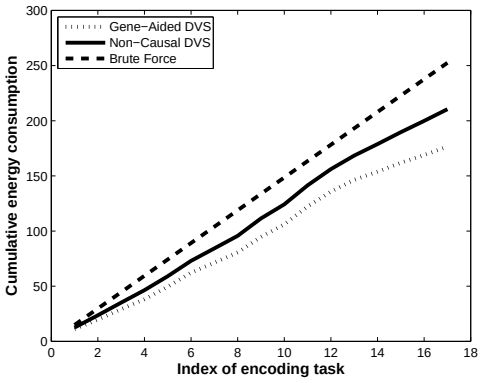
In this subsection, we first compare the energy performance of the following three scheduling algorithms.

(i) Brute-Force Scheduling Algorithm: in this case, the scheduler has access to the overall distribution of all the GOP-encoding tasks (a non-causal estimator) and applies a flat frequency-scheduling policy to meet the encoding delay deadline;

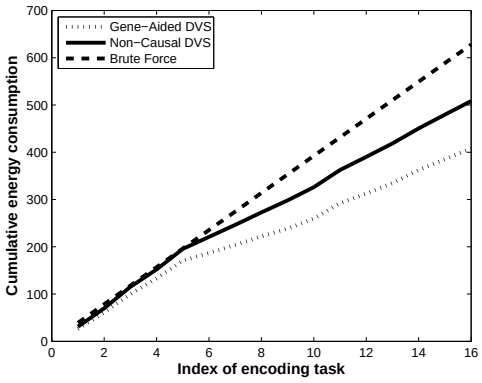
(ii) Gene-Aided DVS Algorithm: in this case, the scheduler knows the exact number of CPU cycles consumed to encode each GOP and applies a flat-frequency scheduling policy to meet the encoding delay deadline;

(iii) Non-Causal DVS Algorithm: in this case, the scheduler has access to the overall workload distribution for all the GOP-encoding tasks (a non-causal estimator) and applies our proposed DVS policy to meet the encoding delay deadline with a target task completion probability.

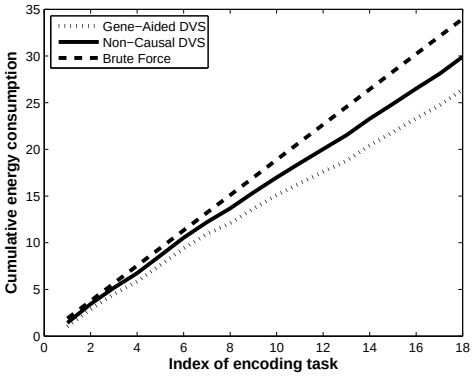
In our first experiment, the task completion probability is set to be 95%. For both the brute force algorithm and the Non-Causal DVS algorithm, the workload distribution is obtained from global experiment data. In Fig. 4.5, we present the simulation results of cumulative energy consumption for the three scheduling algorithms, as a function of the number of encoded GOPs. We first notice that our proposed non-causal DVS algorithm achieves a significant gain compared to the brute Force scheduling algorithm. However, our proposed non-causal DVS algorithm experiences some performance penalty from the Genie-Aided DVS algorithm. The average energy saving gain is illustrated in Table 4.1. We can see that the non-causal DVS algorithm can achieve an energy saving gain of $10\% - 20\%$, and the Gene-Aided scheduling algorithm can provide a gain of $20\% - 35\%$.



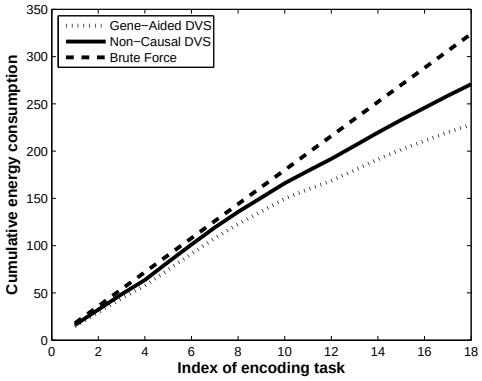
4.5.a: Foreman



4.5.b: Football



4.5.c: Akiyo



4.5.d: Mobile

Figure 4.5: Cumulative energy consumption for three alterative DVS policies

Table 4.1: Energy Saving Over Brute-Force Algorithm

Algorithm	Foreman	Football	Akiyo	Mobile
Gene-Aided DVS	29.7%	35.2%	22.2%	29.4%
Non-Causal DVS	16.4%	18.9%	11.8%	16.2%

Table 4.2: Energy consumption gap compared to Gene-Aided DVS

Video	Non-causal	Recent-k				
		k=3	k=6	k=9	k=12	k=15
Foreman	18.9%	15.9%	17.7%	16.9%	16.7%	17.1%
Football	25.1%	14.8%	22.4%	24.5%	25.9%	27.2%
Akiyo	13.3%	11.3%	14.2%	14.5%	15.1%	15.4%
Mobile	18.7%	10%	13.9%	16.8%	18.5%	18.5%

Our second experiment addresses the issue of workload estimation for DVS. In reality, the non-causal estimator for the workload distribution is infeasible. A more practical approach is to adopt a causal estimator for the workload distribution. In [55], a few sampling methods were investigated for such a purpose. In this chapter we adopt a *Recent-k method* to estimate the empirical workload distribution and evaluate its impact on the energy consumption for real-time video encoding on mobile devices. Specifically, the *Recent-k method* uses the sample of the k most recent encoding tasks to estimate the workload distribution (i.e., the mean and the variance of an embedded Gaussian distribution). The energy consumption penalty ($\delta = (E_{DVS} - E_{Gene})/E_{Gene}$) against the Gene-Aided DVS algorithm is used for performance evaluation. We calculate the energy consumption penalty for the *Recent-k* method and the non-causal DVS algorithm in Table 4.2 for $k = 3, 6, 9, 12, 15$. It can be observed that the estimator using the shortest history ($k = 3$ scheme) can achieve more energy saving compared to long history estimation (Global estimation), and thus is more suitable for video encoding. It can be understood as follows. In real-time video encoding application, due to the dynamics of video motion, a current frame will only take some of neighboring frames as reference, which leads to the strong correlation on neighboring frames encoding. Therefore, it is reasonable that estimation based on a shorter history can conserve more energy, especially for the fast motion videos, such as "Football".

4.4 Summary

In this chapter we investigate the problem of minimizing energy consumption for real-time video encoding on mobile devices via the DVS technology. The problem is formulated as a constrained optimization problem. Under a probabilistic workload model, we obtain closed-form solutions for both the optimal clock frequency configuration and the resulted minimum energy for GOP encoding. Numerical results indicate that our derived optimal solution outperforms the brute-force approach significantly. Moreover, we apply the optimal solution for real-time H.264/AVC video encoding application. Our numerical results suggests that an energy saving of $10\% - 20\%$ can be achieved, compared to the flat clock frequency scheduling.

Chapter 5

Adaptive Job Scheduling for Cloud-based Video Transcoding

In this chapter, we focus on exploring the job adaptation solutions for the job scheduling problem in cloud-based video transcoding system. A study on the trade-off between completion delay and transcoding QoS is conducted. The proposed system serves as a media transcoding service provider which collects the uploaded video streams from content providers and streams out the transcoded video packets to the Internet subscribers. When system capacity cannot afford the increasing job backlog, lower-complexity transcoding configuration is applied to the job to meet the completion delay requirement. The Lyapunov optimization framework [23] is applied as the problem solver, due to its low complexity and efficient performance. It adaptively sets up the transcoding job to guarantee the queue backlog bounded, also taking into account the trade-off of job accomplishment delay and QoS of transcoding. The proposed scheduling policy tries to minimize the time-average bit-rate of the generated video stream, while keeping delay requirement. The simulated performance of real trace is measured. Compared to the static configuration strategy, the proposed framework obtains smooth coding QoS degradation when system load becomes heavy and reports better performance between each of the two static configuration points.

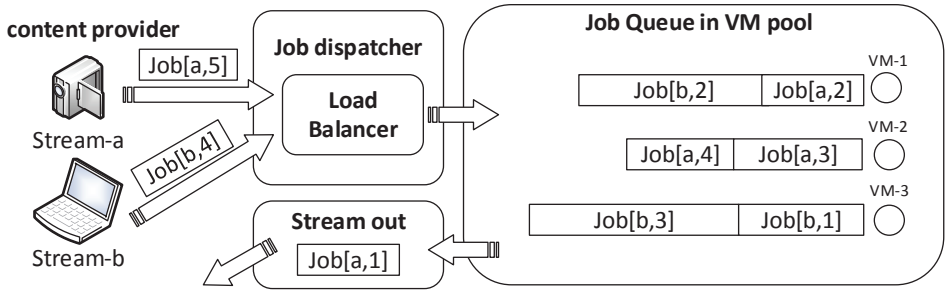


Figure 5.1: The cloud transcoding system framework

5.1 System Models

5.1.1 System Overview

To provide video transcoding as a service, the cloud-based system requires to solve two major problems. One is how to schedule the transcoding jobs including the job dispatching and job configuration. Another one is how to manage the computing resource in the system so as to guarantee the QoS and ensure the computing capacity auto-scaling to adapt to the dynamics of job's arrival rate and resource consumption. In this chapter, we mainly focus on the transcoding job scheduling algorithm. The proposed system architecture of the job scheduling is shown in Fig. 5.1. In our scenario, a certain duration length of video segment files from a content provider (such as Stream-a and Stream-b) are uploading to the cloud transcoding system. Upon arriving at the platform over Internet, each job is dispatched to a virtual machine (VM)[1] within cloud to be proceeded. The load balance among VM is required in job dispatcher. After allocated to the VM, the job will be buffered to the affiliated first-come-first-served (FCFS) queue to wait for transcoding service if VM is busy. Once a VM is free to process the next job, the first job in its Queue will start the transcoding process with an optimized configuration.

[1]The VM is similar to the instance of Amazon EC2

5.1.2 Job Arrival and Dispatch

In practical environment, the job arrival is discrete and random at any time. To simplify the discussion, in the following parts, we analyze the problems within a discrete time slot based system model. It supposes the job arrival event is triggered at the beginning of each time slot, and the duration of the slot is very short.

The admitted video transcoding jobs are heterogeneous on the completion delay requirement and computing resource consumption. Assume there are total N VMs in the resource pool. Upon the arrival of a job, the job dispatcher will allocate one VM from N to handle the processing. The job will be buffered to the according job queue $Q_i(i \in 1, ..., N)$ affiliated to VM_i. To keep the load balance among the VMs, the dispatching follows the shortest queue backlog first policy. The queue backlog is defined as the accumulated computing time required.

As we discussed previously, the video segment files arrive at the cloud system discretely with a certain time interval. We assume that the content is transmitted over HTTP, and the segment duration and transmission interval is fixed to 5 or 10 seconds length. In reality, due to the network bandwidth dynamics and congestion, the arrival interval of segment files will be fluctuated. However, without loss of generality we neglect the effect of transmission jitter, since the proposed job scheduling algorithm is independent of the job arrival pattern. In each discrete time slot, we further assume that there is at most one video segment arrival event in a queue.

5.1.3 Job Adaptive Configuration

For live transcoding application, each job has a soft completion deadline constraint to ensure the continuously streaming the transcoded segments to the subscribers. To fit the fluctuating arrival workload, a dynamic scaling of computing capacity is desired,

so as to guarantee the QoS and soft completion deadline during peak workload and prevent from resource over provisioning during low workload. However, the long VM system initialization time makes it difficult to scale up computing capacity in time to accommodate the bursty workload. Due to the significant lag of VM capacity scale-up, we assume the VM computing capacity for a long period of time is constant and limited even if the system is overload. In this condition, to maintain the bursty video transcoding QoS will cause the queue backlog tends to infinite which dissatisfies the job completion delay requirement.

Adaptive configuration of video transcoding is a feasible approach to solve the problem, since it makes a trade off between the video transcoding QoS and the job completion delay. Let $a(t)$ denote the transcoding configuration decision variables ('preset' parameters) at time slot t. The action variable $a(t)$ is chosen from $\mathcal{A} = \{0, 1, ..., L-1\}$ which represents the alternative L types of transcoding presets. The transcoding preset configuration determines the video coding complexity including B frames number, maximum reference picture number, subpixel estimation complexity, etc. For the fastest speed configuration $a(t) = 0$, the bit-rate of the generated video is expected to be larger than others for the same quantization parameter (QP), since coarse coding algorithm can only exploit limited temporary and spatial redundancy. For the slowest configuration $a(t) = L - 1$, much longer time will be spent on transcoding.

We define the job's reference workload size s_r as the transcoding time on the reference VM with the default transcoding preset $r (r \in \mathcal{A})$ configuration. The workload $c_{a(t)}$ executed within a time slot for the action $a(t)$ is calculated as,

$$c_{a(t)} = t_{slot} \cdot \frac{s_{a(t)}}{s_r}, \qquad \text{(Eq. 5.1)}$$

where t_{slot} is the time slot length, and $s_{a(t)}$ is the workload size with $a(t)^{th}$ preset configuration. When $s_{a(t)} = s_r$, the executed workload length is one time slot.

5.1.4 Transcoding QoS and Resource Consumption

In the video transcoding-streaming application, the QoS depends on many factors such as the quality of the compressed video, the average output bit-rate, etc. Here we consider a simplified QoS model, which is only related to the output bit-rate. With the unique video coding quality parameter configuration, higher compression ratio leads to better QoS for the video transcoding-streaming application. We define the QoS function as the bit-rate reduction. Then the optimal QoS equals to the minimization of the generated bit-rate.

Since the video transcoding is a CPU intensive application, the major resource consumed is CPU cycles. To achieve better QoS, more computing time might be spent on long term motion estimation and mode decision, however it may lead to system overload and long job completion delay. We need to find out the mapping relationship from CPU cycles to the transcoding QoS and completion delay.

Due to the complexity of video content characteristics classification and video coding modeling, it is hard to directly derive an exact complexity-rate-distortion (C-R-D) model for the resource consumption. Here we use a simple way to estimate the transcoding resource consumption.

Along with the video file, a certain codec knowledge is encapsulated, including the original framerate, resolution, QP and coding complexity preset, etc. It is easy to extract the original codec information from media. By leveraging the embedded codec knowledge, we can estimate the required computing resource for different transcoding configurations.

5.2 Problem Formulation and Solution

In this section, we formulate the job scheduling problem. The Lyapunov optimization framework is applied to solve the problem.

5.2.1 Optimization Problem

At each time slot, if no job is running in the VM, the video transcoding scheduler will fetch a job from the FCFS queue. Then it selects a transcoding preset to configure the job and executes the transcoding process. In our proposed scheduler, we aim to minimize the time-average video transcoding bit-rate subject to the constraints of queue backlog stability, which guarantees the job completion delay requirement.

Based on the discussed system model, we define the time-average bit-rate of the transcoding output as,

$$\overline{\mathbf{D}} \triangleq \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} D(t), \tag{Eq. 5.2}$$

where $D(t)$ is the output bit-rate at time slot t. Because the transcoding statistics information can be obtained only upon the job completion in practical situation, it can also be expressed as taking the average value over the sum of all discrete video segment bit-rate.

As discussed previously, we cannot greedily choose the highest compression ratio configuration which may cause the system overloaded, since the queue backlog Q will tend to be unbounded. The time-average queue backlog is defined as,

$$\overline{\mathbf{Q}} \triangleq \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{|Q(t)|\}. \tag{Eq. 5.3}$$

At each time slot t, the system makes an online decision of $a(t)$ to determine which transcoding preset is configured. The problem is formulated as,

$$\min_{a(t)} \quad \overline{\mathbf{D}}, \tag{Eq. 5.4}$$

$$\text{s.t.} \quad \overline{\mathbf{Q}} < \infty, \tag{Eq. 5.5}$$

$$a(t) \in \mathcal{A} \forall t, \tag{Eq. 5.6}$$

where the queue stability is ensured by (Eq. 5.5).

5.2.2 Job Scheduling Solution

To solve the problem defined in (Eq. 5.4), the Lyapunov optimization approach can be applied. By leveraging the Lyapunov framework, the constraint (Eq. 5.5) is converted to queuing style. The workload backlog $Q(t)$ is updated when a new job arrives or the queuing job is executed as,

$$Q(t+1) = max[Q(t) - c_a(t), 0] + \lambda(t), \tag{Eq. 5.7}$$

where $\lambda(t)$ is the workload size of new job arrival in the time slot, and $c_{a(t)}(t)$ refers to the workload executed in the time slot.

To solve the problem, we use the min drift-plus-penalty approach in the following. Firstly we define the quadratic Lyapunov function as,

$$L(Q(t)) \triangleq \frac{1}{2}Q(t)^2, \tag{Eq. 5.8}$$

and the *conditional Lyapunov drift* to describe the variation of queue $\mathbf{Q}$ backlog,

$$\Delta(Q(t)) \triangleq \mathbb{E}\{L(Q(t+1)) - L(Q(t))|Q(t)\}. \tag{Eq. 5.9}$$

A small value of $L(Q(t))$ implies that all the queue backlogs are small; otherwise there is at least one queue that is congested and increasing unboundedly. The drift $\Delta(Q(t))$ is the expected change in the Lyapunov function over one time slot, given the current queue backlog state $Q(t)$ in time slot t.

To exploit the tradeoff between queue backlog and compression performance, we follow the Lyapunov approach to define a *drift-plus-penalty* function,

$$\Delta(Q(t)) + V\mathbb{E}\{D(t)|Q(t)\}, \tag{Eq. 5.10}$$

where scalar parameter $V \geq 0$ is a constant value to control the tradeoff. By setting a larger value of V, the time-average generated bit-rate will become close to the optimal

bound. On the other hand, with smaller value of V, the system can keep the queue at lower backlog, which leads to lower job completion delay.

In the Lyapunov optimization, the *drift-plus-penalty* has the following upper bound [23] for all t and all parameters $V \geq 0$,

$$\begin{aligned} F &= \Delta(Q(t)) + V\mathbb{E}\{D(t)|Q(t)\} \\ &\leq B + V\mathbb{E}\{D(t)|Q(t)\} \\ &\quad +Q(t)\mathbb{E}\{\lambda(t) - c_a(t)|Q(t)\}, \end{aligned} \tag{Eq. 5.11}$$

where B is a positive constant that satisfies the following for all time slot,

$$\begin{aligned} B &\geq \frac{1}{2}\mathbb{E}\{\lambda(t)^2 + c_a^2(t)|Q(t)\} \\ &\quad -\mathbb{E}\{\widetilde{c}_a(t)\lambda(t)|Q(t)\}, \end{aligned} \tag{Eq. 5.12}$$

where $\widetilde{c}_a(t) = \min[Q(t), c_a(t)]$. Such a constant B exists because the arrival is i.i.d.

Rather than directly minimize the drift-plus-penalty of (Eq. 5.10), our strategy is to minimize the bound given in the right-hand-side of (Eq. 5.11). Since $\lambda(t)$ is independent of action $a(t)$, at each time slot we only need to minimize the $a(t)$ associated part, which is,

$$\begin{aligned} \min_{a(t)} \quad & VD(t) - Q(t) \cdot c_a(t) \\ \text{s.t.} \quad & a(t) \in \mathcal{A} \ \forall t \ . \end{aligned} \tag{Eq. 5.13}$$

The dispatching strategy is expressed in the algorithm 1.

5.3 Simulation

In this section, we study the real trace statistics first, and evaluate the basic performance of our proposed algorithm. Then a further analysis is conducted for the performance under different system workload.

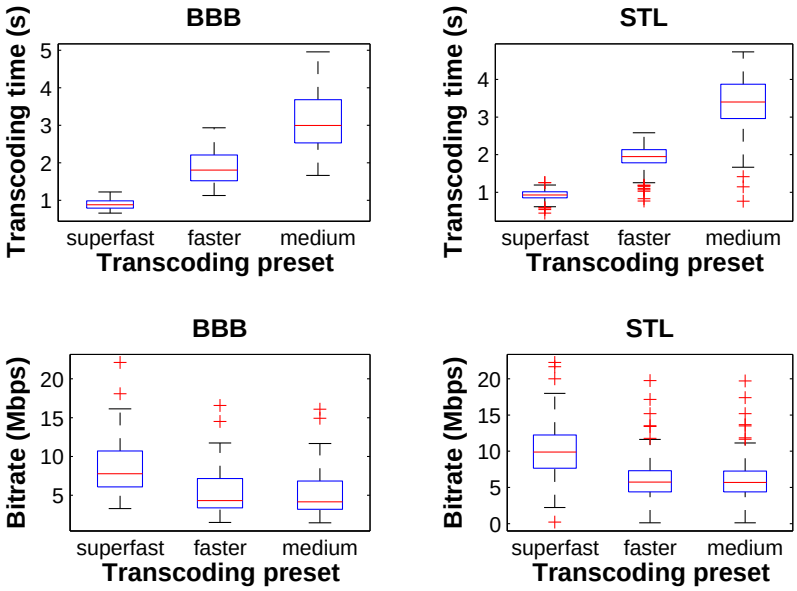


Figure 5.2: Transcoding performance statistics of computing time and generated bit-rate under three presets (superfast, faster and medium). The box is the range of estimation, where the horizontal line within the box is the median value. The horizontal lines of upper and lower outside the box are the boundary value, while the 'plus' symbol denotes the out of range result values.

Table 5.1: The statistics of transcoding performance

Algorithm	Transcoding time(s)			Output bit-rate(Mbps)		
	BBB	STL	Mix	BBB	STL	Mix
S:superfast	0.899	0.915	0.908	8.525	10.028	9.348
S:faster	1.861	1.917	1.892	5.338	6.077	5.743
S:medium	3.094	3.329	3.222	5.138	6.076	5.652

5.3.1 Real Trace

To verify the model and our proposed optimization, we choose two 10 minutes 1080p full-HD videos, "big buck bunny" (BBB) and "Sintel" (STL), as real trace to do the simulation. The original video is encoded with MPEG-4 codec and segmented into 5 second duration video files. Since the compression ratio is used to represent the QoS, the objective video quality parameter (QP) is fixed. We choose three different presets, 'superfast', 'faster' and 'medium', as the coding configuration choices. All real trace data are generated on a computer with a 3.2GHz Xeon CPU E5-1650.

We can see the transcoding performance statistics in Fig. 5.2. From the statistics, we see that, sometime 'faster' preset may cost less time than 'superfast' preset and obtain smaller bit-rate than 'medium' preset. However, on average, the trend of transcoding time is nondecreasing when we select more complex preset, and the trend of generated bit-rate is decreasing and converging to a lower-bound.

The detail statistics are summarized in Table 5.1. By mixing videos BBB and STL, we get another trace (Mix). In Mix trace, with respect to the performance of the 'superfast'

Algorithm 1 Adaptive Job Scheduling

procedure UPDATE
  $Q(t+1) \leftarrow max[Q(t) - c_a(t), 0] + \lambda(t)$
end procedure

procedure SCHEDULING
  for $i = 1, ..., N$ **do** // for each queue
    for $a(t) = 1, ..., k$ **do** // for each mode
      $X(a(t)) \leftarrow V \cdot D(t) - Q(t) \cdot c_{a(t)}(t)$
    end for
    use $a(t)$ where $X(a(t))$ is minimal
    update $Q(t+1)$
  end for
end procedure

preset, 'faster' preset pays 1.1 times cost on transcoding time and gets about 38% bit-rate reduction, while 'medium' preset pays 2.6 times cost on transcoding time, but only gets about 40% bit-rate reduction. We can see that the bit-rate reduction is not linear with the transcoding time consumption. Therefore, the trade-off between transcoding delay and bit-rate reduction is critical for the job scheduling.

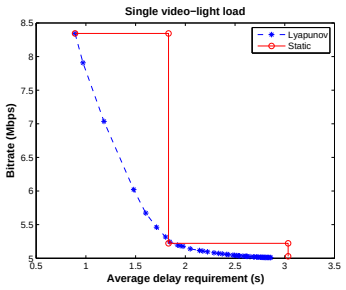
5.3.2 Simulation Setup

We construct a discrete event simulator to evaluated the performance. In the simulation, a static job dispatching strategy is used as baseline, which dispatches all queues fairly and uses a dedicated transcoding preset without aware of the video content and the job arrival rate. The video stream is split into 5 second segments, and pre-encoded in MP4 format.

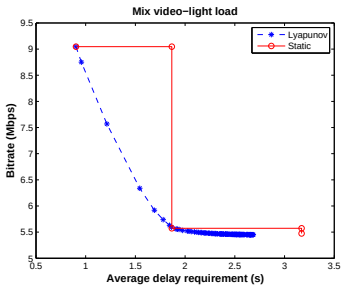
Two parts of simulations are conducted, both taking totally 1000 seconds long running time. In the first part, the interval of the video segment arrival on cloud system is set to 5 seconds, which equals the video segment duration. The video segments are repeatedly sent to the cloud system. In the second part, different system load is considered. The job arrival interval is varying from 3 to 6 seconds.

5.3.3 Result Analysis

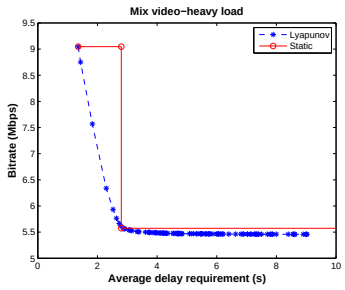
Fig. 5.3 shows the performance comparison with the static scheduling algorithm. The static scheme can only fix to one of the three presets. When the job delay constraint is between the average delay performance of two presets, the algorithm can only choose the faster one. However, the proposed algorithm can adaptively choose the preset between the two presets. From the figure, it can be seen that in the static preset point, the proposed algorithm achieves the same performance, while smooth bit rate gain is achieved between



5.3.a: BBB with one server

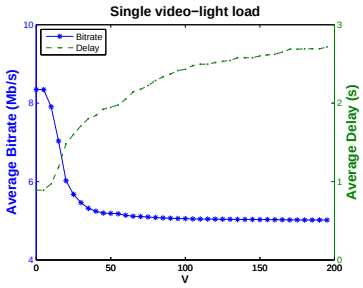


5.3.b: Mix with two servers

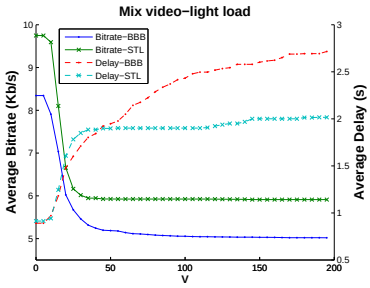


5.3.c: Mix with one server

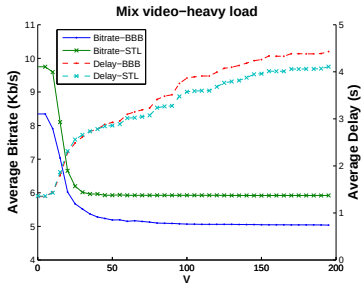
Figure 5.3: Performance comparison of static scheduling and adaptive scheduling



5.4.a: BBB with one server



5.4.b: Mix with two servers

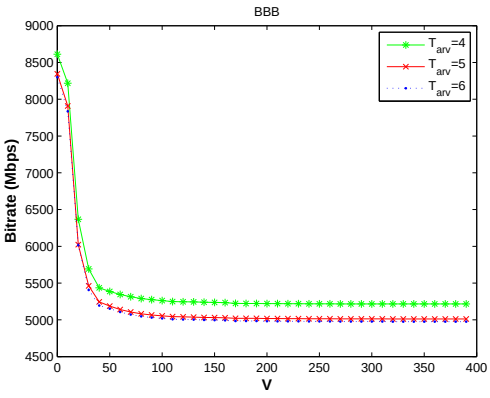


5.4.c: Mix with one server

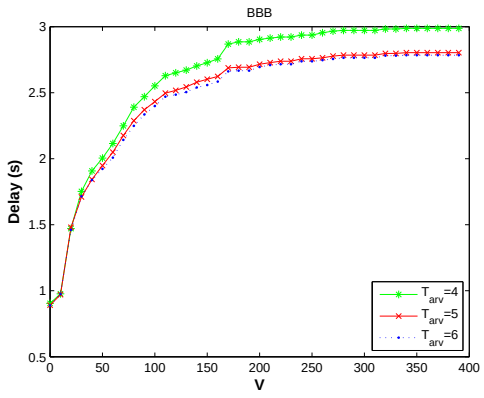
Figure 5.4: Backlog-Bitrate performance for parameter V

any two adjacent preset points. In Fig. 5.3.c, when system is heavily loaded, the static algorithm with "medium" preset suffers a very long delay of 142 seconds for a 5.47$Mbps$ bit-rate. In contrast, the proposed algorithm achieves the same bit-rate with about 5 seconds delay only.

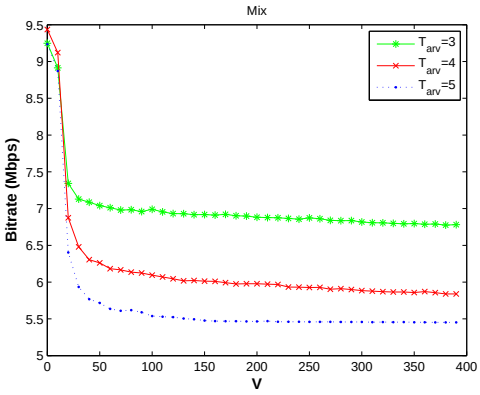
Fig. 5.4 shows the trade-off curve of the proposed algorithm. For the single video case, when trade-off parameter V goes to more than 100, the bit-rate is approximated to the lowest. Moreever, even if the V goes to infinity, the delay is bounded within 3 seconds. That is because the system load is light. Even if all video segments are transcoded in 'medium' preset, the queue backlog is still bounded. For the mix video case, we see the bit-rate curves of the two videos are different from each other. When



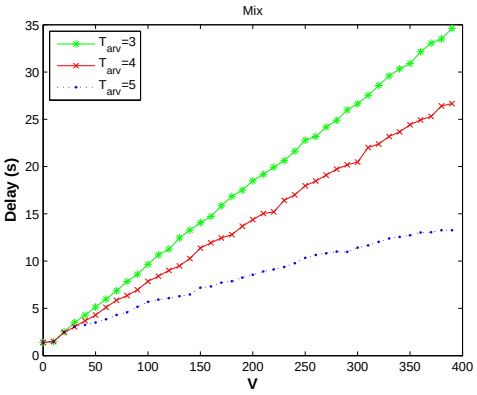
5.5.a: Bit-rate.BBB



5.5.b: Delay.BBB



5.5.c: Bit-rate.Mix



5.5.d: Delay.Mix

Figure 5.5: Performance with different system loads

system load is heavy, the delay curve is close to each other, since all jobs are equally processed. However, in the light load condition, we see the delay curves for two videos reach saturation at different V. This is because the algorithm can only control the overall average performance, instead of the individual job.

In the second part, we analyze the performance under varying system loads. Fig. 5.5.a and Fig. 5.5.b show the bit-rate and delay curves for single video scenario. Three job arrival interval $T_{arv} = 4, 5, 6$ conditions are studied. It can be seen that the bit-rate

curve of $T_{arv} = 4$ is higher than the other two curves, which means the algorithm cannot achieve lower bit-rate while still keeping the same average delay bounded. Fig. 5.5.c and Fig. 5.5.d are the results of serving two concurrent video streams in one server. It shows the similar trend that when system load becomes heavier.

5.4 Summary

This chapter has presented an approach for video transcoding job scheduling on cloud. The scheduling strategy tries to minimize the long-term time-average generated bit-rate of video transcoding under the average job delay constraint. By applying Lyapunov approach, the arrived jobs are adaptively configured under the constraint of queue backlog stability. The tradeoff between job delay and generated bit-rate is exploited. Through the tradeoff, the system can achieve better performance than static scheduling at the most of the delay constraint requirement points.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

The objective of this thesis is to study the advanced scheduling for wireless data transmission and cloud media computing.

Firstly, the MAC protocol of underwater networks is chosen as a study case to investigate how to overcome the drawbacks of long-RTT and low bandwidth wireless communication scenario. An MAC protocol is proposed to tackle the costly protocol handshakes by reducing the protocol handshake overhead and increasing the data transmission opportunities. An efficient dynamic polling-based handshake operation is designed which offers time-bounded collision-free channel assignment. Through extensive simulations, the results show the proposed HCFMA protocol outperforms its counterparts, in terms of throughput and delay, such as ALOHA-CS and SFAMA.

Secondly, we take video encoding as the study case to study the scheduling of the CPU running frequency on video encoding process so as to minimize energy consumption of mobile devices, while meeting the QoS requirements. A probabilistic QoS model is adopted, in which the encoding process completes with a target probability within a specified delay deadline for each GOP. The optimization problem is solved analytically

and obtain closed-form solutions for both the optimal clock frequency schedule and the minimum energy consumption. The numerical results suggest that significant amount of energy can be saved by using our optimal solution.

Lastly, the author studies the trade off between job completion delay and QoS of video transcoding in cloud system, and designs an efficient scheduling algorithm for video transcoding workload. The Lyapunov optimization framework is applied as the problem solver to the scheduler to adaptively setup the transcoding job. Compared to the static configuration strategy, the proposed framework obtains smooth transcoding QoS degradation when system load becomes heavier and reports better delay performance.

6.2 Future Directions

6.2.1 Extension of Scheduling in Data Transmission

For the wireless data transmission, the future works include various extensions of underwater MAC protocol on the aspects of QoS, protocol parameter optimization, and the study in the presence of more general multi-hop scenarios, where the sensor nodes are moving and more types of 3D topology will be used to evaluate the performance.

Considering the development of underwater sensor networks, it is possible to extend the data transmission scheduling from general sensor data to multimedia content such as the image or video captured by sensors. Many existing works have been proposed for multimedia transmission in sensor networks. However, in underwater sensor networks, it is still required to make a new design to overcome the drawbacks of long-RTT and low-bandwidth environment. Especially for some realtime multimedia applications, such as visual navigation of mobile sensor [74, 75, 76] and image analysis [77] on exploring, the end-to-end delay becomes more important, since long end-to-end delay makes it difficult to do real-time navigation and underwater environment analysis. Except for the data

transmission, other emerging technologies can also be applied to improve the QoS, such as better video/image compression algorithm by using compressed sensing theory, which eliminates more noise and redundancy.

6.2.2 Extension of Scheduling in Mobile Media Cloud Computing

For the energy efficient computing scheduling on mobile devices, in the future, we will do more effort to analyze the tight bound of energy consumption. It is known that accurate estimation of video encoding consumption is challenging. Currently, the proposed algorithm utilizes the history workload distribution as estimation, however it is not accurate which limits the algorithm deployment. In the future, we will try to study a new scheduling solution which intelligently classifies the video content on the fly and conducts the job scheduling dynamically without the history knowledge of the content.

The cloud offloading system will also be further studied to achieve high performance and energy saving. Existing offloading systems are rarely rich media concerned which may not be efficient for the considered scenarios. In the media computing, the huge size of raw data may make the cloud offloading impractical, since the overhead on data transmission will exceed the gain of workload offloading, both in energy consumption and processing time. Therefore, we need to carefully design the system. For example, in HEVC video encoding application, we may conduct a simple but fast H.264 video encoding on mobile devices first and offload the HEVC transcoding workload to the cloud system. Additionally we can try to develop a distributed video encoding framework based on the cloud-based offloading concept.

For the job scheduling of cloud media transcoding, in the future we will try to study the service pricing model [8, 78, 79]. For different computing requirements, the corre-

sponding service price will be applied based on the current system workload. The job scheduling will also be extended to support the pricing based service provisioning.

For VM resource management, in the future we will design an algorithm to automatically schedule the computing capacity scaling. Considering the revenue oriented optimization, a new design of VM resource renting scheduler will be developed for various pricing models.

Author's Publications

Journal Papers

(i) **Ming Yang**, Mingsheng Gao, Chuan Heng Foh and Jianfei Cai, "Hybrid Collision-Free Medium Access (HCFMA) Protocol for Underwater Acoustic Networks: Design and Performance Evaluation", *Oceanic Engineering, IEEE Journal of*, vol. PP, no. 99, pp. 1-11, 16 April 2014.

Conference Papers

(i) **Ming Yang**, Jianfei Cai, Weiwen Zhang, Yonggang Wen and Chuan Heng Foh, "Adaptive Job scheduling for Cloud-based Video Transcoding", in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, Lisbon, Portugal, May 2015.

(ii) **Ming Yang**, Jingjing Fu, Yan Lu, Jianfei Cai and Chuan Heng Foh, "An Adaptive Multi-Layer Low-Latency Transmission Scheme for H.264 Based Screen Sharing System", in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, Melbourne, Australia, June 2014.

(iii) **Ming Yang**, Yonggang Wen, Jianfei Cai and Chuan Heng Foh, "Energy Minimization via Dynamic Voltage Scaling for Real-Time Video Encoding on Mobile Devices", in *Proceedings of IEEE International Conference on Communications (ICC)*, Ottawa, Canada, June 2012.

(iv) **Ming Yang**, Jianfei Cai, Yonggang Wen and Chuan Heng Foh, "Complexity-rate-distortion Evaluation of Video Encoding for Cloud Media Computing", in

Proceedings of IEEE International Conference On Networks (ICON), Singapore, Dec 2011.

(v) **Ming Yang**, Mingsheng Gao, Chuan Heng Foh, Jianfei Cai, and Periklis Chatzimisios, "DC-MAC: A Data-centric Multi-hop MAC protocol for Underwater Acoustic Sensor Networks", in *Proceedings of IEEE Symposium on Computers and Communications (ISCC)*, Kerkyra, Greece, Jun 2011.

References

[1] M. Gao, W. Soh, and M. Tao, "A transmission scheme for continuous ARQ protocols over underwater acoustic channels," in *Communications, 2009. ICC'09. IEEE International Conference on*. IEEE, pp. 1–5.

[2] Wikipedia, "Scheduling (computing)." [Online]. Available: http://en.wikipedia.org/wiki/Scheduling_(computing)

[3] C. Smith, *3G wireless networks*. McGraw-Hill, Inc., 2006.

[4] S. Sesia, I. Toufik, and M. Baker, *LTE: the UMTS long term evolution*. Wiley Online Library, 2009.

[5] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A Berkeley view of cloud computing," *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, vol. 28, p. 13, 2009.

[6] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities," in *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*. Ieee, 2008, pp. 5–13.

[7] W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia Cloud Computing," *Signal Processing Magazine, IEEE*, vol. 28, no. 3, pp. 59–69, 2011.

[8] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications*, vol. 1, no. 1, pp. 7–18, 2010.

[9] "Amazon Elastic Compute Cloud." [Online]. Available: http://aws.amazon.com/ec2/

[10] "Google App Engine." [Online]. Available: https://developers.google.com/appengine/

[11] "Heroku, Cloud Application Platform." [Online]. Available: https://www.heroku.com/

[12] "Zencoder." [Online]. Available: https://zencoder.com/

[13] I. F. Akyildiz, D. Pompili, and T. Melodia, "State of the art in protocol research for underwater acoustic sensor networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 4, p. 11, 2007.

[14] V. Rodoplu and M. Park, "An energy-efficient MAC protocol for underwater wireless acoustic networks," in *OCEANS, 2005. Proceedings of MTS/IEEE*. IEEE, 2005, pp. 1198–1203.

[15] A. A. Syed, W. Ye, J. Heidemann, and B. Krishnamachari, "Understanding spatio-temporal uncertainty in medium access with ALOHA protocols," in *WuWNet '07*, Sep. 2007, pp. 41–48.

[16] B. A. Forouzan, *TCP/IP protocol suite*. McGraw-Hill, Inc., 2002.

[17] W. Yuan, K. Nahrstedt, S. Adve, D. Jones, and R. Kravets, "GRACE-1: Cross-layer adaptation for multimedia quality and battery energy," *IEEE Transactions on Mobile Computing*, vol. 5, no. 7, pp. 799–815, 2006.

[18] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14 –23, Oct 2009.

[19] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time CPU scheduling for mobile multimedia systems," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 149–163, 2003.

[20] W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia cloud computing: an emerging technology for providing multimedia services and applications," *IEEE Signal Processing Magazine*, pp. 59–69, May 2011.

[21] N. Chirdchoo, W.-S. Soh, and K. C. Chua, "Aloha-based MAC protocols with collision avoidance for underwater acoustic networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE.* IEEE, 2007, pp. 2271–2275.

[22] M. Molins and M. Stojanovic, "Slotted FAMA: a MAC protocol for underwater acoustic networks," in *OCEANS 2006-Asia Pacific.* IEEE, 2006, pp. 1–7.

[23] M. J. Neely, *Stochastic network optimization with application to communication and queueing systems.* Morgan & Claypool Publishers, 2010, vol. 3, no. 1.

[24] D. Pompili, T. Melodia, and I. Akyildiz, "A CDMA-based medium access control for underwater acoustic sensor networks," *Wireless Communications, IEEE Transactions on*, vol. 8, no. 4, pp. 1899–1909, 2009.

[25] C. L. Fullmer and J. Garcia-Luna-Aceves, *Solutions to hidden terminal problems in wireless networks.* ACM, 1997, vol. 27, no. 4.

[26] F. H. Fitzek and M. D. Katz, *Cooperation in wireless networks: principles and applications.* Springer, 2006.

[27] A. Mishra, M. Shin, and W. Arbaugh, "An empirical analysis of the ieee 802.11 mac layer handoff process," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 93–102, 2003.

[28] P. Karn, "MACA-a new channel access method for packet radio," in *ARRL/CRRL Amateur radio 9th computer networking conference*, vol. 140, 1990.

[29] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: a media access protocol for wireless LAN's," in *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4. ACM, 1994, pp. 212–225.

[30] C. Fullmer and J. Garcia-Luna-Aceves, "Floor acquisition multiple access (FAMA) for packet-radio networks," in *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 4. ACM, 1995, pp. 262–273.

[31] L. Kleinrock and F. Tobagi, "Packet switching in radio channels: Part I–carrier sense multiple-access modes and their throughput-delay characteristics," *Communications, IEEE Transactions on*, vol. 23, no. 12, pp. 1400–1416, 1975.

[32] I. . W. Group *et al.*, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," 1997.

[33] G. Xie and J. Gibson, "A network layer protocol for UANs to address propagation delay induced performance limitations," in *OCEANS, 2001. MTS/IEEE Conference and Exhibition*, vol. 4. IEEE, 2001, pp. 2087–2094.

[34] F. Salvá-Garau and M. Stojanovic, "Multi-cluster protocol for ad hoc mobile underwater acoustic networks," in *OCEANS 2003. Proceedings*, vol. 1. IEEE, 2003, pp. 91–98.

[35] X. Guo, M. Frater, and M. Ryan, "A propagation-delay-tolerant collision avoidance protocol for underwater acoustic sensor networks," in *OCEANS 2006-Asia Pacific*. IEEE, pp. 1–6.

[36] B. Peleato and M. Stojanovic, "A MAC protocol for ad-hoc underwater acoustic sensor networks," in *Proceedings of the 1st ACM international workshop on Underwater networks*. ACM, 2006, pp. 113–115.

[37] I. Kredo, B. Kurtis, and P. Mohapatra, "A hybrid medium access control protocol for underwater wireless networks," in *Proceedings of the second workshop on Underwater networks*. ACM, 2007, pp. 33–40.

[38] A. Syed, W. Ye, and J. Heidemann, "Comparison and evaluation of the T-Lohi MAC for underwater acoustic sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 26, no. 9, pp. 1731–1743, 2008.

[39] Z. Zhou, Z. Peng, J. Cui, and Z. Jiang, "Handling triple hidden terminal problems for multi-channel MAC in long-delay underwater sensor networks," *Mobile Computing, IEEE Transactions on*, no. 99, pp. 1–1, 2010.

[40] A. Carroll and G. Heiser, "An Analysis of Power Consumption in a Smartphone," in *USENIX annual technical conference*, 2010, pp. 271–285.

[41] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[42] J. Baliga, R. W. Ayre, K. Hinton, and R. Tucker, "Green cloud computing: Balancing energy in processing, storage, and transport," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 149–167, 2011.

[43] Z. He, W. Cheng, and X. Chen, "Energy minimization of portable video communication devices based on power-rate-distortion optimization," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 5, pp. 596–608, 2008.

[44] L. Su, Y. Lu, F. Wu, S. Li, and W. Gao, "Complexity-Constrained H.264 Video Encoding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 4, pp. 477 –490, april 2009.

[45] E. Lagerspetz and S. Tarkoma, "Mobile search and the cloud: The benefits of offloading," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*. IEEE, 2011, pp. 117–122.

[46] W. Zhang, Y. Wen, and H.-H. Chen, "Toward transcoding as a service: energy-efficient offloading policy for green mobile cloud," *Network, IEEE*, vol. 28, no. 6, pp. 67–73, 2014.

[47] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, 2009.

[48] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*. ACM, 2011, pp. 301–314.

[49] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[50] T. Burd and R. Brodersen, "Processor design for portable systems," *The Journal of VLSI Signal Processing*, vol. 13, no. 2, pp. 203–221, 1996.

[51] J. Pouwelse, K. Langendoen, and H. Sips, "Dynamic voltage scaling on a low-power microprocessor," in *Proceedings of the 7th annual international conference on Mobile computing and networking*, july 2001, pp. 251–259.

[52] F. Gruian, "Hard real-time scheduling for low-energy using stochastic data and DVS processors," in *Proceedings of the 2001 international symposium on Low power electronics and design*. ACM, 2001, pp. 46–51.

[53] D. Grunwald, C. B. Morrey III, P. Levis, M. Neufeld, and K. I. Farkas, "Policies for dynamic clock scheduling," in *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation-Volume 4*. USENIX Association, 2000, pp. 6–6.

[54] P. Pillai and K. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in *Proceedings of the eighteenth ACM symposium on Operating systems principles*. ACM, 2001, pp. 89–102.

[55] J. Lorch and A. Smith, "Improving dynamic voltage scaling algorithms with PACE," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 29, no. 1. ACM, 2001, pp. 50–61.

[56] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[57] A. Rodriguez, A. Gonzalez, and M. P. Malumbres, "Hierarchical parallelization of an h. 264/avc video encoder," in *Parallel Computing in Electrical Engineering, 2006. PAR ELEC 2006. International Symposium on.* IEEE, 2006, pp. 363–368.

[58] W.-T. Tsai, X. Sun, and J. Balasooriya, "Service-oriented cloud computing architecture," in *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on.* IEEE, 2010, pp. 684–689.

[59] F. Wang, J. Liu, and M. Chen, "CALMS: Cloud-assisted live media streaming for globalized demands with time/region diversities," in *INFOCOM, 2012 Proceedings IEEE.* IEEE, 2012, pp. 199–207.

[60] Z. Huang, C. Mei, L. Li, and T. Woo, "CloudStream: delivering high-quality streaming videos through a cloud-based SVC proxy," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM) mini-conference*, 2011.

[61] X. Wang, T. T. Kwon, Y. Choi, H. Wang, and J. Liu, "Cloud-assisted adaptive video streaming and social-aware video prefetching for mobile users," *Wireless Communications, IEEE*, vol. 20, no. 3, 2013.

[62] W. Zhang, Y. Wen, J. Cai, and D. Wu, "Towards transcoding as a service in multimedia cloud: Energy-efficient job dispatching algorithm," 2013.

[63] H. Sanson, L. Loyola, and D. Pereira, "Scalable distributed architecture for media transcoding," in *Algorithms and Architectures for Parallel Processing.* Springer, 2012, pp. 288–302.

[64] Z. Huang, C. Mei, L. E. Li, and T. Woo, "Cloudstream: delivering high-quality streaming videos through a cloud-based svc proxy," in *INFOCOM, 2011 Proceedings IEEE.* IEEE, 2011, pp. 201–205.

[65] A. Ashraf, F. Jokhio, T. Deneke, S. Lafond, I. Porres, and J. Lilius, "Stream-Based Admission Control and Scheduling for Video Transcoding in Cloud Computing," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, 2013, pp. 482–489.

[66] Y. Song, M. Zafer, and K.-W. Lee, "Optimal bidding in spot instance market," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 190–198.

[67] H. Ma, B. Seo, and R. Zimmermann, "Dynamic Scheduling on Video Transcoding for MPEG DASH in the Cloud Environment," in *MMSys, 2014 Proceedings ACM*.

[68] M. Gao and H. Jiang, "A JSW-based cooperative transmission scheme for underwater acoustic networks," in *Proceedings of the Seventh ACM International Conference on Underwater Networks and Systems*. ACM, 2012, p. 22.

[69] M. Isik and O. Akan, "A three dimensional localization algorithm for underwater acoustic sensor networks," *Wireless Communications, IEEE Transactions on*, vol. 8, no. 9, pp. 4457–4463, 2009.

[70] J. Lorch and A. Smith, "PACE: A new approach to dynamic voltage scaling," *Computers, IEEE Transactions on*, vol. 53, no. 7, pp. 856–869, 2004.

[71] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge Univ Pr, 2004.

[72] VideoLAN Organization, "x264." [Online]. Available: http://www.videolan.org/developers/x264.html

[73] "Oprofile 0.9.7." [Online]. Available: http://oprofile.sourceforge.net

[74] N. R. E. Grácias, "Mosaic–based visual navigation for autonomous underwater vehicles," Ph.D. dissertation, Universidade Técnica de Lisboa, 2002.

[75] N. R. Gracias, S. Van Der Zwaan, A. Bernardino, and J. Santos-Victor, "Mosaic-based navigation for autonomous underwater vehicles," *Oceanic Engineering, IEEE Journal of*, vol. 28, no. 4, pp. 609–624, 2003.

[76] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke, "Data collection, storage, and retrieval with an underwater sensor network," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*. ACM, 2005, pp. 154–165.

[77] T. O. Binford, "Survey of model-based image analysis systems," *The International Journal of Robotics Research*, vol. 1, no. 1, pp. 18–64, 1982.

[78] C. Weinhardt, D.-I.-W. A. Anandasivam, B. Blau, D.-I. N. Borissov, D.-M. T. Meinl, D.-I.-W. W. Michalk, and J. Stößer, "Cloud computing–a classification, business models, and research directions," *Business & Information Systems Engineering*, vol. 1, no. 5, pp. 391–399, 2009.

[79] A. Andrzejak, D. Kondo, and S. Yi, "Decision model for cloud computing under sla constraints," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on.* IEEE, 2010, pp. 257–266.