uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Yan Du

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M. (Computer Science)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

A Progressively Reliable Image Transport Protocol over Wireless Sensor Networks

TITRE DE LA THÈSE / TITLE OF THESIS

Azzedine Boukerche

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Shervin Shirmohammadi

Peter X. Liu

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

A Progressively Reliable Image Transport Protocol over Wireless Sensor Networks

by

Yan Du

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.Sc. degree in
Computer Science

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-50874-9
Our file Notre référence
ISBN: 978-0-494-50874-9

Abstract

Images are extensively used in Wireless Multimedia Sensor Networks (WMSN) for surveillance and object monitoring. The transmission of image data constitutes a significant portion of network bandwidth. Given the inherent differences of Wireless Sensor Networks (WSN) from wired networks, such as high probability of non-congestion related packet loss, very low bandwidth, Quality of Service (QoS) requirements, and limited processing power, the traditional transport protocol – the Transmission Control Protocol (TCP) – is not suitable for transferring images over WSN. In this thesis, a Progressively Reliable Image Transport Protocol – PITP, is proposed. First, with the support of progressive JPEG and out-of-order delivery of packets, this new protocol can display images smoothly with incremental quality. Second, this protocol guarantees receiver-controlled reliability and sound congestion control. In addition, a synchronization control mechanism is proposed within the protocol. A series of experiments are designed and simulated with ns2 to evaluate the performance of PITP. According to the results, PITP is proved to be suitable for image transmission over WSN.

Acknowledgements

I take great pleasure in thanking the many people who have helped me in my studies at the University of Ottawa. First, I would like to thank my supervisor, Professor Azzedine Boukerche, for his invaluable guidance and advice, as well as financial support. Second, I would like to thank Professor Shervin Shirmohammadi and Professor Peter Liu, for their careful reviewing on my thesis and serving on the exam committee, as well as Professor Dan Ionescu, for serving as the chair of the exam committee. I also would like to thank PhD. candidate Jing Feng, for his kindly help for my research work.

Finally, I would like to extend my thanks to my parents and my friends, whose love and encouragement have always sustained me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Wireless Sensor Networks (WSN) were initially motivated by military applications for battlefield surveillance [26, 49]. In recent years, the usage of WSN in civilian situations, such as disaster detection, health care systems, environment monitoring, weather forecast system, and habitat monitoring [2, 43], has been widely studied. A large-scale deployment of sensors can be used to collect physical and environmental information. Still images taken by digital cameras, which are one kind of sensors, are very useful for presenting remote on-site views in surveillance applications such as fire prevention systems and damage observation applications. Due to the inherent characteristics of WSN such as loss-prone, low-bandwidth and order-changing transmission of packets, the question of how to provide reliable, short delay image transmission over WSN becomes a demanding and interesting research topic. The motivation behind this thesis stems right from the demand of having a suitable image transport protocol for a surveillance system that is able to obtain progressively refined images from remote sensors.

1.1 Motivation

Images/video are extensively used for monitoring in surveillance applications. With the help of remotely located digital cameras, the host application can get on-site views from

far away places. Due to the fact that image files are usually large in size and the amount of images being transmitted is numerous, the transmission of images takes up a great portion of traffic over the networks. Traditionally, images are transmitted over reliable, high-bandwidth wired links. Therefore the duration between an image being requested and finally being displayed is quite short and almost negligible to the user. However, if images are transmitted over an unreliable channel, for example, the error-prone, order-changing and lossy wireless link, the duration will have to be prolonged and images may not be displayed smoothly to the end user.

Two aspects of exploration are required to solve this problem. First, transport layer protocols should be implemented to guarantee the reliability of the connection so that the errors and losses in the link can be hidden by retransmissions from the upper layer. Appropriate congestion control mechanism should be provided to avoid congestion in the traffic. Second, a proper image/video compression technique should be chosen to minimize the size of images without the sacrifice of image quality. Progressive displays of images should be supported to shorten the response time of images being displayed to the user.

As to transport layer protocols, the most popular transport protocols for wired networks are *Transmission Control Protocol* (TCP) [16, 30, 37] and *User Datagram Protocol* (UDP) [16, 29]. However, according to the inherent differences of WSN from wired networks, both of TCP and UDP are not suitable for transferring images. The reasons are described as follows respectively.

TCP is connection-oriented and has been optimized for data transmission over wired networks. TCP guarantees the reliability of transmission, in other words, TCP provides in-order, error-free and lossless packet transfer. TCP has a set of congestion control mechanisms such as slow start, congestion avoidance, fast retransmit and fast recovery to avoid congestion in the traffic so that higher throughput can be achieved [39]. TCP assumes that congestion is the only reason for packet loss. Therefore, a conservative decrease in window size and an increase in the *Retransmission Timeout* (RTO) can be

invoked to keep the data flow below the rate that can trigger the congestion collapse in the channel. This mechanism works well for wired networks, however, it is not suitable for wireless links. Due to the characteristics of wireless networks, the frequent bit errors and hand-off problems introduce a significant amount of packet losses other than congestion. Thus unnecessary congestion control actions are taken if applying TCP over wireless links. As a result, network performance is degraded. In addition, TCP requires strict ordered packet transfer. Complicated computation is required to maintain the order of packet sequence. Moreover, the in-order requirements prevent the image from being displayed progressively at the user end. Therefore, the TCP in-order packet transfer is inefficient for energy-conserved, low-bandwidth wireless networks.

UDP, different from TCP, is a connection-less transport protocol. It does not offer any mechanism to ensure the reliability of data transfer. It is possible that image data is lost, damaged or reordered during transmission. The benefit is that less computation is needed and transmission speed is faster. UDP is useful for transferring multimedia data, such as video and audio, whose users tend to be more sensitive to latency than small decreases in quality. However, for image transmission, image quality is an essential factor. Although the response time for displaying an image should be minimized, significant quality penalty should not be paid for it. Therefore, the unreliable UDP does not fit WSN for image transmission.

In terms of image/video compression technique, JPEG [38] is currently the most popular format for storing and transmitting image files, as it offers an excellent combination of image quality and compression ratio. A continuous sequence of JPEG still images can give a movie-like, motional appearance, which is termed as Motion JPEG or MJPEG [61]. In addition to MJPEG, the MPEG committee provides a variety of standards for video encoding, such as MPEG-1, MPEG-2, and MPEG-4 [63]. Particularly, a recent trend in using MPEG-4 for video compression and transmission is arising. The chief benefit of MPEG-4 is that it either renders better quality at the same compression ratio, or it results in smaller file size at the same quality [62]. However, MJPEG is preferred for

surveillance over WSN for the following reasons. First, MPEG-4 compression uses inter-frame prediction [52] to improve compression capability. More compression is achieved; meanwhile the computation and compression process is much more sophisticated. Thus, it is inefficient to run such complicated compression algorithms over sensors in WSN, since sensor nodes are always in the shortage of battery, memory and processing capacity. Second, without inter-frame prediction, MJPEG is easier to edit, thus further processing of still images, for instance, *Image-Based Modelling and Rendering* (IBMR) [14, 27], is possible. In sum, in comparison with MPEG-4, MJPEG provides a satisfactory combination of quality and compression ratio with lower overhead. Furthermore, JPEG has a progressive encoding mode, which compresses images through multiple scans smoothly with incremental quality. Progressive JPEG is useful when the network is low-speed and loss-prone. Therefore, with motional progressive JPEG images, users can obtain a quick preview before all data has been received.

According to above discussions, existing transport protocols for wired networks such as TCP and UDP do not suit for image transmission over energy-hungry, lossy WSN. Therefore, new protocols should be designed to provide reliable image transmission for WSN. In addition, compared with the MPEG standard, the MJPEG compression technique is preferred for sensoring and surveillance over WSN.

1.2 Thesis Objective

The main objective of this thesis is to design and implement an image transport protocol that can provide reliable, energy-efficient and progressive image transmission over WSN. Towards this objective, this thesis addresses the following relevant work:

- A comprehensive study of transport layer protocols over WSN is presented. The study is carried out in terms of the inherent differences of WSN from traditional wired networks, the performance metrics for transport protocols running over wireless links, and some existing wireless transport protocols.

- The JPEG still image compression standard is studied for image processing. The progressive JPEG encoding mode, which is useful for transferring images over low-bandwidth, and loss-prone wireless links, is reviewed. Several existing transport protocols for image transmission, such as TCP [16, 30], ITP [55], NCCI [53] and PCTP [4], are discussed.

- A new transport protocol for transferring JPEG images over WSN, which we have called "A Progressively Reliable Image Transport Protocol over Wireless Sensor Networks (PITP) ", is proposed. PITP is able to provide reliable, efficient image transmission with progressive output display at the receiver end. In addition, this protocol provides a synchronization control mechanism to manage the transmission progresses of different sensor nodes to ensure that images arriving at receivers are of the same quality level, so that received images can be further processed.

1.3 Contribution

The main contributions of this thesis are:

1. A new transport protocol specialized in image transmission, PITP, is proposed. Receiver-controlled facility is hired to guarantee reliability of the transmission. Progressive JPEG encoding for images is incorporated for higher user interactivity, and idea of TCP-ELN is employed for congestion control. The proposed protocol can be well-tuned to provide reliable, progressive image transmission over wireless networks.

2. A synchronization control mechanism within our PITP, is proposed for the first time. The synchronization algorithm can be used to schedule image transmission at the frame level and quality level. It maintains the correct temporal relationship among images from various sensors, guarantees that images are received at the same quality level, and achieves fair allocation of bandwidth.

1.4 Thesis Organization

The rest of the thesis is organized as follows.

- Chapter 2 presents a background study of challenges to utilize traditional wired-network transport protocols over WSN, the performance metrics for transport protocols over WSN and some existing popular wireless protocols at the transport layer.
- Chapter 3 reviews the JPEG still image compression standard, the encoding/decoding processes, and particularly the progressive JPEG encoding mode. The merits and demerits of some existing transport protocols specialized in image transmission are also discussed.
- Chapter 4 presents a detailed introduction of our transport protocol, PITP, for image transmission over WSN in terms of the motivation, the proposed environment for the protocol and the important characteristics of the protocol.
- Chapter 5 discusses some issues we are confronted with during the implementation phase of the protocol.
- Chapter 6 presents the performance evaluation of PITP, and the experimental results are presented.
- Chapter 7 concludes this thesis, followed by some suggestions for additional research in image transport protocols over WSN.

Chapter 2

Transport Protocols over Wireless Sensor Networks

Wireless Sensor Networks (WSN) is different from wired networks in terms of its high bit error rate, limited resources and etc. According to the characteristics of WSN, traditional transport protocols over wired networks are not suitable for transmissions over WSN. In this chapter, the general challenges of traditional transport protocols over WSN will be presented, followed by an introduction of the performance metrics for wireless transport protocols. Furthermore, some existing transport protocols for WSN will be introduced.

2.1 General Challenges of Traditional Transport Protocols over WSN

Initially motivated for surveillance of the battlefields in the military context, WSN has become an emerging area of research in recent years. A WSN is a wireless network composed of one or more sinks (base stations) and a large number of distributed sensor nodes [2, 50]. Typically, the sensor nodes that carry a limited source of battery, and equipped with radio transceivers or other communication devices, are in charge of gathering physi-

cal or environmental information of a target area or object. After that, the nodes process the information and report it to the sinks. The sinks then ask for more information from the sensor nodes in response [10]. Currently, the use of WSN in civilian applications such as smart homes, ambient intelligence health care systems, disaster recognition and prevention systems etc. are being extensively researched [?, 50].

However, compared with wired networks, WSN has some inherent characteristics so that traditional wired transport protocols do not perform well over WSN.

- *High bit error rate*: Wireless networks are more loss-prone than the wired ones. In wired networks, most packet losses are caused by congestion of the traffic, while this is not necessarily the case for WSN. In addition to congestion, high bit error rate and handoff problems are important reasons for packet losses in WSN as well [53]. For example, the bit error rate over a fiber-optic link is typically less than 10^{-12}, and a wireless link usually suffers from bit error rate of 10^{-6} or even worse [47]. TCP generally regards network congestion as the only reason for packet loss. Thus whenever loss of packet is detected, TCP invokes its congestion control mechanism to enter the slow start and congestion avoidance mode, which involves decreasing the transmission window and increasing the retransmission timeout (RTO) [5]. Although TCP works well at congestion control for wired links, as it responses to non-congestion related losses with unnecessary control actions, it will definitely degrade the throughput of wireless networks.

- *Limited resources*: As mentioned before, each sensor node in WSN is a small and light-weight device, which is self-contained with a tiny processing unit, memory, sensor, a communication device and battery. Due to its portability and mobility features, the resources for each sensor node are constrained. As the battery equipped is typically limited and nonchargeable, the processing power of the sensor is quite low. Therefore, energy consumption is one essential issue of designing transport protocols for WSN. In addition to limited source of power, the computa-

tion ability of sensor node is low, the memory and communication range is rather small [10]. To minimize the usage of resources and increase energy efficiency, the protocols cannot be as complex as for wired network and should be as simple and energy efficient as possible. Traditional TCP that requires a lot of computation and memory buffer to maintain the in-order delivery of packets does not perform well in this aspect [53].

- *Quality of Service (QoS) requirements*: QoS is perceived by people with diversified concepts from various aspects. In the context of networks, QoS typically refers to a certain combination of services that should be guaranteed by the network when data is transferred from one end to the other end [3]. QoS can be measured with a set of parameters such as jitter, latency, bandwidth, and delay etc. QoS is especially important for bandwidth-hungry multimedia applications [12]. However, due to its inherent characteristics, such as the relatively low bandwidth, power limited sensor node, and ad hoc topology etc., WSN is confronted with greater challenges to ensure the network performance. For example, traditional transport protocols without tuning may not satisfy the user of a latency-sensitive application, who is downloading an image from a loss-prone, low-bandwidth wireless network. Therefore, WSN requires special mechanism to provide satisfactory QoS.

In summary, due to the significant differences of WSN from wired network, such as high bit error rate, limited resources of sensor nodes, and special QoS requirements, traditional transport protocols do not perform as well as over wired networks. As a result, transport protocols tailored to WSN are in great demand.

2.2 Performance Metrics for Transport Protocols over WSN

In Section 2.1, the distinctive characteristics of WSN have been discussed, and it has been concluded that traditional transport protocols for wired network are not suitable for WSN. Thus to provide guaranteed end-to-end transmission over WSN, new protocols in the transport layer are required. In this section, the performance metrics to evaluate various wireless transport layer protocols will be introduced. Generally, the transport protocols for WSN should provide functionality in the aspects of reliable message delivery, energy efficiency and sound congestion control [32].

2.2.1 Reliable Message Delivery

As a WSN is a loss prone network where packet losses may result from either congestion or bit errors, transport protocols are required to guarantee reliable data transmission. There are several ways to classify reliability for WSN. Wang et al. categorize it as packet reliability and event reliability, where packet reliability refers to successful transmission of all packets and event reliability means successful detection of all events [10]. Jones et al. classify reliability into three types by direction of packets: point-to-point, point-to-multipoint and multipoint-to-point, which stands for communication between sink and a remote host, between sink and sensor nodes and between sink and multiple wireless sensors respectively [32]. Nevertheless, in our context, the focus will be put on whether the protocols are able to provide reliable packet delivery. The typical approaches to ensure successfully transmission of packet involves using notifications, acknowledgements and packet retransmissions etc. They will be discussed in details according to specific protocols in the next section.

2.2.2 Energy Efficiency

Sensor nodes can only carry limited amount of battery, which leads to the energy scarcity of WSN. Thus it is vital that transport protocols for WSN can work in an energy efficient approach. The high bit error rate in WSN introduces extra packet losses than in wired networks, so that additional retransmissions are required to guarantee reliable message delivery. Nevertheless, each retransmission implies the expense of energy. Therefore, the more retransmissions are invoked, the sooner the sensor node ends its life. As a result, to maximize the efficiency of energy utilization, attention should be paid to several aspects, including the number of retransmissions and the range of retransmission [10, 56].

2.2.3 Congestion control

There are generally two directions of traffic flow in WSN. One is the downstream traffic where data is multicasted from the sink to sensor nodes, and the other is the upstream traffic where various sensor nodes feed information to the sink [9]. Sensor nodes close to the sink always carry more collected data in the upstream traffic, which forms one major reason of network congestion [10]. Congestion brings more packet losses and longer latency, as when the data buffer overflows, new arrived data will have to be discarded. Moreover, delays and retransmissions of packet will degrade energy efficiency and shorten the lifetime of the sensor nodes. Therefore, appropriate congestion control mechanism at the transport layer is required for the protocols to ensure the QoS of the network.

2.2.4 Summary

In brief, the abilities to support reliable packet transmission, optimize energy utilization and provide sound congestion control are the major factors to evaluate the performance of transport protocols over WSN. Thus these metrics should deserve particular attention from the designers of new protocols without doubt.

2.3 Transport Protocols over WSN

As mentioned in Section 2.1, WSN differs from wired networks on several facts, including high probability of non-congestion related packet loss, energy scarcity, special QoS requirements, dynamic topology, and etc. Therefore, traditional transport protocols, the most famous ones of which are TCP and UDP, are not suitable for WSN any more.

A number of transport protocols have been proposed and implemented to guarantee reliable data delivery with appropriate congestion control mechanism in an energy efficient approach for WSN. Some of them will be discussed in detail as in the following.

2.3.1 Wireless TCP

According to the differences between wired and wireless network, TCP is not well suitable for WSN. A variety of researches have been carried on to improve the performance of TCP over WSN. They can be classified into two categories. The first approach is to hide the non-congestion related packet losses from the sender and retransmit the lost packets locally. In this way, the link is believed less error-prone and the detected losses are generally caused by congestion [5, 24, 25]. Some popular instances of this method covers the proxy-based SNOOP protocol [24, 25], which buffers the unacknowledged packets at the base station and do local retransmissions, and the split-connection protocols such as M-TCP [33] and I-TCP [1]. The second approach attempts to provide the sender with loss information such as selective acknowledgment [41] or explicit loss notification (ELN) [19] so that the sender is able to distinguish congestion-related packet losses from the others. According to Balakrishnan et al.'s research, protocols implemented in the first approach can improve the throughput by 10% - 30%, and by following the second approach, the performance of TCP can be enhanced by a factor or more compared with TCP Reno [24]. A comparison of some popular wireless TCP protocols is shown in Table 2.1.

Table 2.1: Comparison of wireless TCP protocols

Protocols	Reliability	Energy Efficient	Congestion Control
SNOOP [24, 25]	End-to-end	No	Yes
M-TCP [33]	Split connection	No	Yes
I-TCP [1]	Split connection	No	Yes
STCP [41]	End-to-end	Yes	Yes
TCP-ELN [19]	End-to-end	Yes	Yes

2.3.2 Pump Slowly Fetch Quickly (PSFQ)

Wan et al. have proposed a reliable transport protocol for WSN, the key idea of which is to pump slowly, fetch quickly [11]. To pump slowly means data is distributed slowly at the sender's side. Whenever loss is detected, a negative acknowledgment (NACK) is notified and sensor nodes that suffer loss fetch data segments quickly from their immediate neighbors. PSFQ is based on the assumption that data loss is mainly introduced by transmission error in the wireless link rather than congestion, because PSFQ focuses on transmission of binary images over WSN and the authors believe that the traffic is light. Therefore, this protocol emphasizes transmission reliability, whereas active congestion control is not taken into consideration. Wan et al. show that PSFQ outweighs existing related techniques in the aspect of responses to diverse error conditions. However, due to the lack of congestion control scheme, PSFQ is not suitable for multipoint-to-point scenarios where heavy traffic load is experienced [32].

2.3.3 Reliable Multi-Segment Transport (RMST)

RMST was first proposed by Stann et al. [18]. It is implemented by extending the directed diffusion [7], and can be regarded as the transport layer protocol running above the diffusion stack. RMST can be attached to a sensor node and configured without recompilation. RMST extends diffusion to support two transport layer services by providing

reliable message delivery and effective management and reassembly of data fragmentations. NACK is used in the protocol to notify losses of packets and request retransmission to recover. Both end-to-end recovery and hop-to-hop recovery at the transport layer can be achieved by configuring RMST at the sensor nodes end. Moreover, Automatic Repeat Query (APQ) at the MAC layer can also be configured to work with the transport layer protocol in combination. Similar to PSFQ, RMST focuses on reliable data transmission between sensor nodes and the sink, and little congestion control mechanism is implemented. However, the diffusion stack, on which RMST is realized, does offer minimum congestion control [7, 32].

2.3.4 Event to Sink Reliable Transport (ESRT)

One distinction of WSN from wired network lies in that WSN is an event-based system, whereas most transport protocols are implemented in an end-to-end scheme. ESRT, proposed by Sankarasubramaniam et al., achieves reliable data transfer over WSN from an event-to-sink perspective, rather than the traditional end-to-end notion [67]. A congestion control mechanism is used to achieve reliable detection of events and minimize energy expenditure. ESRT algorithms are running mainly at the sink and least computation is required at the energy-conserved sensor nodes . Different grades of reliability can be achieved by adjusting the reporting frequency of sensor nodes according to current congestion condition and reliability level of the network. This self-configuring feature of ESRT makes it suitable to work over WSN, the topology of which is much more dynamic than the wired network. The authors have conducted a series of experiments and simulations and come to the conclusion that ESRT can achieve both reliable and energy-efficient transmission.

Table 2.2: Comparison of transport layer protocols over WSN

Protocol	Reliability	Energy-efficient	Congestion control
PSFQ [11]	Hop-by-hop	Yes	No
RMST [18]	Hop-by-hop or end-to-end	Yes	No
ESRT [67]	Hop-by-hop	Yes	Yes

2.3.5 Summary

Plenty of efforts have been made to implement reliable and energy-efficient data transfer transport protocols for WSN. A variety of protocols are proved to be better suited to WSN than traditional transport protocols with different emphasis. Jones et al. compared several transport protocols for WSN with more details, including the four protocols mentioned above. A comprehensive evaluation was conducted according to the performance metrics, which covers reliability, efficiency and congestion control and the conclusion is shown in Table 2.2 [32].

Nevertheless, existing WSN-specific protocols have some limitations as well. First, most protocols focus on either reliability or congestion control; very few of them address both aspects (e.g., ESRT). Second, the ability to prioritize and schedule transmission among multiple sensors, which is desirable in distributed applications, is not provided by existing solutions. Finally, particularly for image transmission, semantics of image encoding standard, which can be employed to optimize protocol performance, is absent from current protocols. Therefore, designing new image transport protocol for WSN is highly demanded.

Chapter 3

JPEG Image Processing and Transmission

Currently, JPEG is one of the most popular image compression standards. By using JPEG standard, high compression ratio can be achieved without degrading image quality. In this chapter, the JPEG still image compression standard will be reviewed, with a detailed discussion of the compression and decompression processes. Further, the progressive encoding mode of JPEG images, which is useful for progressive and multiple image displays, will be reviewed. In addition, some existing transport protocols, which specialize in image transmission, will be discussed.

3.1 The JPEG Still Image Compression Standard

JPEG, stands for Joint Photographic Experts Group, which is the committee that creates the JPEG standard [38]. However, JPEG is much more often referred to as the standard and its implementation, rather than the committee. The JPEG Still Image Compression Standard was proposed in 1992 and was approved as ISO 10918-1 in the year 1994. JPEG represents both the compression standard and the file format used to contain the compressed data. The strength of JPEG lies in that it can produce high quality

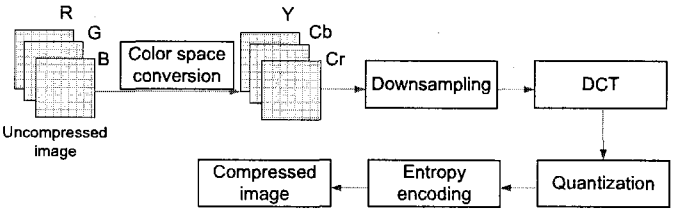

Figure 3.1: Common JPEG encoding process

image files, especially photographs, with high compression ratio. JPEG is one of the most commonly used approach of photograph compression, and the JPEG format is the most popular format for image storing and transferring over the Internet. The JPEG compression and decompression processes will be discussed in detail in the following subsections.

3.1.1 JPEG Encoding Process

There are many approaches to encode JPEG files, and the most common way is carried out with *JPEG File Interchange Format* (JFIF). The encoding process is made up of the following 5 steps [20].

1. *Color space conversion.* The original image is stored as RGB bitmap, where RGB represents the three color channels Red, Green and Blue. The first step of JPEG encoding is to convert the image from RGB to YCbCr. YCbCr is composed of three components: Y, Cb and Cr. The Y component represents the brightness – luminance, and the Cb and Cr components represent the color – chrominance of the pixel, where b and r stands for blue and red respectively.

2. *Downsampling.* According to the fact that the eyes of human being are more sensitive to brightness than color, downsampling of the two color components Cb and Cr is conducted. The result is that the resolution of the color components is reduced, usually by a factor of 2, and 33% or 50% space of the image is saved [60].

3. *Discrete Cosine Transform* (DCT). After downsampling, each component of the image is divided into blocks of 8×8 pixels. Then the two-dimensional DCT [35] is used to convert each component into the frequency domain, and the output is a two-dimensional array of 64 DCT coefficients. The coefficient in the top-left corner is called *Direct Current* (DC) coefficient representing the zero frequency; the remaining 63 coefficients are *Alternating Current* (AC) coefficient. By using DCT, most of the signal in the lower spatial frequencies is concentrated in the upper-left corner.

4. *Quantization.* The human eyes can distinguish more tiny differences in color or brightness over a relative area than the distinctness among the strength of high-frequency brightness variations. Due to this fact, plenty of information in the high-frequency components can be reduced by quantization, which is to divide each component by a constant and the round it to the nearest integer. The table that stores the corresponding constant to each component is called *Quantization Table.* As a result, most components that have high-frequency are rounded to zero, and the others are small positive or negative integers. In this way, the space required to store the image is significantly reduced.

5. *Entropy Encoding*, which consists two parts: *Run Length Encoding* (RLE) and *Huffman Coding.* After quantization, the quantized coefficients are re-arranged in a one-dimensional array by reading the original array in a zigzag way. Following this, RLE is applied by grouping the similar frequencies and inserting length code, finally Huffman coding is used on the result [20].

These are all the procedures involved to compress image data according to JPEG standard. The compression processes can be shown in Figure 3.1 [40]. High compression ratio is achieved gradually through each step without degrading the quality of image.

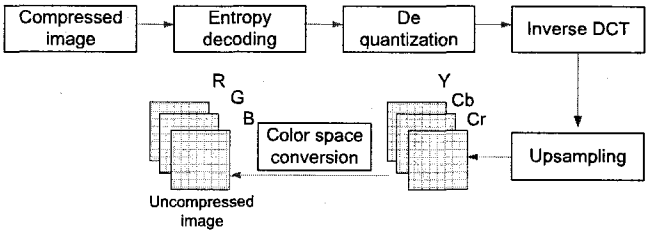

Figure 3.2: Common JPEG decoding process

3.1.2 JPEG Decoding Process

JPEG is a symmetric algorithm, and the JPEG decompression is processed by running the compression in a reverse route. The procedures will be introduced without digging into details, which have already been discussed in Section 3.1.1. To decode a JPEG image, entropy decoding is applied, which involves Huffman decoding and run-length decoding. Following this, the zigzag sequence of coefficients is re-ordered into a two-dimensional array. The entries in this array, which are quantized coefficients will be dequantized based on the same quantization table. After that, inverse DCT and chroma upsampling is applied orderly. Finally the image is converted from YCbCr color space to RGB color space and we will get a bitmap of the original JPEG format image. Figure 3.2 describes the decompress processes [40].

3.1.3 Progressive JPEG

The JPEG standard supports the following four types of encoding/decoding modes [20].

- *Sequential Encoding:* This is the mode most commonly referred to and the encoding and decoding processes we discussed earlier are based on this mode. Sequential encoding encodes images in the top-to-bottom, left-to-right route.

- *Progressive Encoding:* Rather than compress the whole image in a single scan as in sequential mode, progressive encoding encodes an image through multiple scans. The initial scan shows the image at relatively low quality, as a result little space

is required to store it. Following scans add information to the existing scans, and gradually refine the quality of the image. The final size of the image using progressive encoding is roughly the same as that applying sequential encoding. Figure 3.3 shows an image in four different scans with incremental quality.

- *Hierarchical Encoding:* In this mode, a set of compressed images of different resolutions are created. Low resolution images of smaller sizes are created first and the following higher quality images are produced based on the prediction from the former ones.

- *Lossless Encoding:* This encoding mode produces high quality image with guaranteed accuracy at some sacrifice of the compression ratio. Although JPEG is believed to work well at producing good quality images with high compression, it has to be realized that there is always a tradeoff between the quality of images and the compression rate. The more the images are compressed, the lower quality they render.

Among the four encoding modes, progressive encoding attracts our attention the most according to the following reasons. In some applications, the size of images may be large and it takes several minutes to transmit them, especially when network bandwidth is low. As progressive JPEG encodes the image by multiple scans, users of latency-sensitive and bandwidth-conserved applications can have a coarse preview of the image within a short time, and get the more precise views later. In addition, users can chose either to wait for a complete image or switch to the next image according to the rough image. Otherwise, users have to wait for a long time in order to get a full quality, and top-to-bottom display of image. The advantages of progressive encoding are not obvious given a high-speed network link, while if we are using the modem-speed link, the differences between using progressive and sequential encoding are quite explicit. Therefore, for our application which is running over low-bandwidth WSN, progressive JPEG is very attractive and useful.

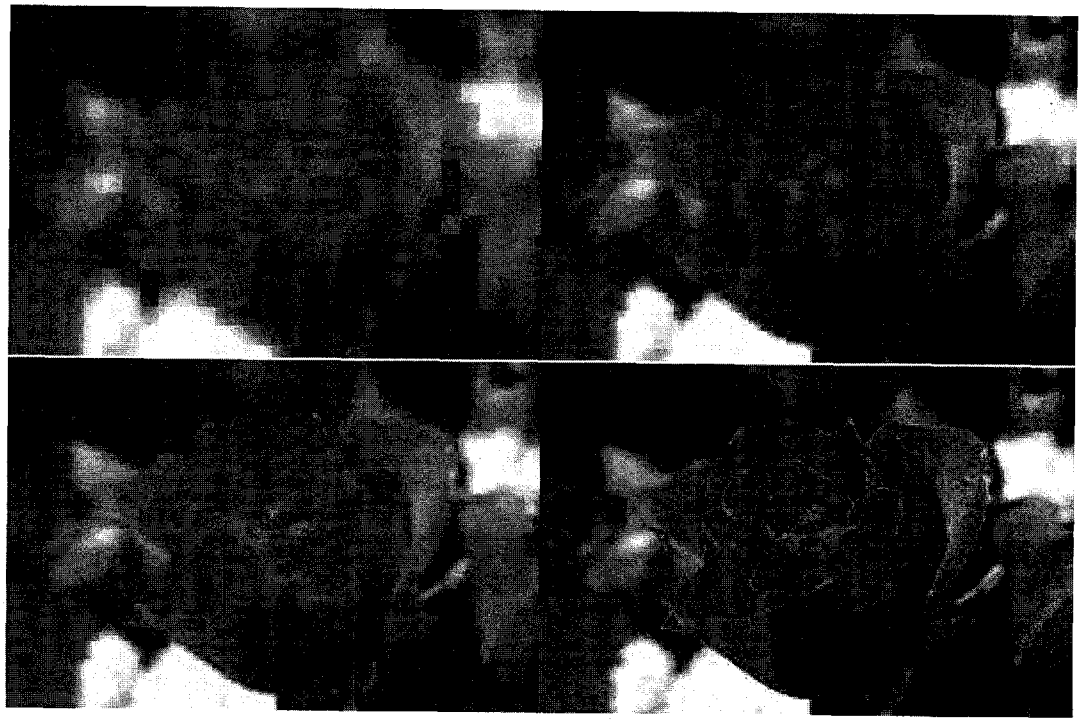

Figure 3.3: Progressive JPEG image in multiple scans

Technically, progressive encoding is exactly the same as sequential encoding in the first four steps of image compression, which includes the color space conversion, down-sampling, DCT and quantization. However, after quantization, a buffer is added to store the quantized coefficients. For the entropy encoding step, two algorithms, spectral selection and successive approximation, can be used to achieve the multiple scans of images with increasing quality [8, 21, 40].

- In spectral selection, instead of passing all DCT coefficients for one block of image in a zig-zag sequence in one encoding pass, the coefficients of a specific set of spectral bands for all blocks are passed to the encoder at one time. For example, in the first scan, DC coefficients for all blocks are passed to the encoder, so that the receiver can start decoding from the low-frequency bands and get a rough preview of the image. In following scans, AC coefficients for all blocks are passed, and the initial coarse image can be refined progressively. Figure 3.4 shows how a data unit

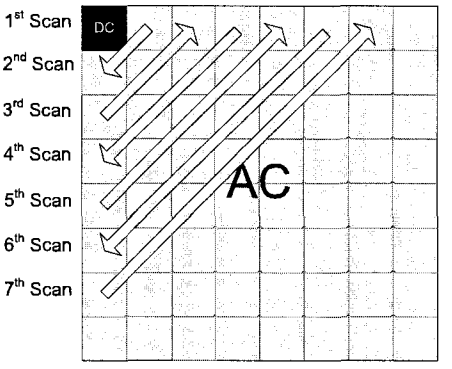

Figure 3.4: Encoding a data unit in multiple scans by spectral selection

is processed in multiple scans by spectral selection [8, 21].

- In successive approximation, instead of processing one coefficient at one time, certain bits of the coefficients are cut off at first and then processed in later scans. For example, the most significant bits of all coefficients are passed to the encoder initially, and the rest bits are passed later according to order of their significance. In addition, combinations of spectral selection and successive approximation are possible in JPEG.

3.1.4 The JPEG 2000 Standard

In the year of 2000, the Joint Photograph Experts Group, which proposed the JPEG standard in 1992, created a new standard for image compression – the wavelet-based JPEG 2000 Standard [38]. In addition to its superior compression performance, the most distinguished characteristic of JPEG 2000 is "compress once: decompress many ways" [17, 40]. This characteristic includes the following advantages of JPEG 2000: multi-resolution representation, region-of-interest (ROI) coding, lossy to lossless progression, and progressive transmission. On top of that, additional functionality such as error resilience, side channel spatial information, and image tiling, are provided. JPEG 2000 achieves significant improvement in image compression, however, it has not been widely supported by web browsers so far. Therefore, it still takes time for JPEG 2000 to be

popular used and JPEG is currently the most frequently used standard and format for storing images on the World Wide Web.

3.1.5 Summary

In this section, the general procedures to compress bitmap images into JPEG format and the reverse decompression procedures are introduced. After that, four types of JPEG encoding modes were presented and their characteristics were given respectively. By further exploiting the advantages of progressive encoding, the conclusion that it is most appropriate for our application, the network environment of which is the error-prone and bandwidth-limited WSN, has been drawn. The JPEG 2000 standard is introduced as well, however, due to its unpopularity over the Internet, it is not supported in our protocol at this moment.

3.2 Existing Image Transport Protocols

Currently there are many existing protocols used for image transmission. TCP is the most popular protocol over the World Wide Web for general data transmission, including image data. ITP [55] is a specialized protocol introduced by Raman et al. for transferring JPEG images. Iren et al. proposed a Network-conscious Compressed Images (NCCI) [53] protocol for transferring images over wireless networks. Cheng et al. introduced a protocol that code and transmit JPEG images in a priority-driven, progressive approach [4]. These protocols will be discussed with more details in the following.

3.2.1 TCP

The Hypertext Transfer Protocol (HTTP) [48] uses TCP to transmit images. Currently, this is the most commonly used protocol, and it works well generally as the network technologies are improving significantly. However, the inappropriateness of applying TCP

for WSN has been discussed in details in Chapter 2. Generally speaking, as TCP counts traffic congestion as the only reason for packet loss, unnecessary congestion control is invoked when link errors are the real reason that introduce the loss; therefore, the overall throughput is degraded. Besides, the strict in-order delivery of TCP requires a great amount of computation, which is not applicable for WSN as the sensor nodes are energy and resource conserved. As a result, TCP is not suitable for transferring data over high error-rate WSN, thus it is not applicable for image transmission over WSN, either.

3.2.2 ITP

An Image Transport Protocol for the Internet (ITP) is proposed by Ramen et al. in the year 2002 [55]. In the paper where ITP was introduced, the authors stated the disadvantages of applying TCP for image transmission over a loss-prone wired or wireless network, and introduced the ITP, which was proved to achieve better performance. ITP is implemented above UDP, with additional mechanism to guarantee reliable message delivery and congestion control in the traffic. ITP supports out-of-order delivery of data unit at application level, which decreases the waiting time at the receiver end. Receiver-controlled selective acknowledgment is used to request retransmission of lost packets. The *Congestion Manager* (CM) [23] is incorporated to provide end-to-end congestion control. Both JPEG and JPEG2000 formats are supported. In addition, error concealment algorithms are used to reconstruct images so that the part corresponding to the lost data can be predicted from existing data. In this way, better performance of interactivity can be achieved. The authors conduct a serials of experiments to compare the performance of ITP and TCP at different loss rates, and the results show that ITP, which renders better interactivity and responsiveness, surpasses TCP for image transmission over loss-prone networks.

3.2.3 NCCI

Iren et al. applied the concept of network-conscious [66] into image compression. *Network-conscious Compressed Images over Wireless Networks* (NCCI), are encoded to provide the optimized combination of response time and image quality to the end user [53]. NCCI has generally the following key characteristics. Firstly, it incorporates the *Application Level Framing* (ALF) [13]. Image is transmitted through *Application Data Units* (ADU) which carry semantics, so that each data unit received can be decoded independently from the other ADUs. In this way, out-of-order ADU can still be decompressed and part of the image will be displayed. Secondly, progressive display of the image is supported as each ADU received at the end user can be processed immediately and render an initial rough image, and the quality of the image can be improved with more data units received later on. The authors implemented the network-conscious GIF – GIFNCa [44], by applying network-conscious concept into GIF98 standard. The performance of GIF98 and GIFNCa was compared by running them over two reliable transport protocols, one of which requires in-order delivery and the other one does not. The results show that, when the network condition is good, which means packet loss and reordering rarely occur, GIF98 is better as the compression ratio is higher. However, when running over an unreliable network, GIFNCa excels as the progressive display of image decreases the idle time at the end user. Obviously, there is a tradeoff between compression ratio and progressive display for GIF images. For image transmission over wireless networks, which are not reliable, GIFNCa is believed more suitable than GIF98.

3.2.4 PCTP

A.M.K.Cheng et al. proposed a strategy of *Priority-driven Coding and Transmission of Progressive JPEG Images for Real-time Applications* (PCTP) [4]. This adaptive strategy can be used to provide high quality image/video compression and transmission in power-limited wireless applications. Images are segmented into multiple parts. Parts of more

Table 3.1: Comparison of image transport protocols

Protocols	Message Delivery	Congestion Control	Progressive Display	Receiver-controlled
TCP [16, 30]	In-order	Yes	No	No
ITP [55]	Out-of-order	Yes	Yes	Yes
NCCI [53]	Out-of-order	Not mentioned	Yes	No
PCTP [4]	In-order priority-driven	Not mentioned	Yes	No

significance are given higher quality, whereas less important parts are compressed at higher compression ratio to save time/energy. Thus a balanced combination of image quality and resource consumption can be achieved with acceptable sacrifice of quality level of the unimportant parts. In addition, a priority-driven scheduling approach is incorporated in PCTP, such that the important parts of images can be transmitted earlier than other parts. Progressive JPEG standard was used in PCTP for image encoding and transmission, and pattern recognition technique was utilized to segment images into sub-images of different significance. Experimental results showed that with this strategy, a combined compression ratio above 150 can be achieved and the size of images can be smaller than 20KB. In conclusion, the authors suggested that PCTP is suitable for image transmission in real-time, and time/bandwidth/power limited wireless applications.

3.2.5 Summary

In summary, some existing transport protocols for image transmission are presented, and their usefulness over WSN is discussed. Table 3.1 shows the comparison of existing image transport protocols, in terms of message delivery, congestion control, progressive display support and receiver-based control. In conclusion, TCP is not suitable for transferring images over unreliable networks. ITP and NCCI, with efforts put in transport layer control and progressive display of images respectively, far outweigh traditional TCP for image transmission over loss-prone, and low bandwidth networks. PCTP, which employs an adaptive strategy for priority-driven compression and scheduling, is also suitable for

image transmission in wireless applications. However, synchronization control should be an important property of transport protocols over WSN, where usually a variety of sensor nodes are involved. None of the protocols mentioned above has provided certain mechanism to synchronize multiple sensors. Therefore, we come up with our PITP – a progressive reliable image transport protocol over WSN, which will be discussed in detail in Chapter 4.

Chapter 4

PITP: A Progressively Reliable Image Transport Protocol over WSN

Existing image transport protocols such as TCP [30], ITP [55], NCCI [53], and PCTP [4] have been reviewed in Chapter 3. Based on merits and demerits of each protocol, we are motivated to design and implement PITP – a Progressively Reliable Image Transport Protocol, by combining the strengths of existing image transport protocols, with additional mechanisms for congestion control and synchronization control. In this chapter, a detailed discussion of PITP will be provided, in terms of the motivation behind the protocol, the proposed environment for applying the protocol, and the important characteristics of this protocol.

4.1 Motivation

In Chapter 3, the currently most popular image compression technique and four types of transport layer protocols responsible for image transmission were introduced, which are TCP, ITP, NCCI and PCTP. It has been concluded that TCP is not suitable for transmitting images over the WSN due to its congestion control mechanism and strict in-order packet delivery requirements. ITP, which supports out-of-order delivery and ensures the

receiver-controlled reliability, performs better than traditional TCP for transmission of images over loss-prone links. NCCI, which transmits images through *Application Data Units* (ADU) and provides multiple displays of images with progressively increasing precision, is also preferred as it shortens the response time at the user end without degrading the quality of image. PCTP divides images into sub-images of different importance, and applies priority-driven coding and scheduling according to the significance of each sub-image.

ITP and NCCI are both proved to exceed TCP's performance at transmitting images over the error-tolerant, energy-conserved network, and PCTP is believed to be well-tuned for image transmission in time/energy limited wireless applications. However, in our opinion, there is still space for improvement. A WSN usually contains a large-scale deployment of distributed sensor nodes and at least one sink node. For a distributed application, synchronization control is of great significance, since various sensor nodes should be synchronized to schedule the data transmission. It is possible that the bandwidth is not fairly allocated among multiple sensors. In addition, images received at the sink may be further processed for *Image-Based Modelling and Rendering* (IBMR) [14, 27]. For example, applying image mosaicing on resulting images to construct a panoramic view [51]. Some of the reconstruction algorithms require correct temporal relationship and quality consistency among images from different sensors. None of the existing image transport protocols reviewed in Chapter 3 has provided such a synchronization control mechanism. Therefore, we proposed a new protocol for image transmission over WSN: a Progressively Reliable Image Transport Protocol (PITP). PITP provides reliable and progressive transmission of images over wireless links. Moreover, PITP incorporates a synchronization mechanism to schedule the data transmission of various sensor nodes. The synchronization control of PITP ensures that higher quality images from certain sensor nodes are not transmitted until lower quality images from all sensor nodes have been received. A comparison table for important features of TCP, ITP, NCCI and PCTP has been provided in Section 3.2.5. Characteristics of PITP will be introduced respectively

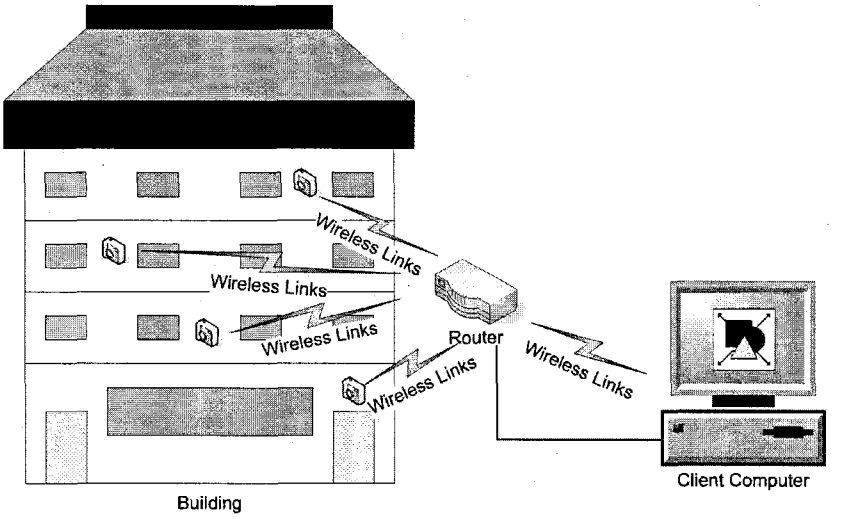

Figure 4.1: An example of the applied environment of PITP

with details later in Chapter 4.

4.2 The Applied Environment of PITP

With the development of electronic technologies in both software and hardware, surveillance technology is improving with leaps and bounds. Electronic surveillance is widely used in various types of applications. Originally designed for monitoring the battlefields in the military, nowadays surveillance is extensively researched about its use in civilian applications. Cameras and digital video cameras are popularly used for surveillance to ensure security. It is said that more than 4 million video cameras are employed in Britain to monitor the public areas such as streets, parks and buildings [57]. Images or media stream is taken by recording devices and transmitted through the network. The proposed PITP is responsible for providing reliable, synchronous image transmission with short latency at the transport layer.

An example of the applied environment of PITP will be introduced in a scenario-based approach. Figure 4.1 describes this example. There are 4 digital cameras equipped in a building to sense the presence of fire in the building, two located at the lobby, and

another two located at the emergency exit and the elevator exit respectively. All cameras are connected with the control room in this building through WLAN. The cameras are taking pictures at 10 *frames per second* (fps) and the pictures are transmitted to the control room. The building administrator, David, at the control room can view the images to obtain knowledge of the situations in the building. No special attention is paid until when a sign of fire occurs. For example, someday a fire takes place at the building. David is carefully watching his screen in the control room. The images from four cameras do not arrive exactly at the same time, some come faster and some come slower due to network conditions. The images are displayed progressively, with a rough view displayed initially and refined gradually with time. The transmission of the images are synchronized, which means that images from different cameras are transmitted and displayed at the same quality. Otherwise, if no synchronization control is provided, the image from cameras at the lobby may be shown in full quality while the image from the camera at the elevator exit is still being shown in its initial view. As a result, with the synchronization control, David can learn about the information from 4 cameras equally. According to the images displayed, David can understand the situation on the spot and ask the firefighters to take corresponding actions.

From the example above, it can be seen that the goal of PITP is to realize reliable, progressive, and synchronized transmission of images over wireless networks, so that the end user can view the images with low latency at intended image quality. However, according to the described scenario, the following assumptions should be accentuated in order to distinguish the proposed environment of PITP from typical WSN applications. (1) Sensor nodes in WSN are always connected to each other, or are able to communicate with each other for sharing information, time synchronization, routing and etc. Nevertheless, within the context of the applied environment of PITP, there are two types of sensor nodes of distinct responsibilities. The first kind of sensor nodes, which we call "the data nodes", are responsible for capturing and transmitting images. These sensor nodes are independent from each other, and no information is exchanged between them.

The other sort of sensor nodes – the routing nodes, are responsible for routing so that the transmitted data can reach the sink nodes. Image data is transmitted from data nodes to sink nodes through routing nodes. Different sink nodes can communicate with each other with traditional transport protocols. (2) Within the typical WSN, there is usually no static infrastructure of the networks and the topology of sensor nodes is dynamically changing all the time. In other words, a typical WSN is ad hoc and dynamic. However, the network environment for PITP is infrastructure-supported. Sensor nodes are statically configured and the topology could remain unchanged during a long period of time. Therefore, in summary, the network environment where PITP is going to be applied involves a large scale of distributed sensors and wireless links. Nevertheless, it is not exactly a typical WSN due to the differences in the communication capability and mobility of sensor nodes, and the stability of the network topology.

4.3 Important Features of PITP

4.3.1 Receiver-Based Negative Acknowledgement

The traditional TCP uses *Cumulative Acknowledgement* (CACK) to guarantee the reliability of transmission. Each packet sent is assigned a sequence number, and the receiver sends an acknowledgement (ACK) back to the sender to confirm the arrival of each packet. The sender is responsible for retransmitting lost packets when it detects packet loss. For example, if a packet of sequence number *SeqNo* is received, an ACK with the next expected byte is sent back to the sender. The TCP CACK mechanism works in the following way. For example, 4 bytes with sequence number *101*, *102*, *103* and *104* are sent from the sender (Assume the packet size is 1byte). When the receiver successfully receives all 4 packets, an ACK with sequence number *105* is sent and the sender will know that all packets with sequence number under *105* have been received. The acknowledgement schema works well in this case. However, in the case that the first packet, the *101* packet, is lost during transmission. The receiver will not acknowledge any packet

even if the following 3 packets all arrive. The sender will not learn about the loss either until a timeout occurs or 3 duplicate ACKs arrive. Then the lost packet is retransmitted and the sender can start transmitting new packets upon ACK for packet 106. CACK is good enough for wired networks, where the bandwidth is high and resources are sufficient. Nevertheless, for the low-bandwidth, energy-conserved and error-prone WSN, a more efficient mechanism should be utilized.

From the discussion in the preceding paragraph, we come to the conclusion that CACK is not suitable for transport protocols running over WSN for two reasons. First, according to CACK mechanism, received packets are acknowledged and the sender transmits the next packet upon arrival of ACKs. The bandwidth of WSN is inherently much lower than that of the wired link, therefore it does not make sense to have much of the traffic occupied for transmitting ACKs rather than the real data. Second, to ensure sender-controlled reliability, a great deal of computation is required at the sender. The senders in the context of PITP are sensors or digital cameras, which only carry a limited amount of energy and power. To minimize the energy consumption of the sender as much as possible, we are better off making efforts to decrease computation complexity and the number of transmissions. Therefore, PITP incorporates the receiver-controlled mechanism, to guarantee transmission reliability in an efficient way.

Unlike in TCP where receiver acknowledges the packets that have been received, in PITP the receiver sends ACKs for the packets that have not been received. *Negative Acknowledgements* (NACK) for lost packets are sent to the sender by the receiver. The receiver is now responsible for deciding when and which packets should be retransmitted and then requests the sender for this packet directly. The sender's work is only to retransmit the requested packets. Given the same example as for TCP above, 4 packets are sent and the first packet is lost. Within PITP, the receiver is able to determine that the packet with sequence number *101* is lost, as there is a gap between the sequence number of last packet received (for example: *100* initially) and the smallest sequence number (*102*) among the packets received in the nearest transmission. Then, the receiver

sends an ACK with sequence number *101* to the sender and the sender will retransmit that packet immediately. As a result, ACK transmissions are reduced; moreover, the computation of estimating which packet is lost has been moved from the sender to the receiver.

The receiver determines which packet to request in two cases. The first one is similar to what has been discussed in the example above. When the receiver detects that there is a gap between the sequence number of last received packet and the current one, it will consider that the packets with sequence number in between have been lost. Retransmission requests for those packets will be sent to the sender. In another case, the last packet, given our example, the packet of sequence number *104* is lost. As there is no subsequent packet received, the receiver will never find the gap and the sequence number of the lost packet cannot be known. In this case, a timeout needs to be counted for the packet loss. The receiver maintains a timer, when there is no activity during a timeout session, the receiver will assume that the tail loss occurs, and then requests for the packet whose sequence number is the biggest sequence number so far (*103*) plus 1. The value of the timeout is determined by the *Round Trip Time* (RTT). RTT is calculated dynamically by the sender in the same way as TCP, and is transmitted to the receiver as a field in the packet header.

In summary, by using the receiver-controlled NACK instead of the sender-controlled CACK, only lost packets are acknowledged and retransmitted. Furthermore, the computation for the decision of what packet to retransmit is transferred from the energy-scarce sender side to the receiver. Therefore, the receiver-controlled NACK is more suitable for transmission over WSN.

4.3.2 Out-of-Order Packet Delivery

Traditional TCP has strong restrictions on the order of received packets. The order requires that the packets are received with continuously ascending sequence numbers. For instance, if the last packet received has a sequence number of *100*, then the next

expected packet should have *101* as its sequence number (assume the size of each packet is 1byte). Otherwise, the later received packet is considered as out of order and will not be acknowledged. The example used in the previous section is taken to explain this mechanism in detail. 4 bytes of sequence number *101*, *102*, *103* and *104* are sent from the sender to the receiver. The first packet, *101* is received successfully. The packet of sequence number *102* is lost during the transmission and now packet *103* arrives at the receiver. Upon the arrival of packet *103*, the receiver compares its sequence number with the last acknowledged one (the packet of sequence number *101*), as $103 > 101 + 1$, packet *103* is regarded as out of order and not acknowledged. Similarly, the arrival of the following packet, packet *104,* is not confirmed by the receiver. As no packets are acknowledged except *101*, the sender will either receive duplicate ACKs for *102* or receive no reply during a timeout session, and both of them will result in long latency and low bandwidth utilization ratio.

The strict in-order mechanism is inefficient in the consumption of both time and power. The response time is increased. Therefore, PITP is required to support out-of-order packet delivery, in order that only lost packets need to be retransmitted and out-of-order packets can be accepted and processed by the receiver. Figure 4.2 shows the differences between in-order and out-of-order delivery in image transmission [53].

Especially for image transmission, as out-of-order packet delivery is allowed, each packet should carry information independent from each other so that it can be processed immediately upon arrival. Therefore, each packet is a distinctive, independent data unit that can be decoded by the receiver. As the progressive display of images is also supported by PITP, as soon as a packet is processed by the receiver, additional information is added to the image so that the quality of the image is refined. Consequently, the client is able to view the image in short response time instead of being idle for a long time until the image is displayed sharply.

As out-of-order delivery is allowed, it is not necessarily the case that the sequence numbers of packets received by the receiver are continuously ascending. Therefore, the

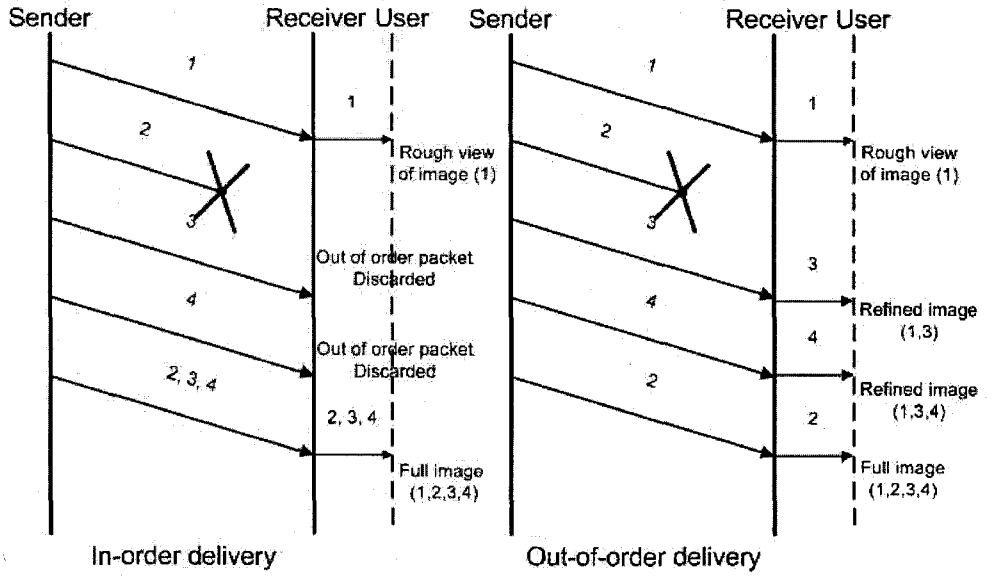

Figure 4.2: In-order delivery vs. Out-of-order delivery

receiver is unable to distinguish lost packets from those are out of order by comparing the sequence numbers. According to Paxon's analysis [65], for reordered packets in TCP, there is always a small delay before sending the ACK. In TCP, a threshold of 3 duplicate ACK packets is counted before the retransmission. PITP incorporates the approach proposed in ITP [55] to solve this problem. Before sending an ACK to request a packet, the receiver waits for a short time, which is suggested as $3/r$ in ITP, where r is the transmission rate (packets per second) of the sender. Packets that arrive within this time do not require retransmissions, whereas the retransmissions of those that do not arrive will be requested by the receiver.

In summary, out-of-order delivery of packets minimizes retransmissions of successfully received data and enables immediate processing of the packets that arrived. Therefore, it is beneficial to improvements in protocol efficiency and user interactivity. As a result, out-of-order packet delivery surpasses in-order delivery in image transmission over WSN.

4.3.3 Progressive Output Display

Most JPEG images on the Internet are compressed in the sequential encoding mode, where images are compressed in a single pass, and transmitted from top to bottom, stripe by stripe. Under these circumstances, images displayed at the end user are growing in height. Progressive JPEG format, in which an image is compressed through multiple scans with progressively increasing details, is another encoding mode of JPEG images and is not widely supported by web browsers. As to our context, where images are transmitted through low-bandwidth, loss-prone WSN, it is likely that an image cannot be displayed entirely at one time. For example, given the same transmission time, the sequential JPEG image is displayed with a couple of stripes whereas the progressive JPEG format image is displayed entirely at a lower quality. Given the time, the user can learn more information from a rough, but integrate view of the image than that from a few stripes with full quality. In other words, the user is more sensitive to the scale of the image than to the quality of the image at the beginning. Therefore, progressive JPEG leads to higher interactivity between the application and the user than sequential JPEG encoding does. The proposed PITP provides support for progressive JPEG encoding images, in order to display the output images smoothly with increasing precision.

The Independent JPEG Group have developed the *libjpeg* library, which is an open-source, widely-used implementation of JPEG encoders and decoders [22, 58]. Support for the progressive JPEG encoding mode is also provided by the *libjpeg* library. PITP makes use of *libjpeg* to read and write images of JPEG format. Progressive JPEG images are assumed to be available at the sender end. This assumption is possible as baseline/sequential JPEG format is widely used for storing images and it can be converted to progressive JPEG format image easily with *libjpeg* [58]. The images are read through multiple scans at the sender and the compressed image data is stored locally in a buffer. The total number of scans can be specified externally by the end user and passed to the sender in the header of the protocol. As soon as a pass of scan is completed, the data in the buffer is fragmented into packets and transmitted to the receiver. The image

data is stored and transmitted through *Minimum Coded Units* (MCU) [28], which is a tile of compressed image with a size of 8×8 pixels. Each MCU is independent from each other. Upon arrival of the MCU at the receiver, the receiver can decode the MCU immediately. The initial scans contain a little portion of data, thus images are of low quality. Following scans add more details to previous images progressively; therefore, images can be refined gradually.

To summarize, by taking advantages of the progressive JPEG encoding mode, the user is able to have a series of views with progressively increasing quality. In addition, even if not all the image data has been received, a rough preview of the image is still available. Therefore, in comparison with sequential JPEG encoding mode, progressive JPEG can lead to more interactivity between the user and the application.

4.3.4 Synchronization Control

Unlike traditional wired networks, a WSN usually contains a variety of distributed sensor nodes and at least one sink node. Synchronization control is always an indispensable feature for any distributed system. Many solutions relevant to time synchronization problems have been proposed, and the question of how to adapt time synchronization to WSN has been well-studied in a number of proposed solutions [15, 31, 42, 46]. However, they are not enough for PITP. PITP is designed in particular for image transmission, thus the ability to synchronize image transmission progress of various sensors is of great importance. Furthermore, synchronization is highly desirable to maintain correct temporal relationship among images, thus reconstruction of resulting images can be enabled. None of the existing protocols has provided algorithms to synchronize the transmission of images. Therefore, in addition to time synchronization, new synchronization control mechanism should be designed for image transmission scheduling in PITP.

The design goals of PITP synchronization control mechanism are composed of the following aspects: energy efficiency, control of image transmission, and accuracy. In terms of energy efficiency, the synchronization algorithm should be simple, and require

minimized computation of the energy-hungry sensor nodes. Therefore, in PITP, most of the synchronization control is processed by resource-sufficient sink nodes and sensor nodes are only involved by setting some flags in their packet headers.

The synchronization of image transmission is realized at two levels: the frame level and the quality level. Initially, local clocks of all sensor nodes can be synchronized with the clock of the sink node using existing time synchronization algorithms [31, 42]. Based on the assumption that all sensor nodes share the same clock with the sink node, the image transmission can be synchronized at the following two steps:

- *Frame Level Synchronization.* Each sensor node transmits images at the same frame rate. Whenever a new frame is captured, a timestamp can be used to record this time. In addition, each sensor node is labelled with a unique number. A pair of the timestamp and the sensor node number can identify a single frame transmitted from a sensor node. Therefore, it is easy to ensure that different sensor nodes transmit the same frame of images at a specific time with their value pairs.

- *Quality Level Synchronization.* As progressive JPEG is incorporated, each single frame of image is transmitted through multiple sub-images of increasing quality. The main goal of quality level synchronization is to guarantee that images from different sensors are of the same quality. This level of synchronization is very useful in the following two cases. (1) For image transmission over low-bandwidth wireless networks, it does not make sense to take up the traffic to transmit a high quality image from one sensor, when a coarse image from another sensor has not been received yet. (2) Further processing of received images such as *Image-Based Modelling and Rendering* (IBMR) is possible. Some of the rendering algorithms require that the images to process are of the same quality. Thus synchronization at quality level is necessary. To achieve this, different scans of an image can be distinguished with a number notifying the quality level. As each frame can be identified, each scan of image can be identified with the frame number and scan

number. Therefore, the quality level synchronization is realized.

Accuracy is an important aspect of synchronization control, however, trade-off between precision and efficiency exists. The more precision expected, the more complicated algorithm is required, and thus the less efficiency is achieved. Particularly, as to image transmission, human eyes are much more sensible to changes during seconds than that during milliseconds. Therefore, to guarantee accuracy as fine as millisecond is both unnecessary and inefficient. Experiments could be carried out to find out the appropriate level of accuracy.

In the current phase of our PITP implementation, the synchronization algorithm guarantees that images from multiple sensors are received at the same level of quality. This mechanism is specially for applications where image reconstruction algorithms are enabled. However, for other applications without further image processing, it is unnecessary to have such strict requirement on image quality. Future works will be conducted either to incorporate priority-oriented scheduling, or to design more flexible scheduling scheme, so that the control facility at the receiver end can be enhanced.

4.3.5 Congestion Control

The congestion control mechanism of TCP is one major reason for its poor performance over lossy wireless links. In TCP, congestion is considered as the only reason for packet losses by default. Thus, whenever a packet loss is detected, actions such as to decrease the window size and to prolong the retransmission timeout, are taken to avoid congestion. However, in wireless networks, a variety of packet losses are introduced by other reasons, for example, the bit errors in channels. Therefore, unnecessary congestion control actions are invoked if simply applying TCP's congestion control policies over WSN, and consequently, throughput is degraded significantly.

Many solutions have been proposed to improve TCP's performance over wireless links, such as the SNOOP protocol [24, 25], I-TCP [1], and TCP-ELN [19], which have

been reviewed in Section 2.3.1. The idea of TCP-ELN proposed by Buchholcz et al. is incorporated in PITP. ELN stands for *Explicit Loss Notification*. It can be used to distinguish different kinds of packet loss. Packet corruption resulting from link errors can be detected by the receiver at the MAC layer. The retransmission requests for such packets could include an ELN field in the header so that the sender can use it to differ the non-congestion related packet losses from the others. TCP congestion control mechanism is taken for congestion related packet losses. Otherwise, no congestion control is invoked and the transmission rate is not reduced.

Among numerous TCP variations, the ELN approach is chosen for the proposed protocol as it is easy to implement and does not require any modifications of the end-to-end scheme. In addition, it has been proved that TCP's performance can be improved significantly with ELN [19, 24]. Therefore, in PITP, ELN is used to distinguish packet losses of congestion and corruption, and congestion control actions are only taken for congestion related losses.

Chapter 5

Design and Implementation of PITP

The motivations and the applied environment of PITP have been introduced in Chapter 4. Furthermore, general solutions of important PITP characteristics have been proposed at the design level. In this chapter, the workflow of PITP is going to be introduced, and implementation issues related to PITP features are going to be discussed in detail.

5.1 Workflow of PITP

5.1.1 Workflow of the Sender

The workflow of the sender in PITP is graphed in Figure 5.1. In brief, it involves the following steps:

1. Connection establishment. The connection is initiated by the receiver, and details of the establishment procedures are discussed in Section 5.2.1.

2. When the connection is established, the sender starts sending packets with best efforts.

3. A new packet arriving at the sender end implies 3 possibilities. (1) The *RSN* field of this packet is set to *FIN*. The sender should terminate the connection as

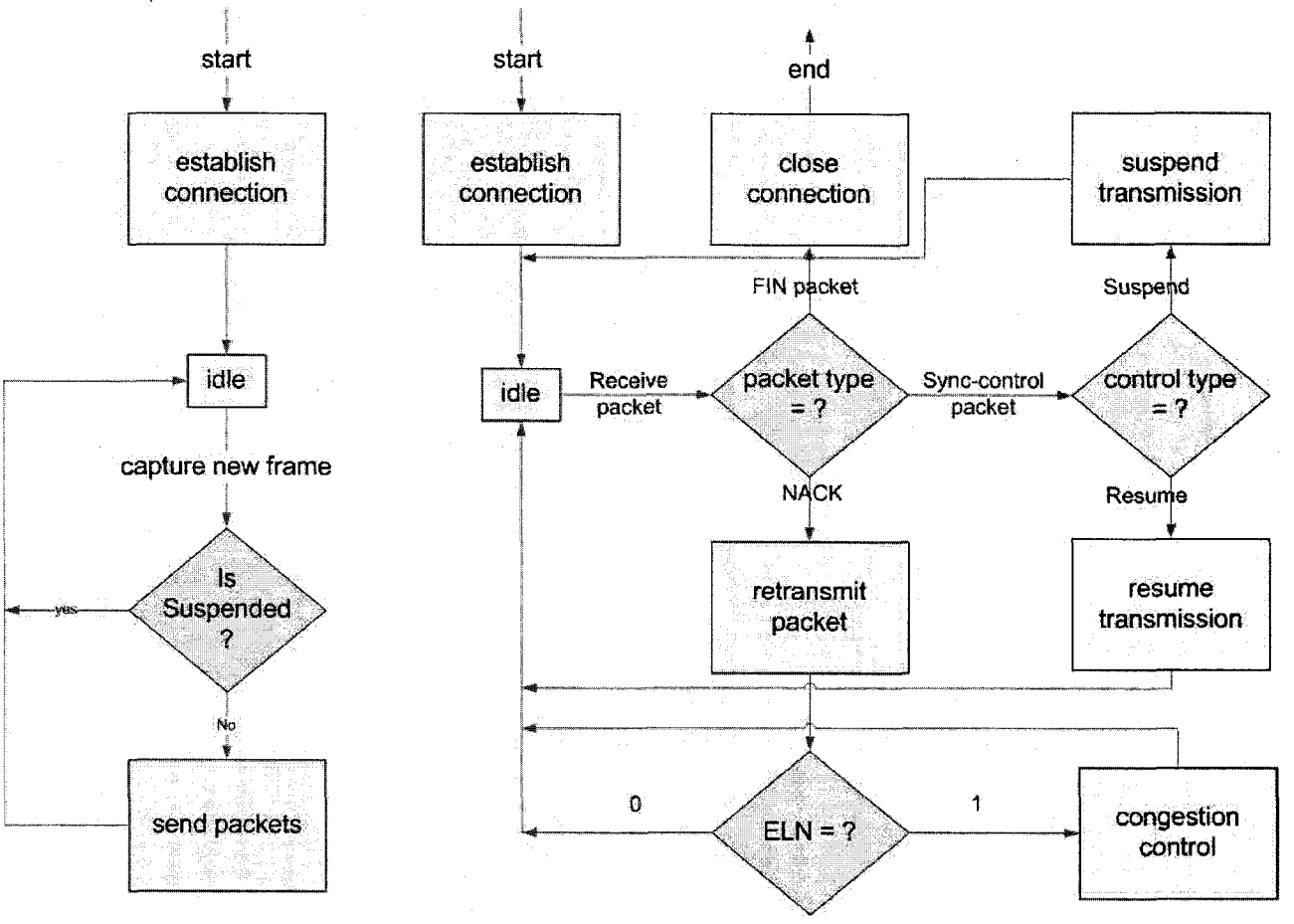

Figure 5.1: Workflow of PITP sender

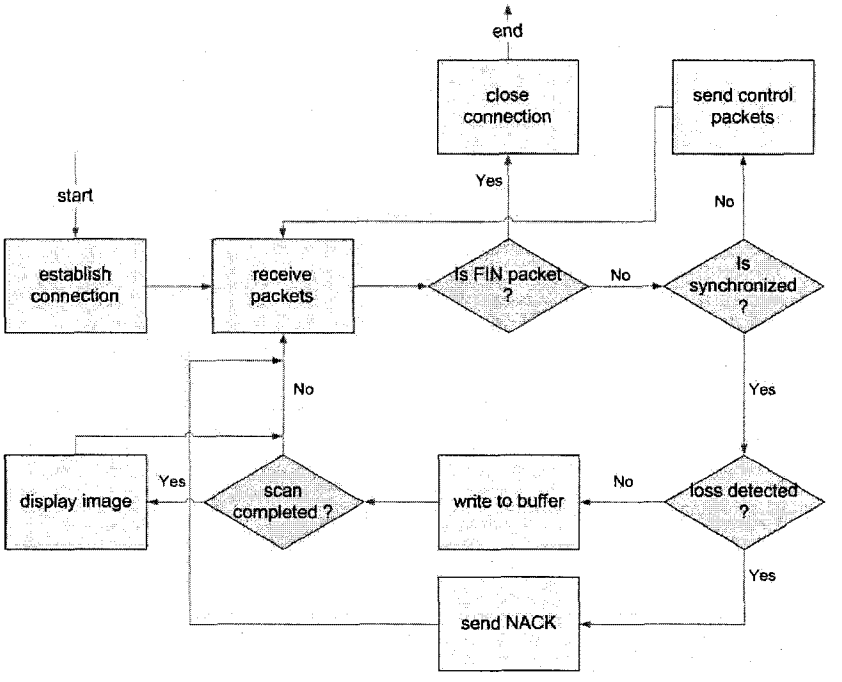

Figure 5.2: Workflow of PITP receiver

soon as possible. (2) The *RSN* field of this packet is set to *CTRL*. This packet is used for synchronization control, and the sender should either suspend current transmission or resume the transmission. (3) This packet is a simple NACK that requests retransmission of a packet. For this case, the sender should first retransmit the required packet. After that, the sender should check the *ELN* bit in the NACK header. If *ELN* is "1", which implies that the packet loss is caused by congestion, congestion control should be invoked. Otherwise, the sender continues sending packets as normal.

5.1.2 Workflow of the Receiver

Figure 5.2 describes the workflow of the receiver in PITP. Generally, it covers the following procedures:

1. Connection establishment. The receiver initiates the connection, and details of the

establishment procedures are provided in Section 5.2.1.

2. When a new packet arrives, the receiver checks the *FIN* bit of the packet header. If it is set, the receiver should terminate the connection directly. Otherwise, the packet is a data packet, and the following steps are going to be underwent.

   (a) The receiver checks if the transmission from the source of this packet is synchronized with other sensor nodes. This is done by comparing the *QualNo* fields in the packet header. If the transmission goes ahead of other nodes, the receiver should send a control packet to tell the sender to suspend transmission. This is presented in details in Section 5.2.4.

   (b) The receiver checks the sequence number of this packet. If loss of packets is detected, the receiver should send NACK to ask the sender to retransmit the lost packets. The *ELN* bit in the NACK header should be set if the packet loss is due to congestion. More details of this step are covered in Section 5.2.2.

   (c) Otherwise, the received packet is both synchronized and in-order. The data in the packet is decoded and written to a buffer. When all data for one scan of image has been received, the image could be displayed at certain quality.

5.2 PITP Implementation

5.2.1 Receiver-Based Negative Acknowledgement

According to the discussion in Chapter 4, PITP uses receiver-controlled negative acknowledgement for reliability of transmission. As a result, unnecessary transmissions of ACKs can be avoided, and the computation of lost packet determination can be moved from the energy-conserved sender side to the receiver. The receiver-based reliability is realized as follows.

- *Connection establishment.* Traditional TCP uses a three-way handshake to establish a connection. The three-steps establishment works in the following way: (1) The sender sends a *SYN* packet to the receiver to request an initial connection. (2) Upon the arrival of the packet, the receiver replies with an ACK of the *SYN* packet to tell the sender that the request is accepted. (3) Once the ACK is received, the sender acknowledges this ACK to the receiver. So far, both the sender and the receiver received an acknowledgement from their counterpart to demonstrate their availability, therefore the connection is established. PITP goes through the same procedures to initiate a connection as TCP does except that the receiver, not the sender, is responsible for the initial request. PITP involves two types of packets. The sender, usually a sensor node, transmits data packets with PITP headers. In response, the receiver sends ACK packets with ACK headers. The structures of the basic PITP header and ACK header are described in Figure 5.3 [55] and 5.4. Additional fields can be added to the basic headers in order to realize a specified feature, such as synchronization and progressive output display. The 28-bytes PITP header has 8 fields in total. The fields of *SeqNo*, *Offset* and *Length* together identify a data fragment. The *Timestamp* records the time of a packet being transmitted. The *AckNo* is useful for connection establishment and retransmission, and it tells which ACK the sender is responding to. The *LatestRtt* is the latest *Round-trip Time* (RTT) calculated by the sender encapsulated in the header for the receiver's interest. The ACK header has a length of 24 bytes, but the other fields are the same as in PITP header. However, there is an additional *RSN* (Reason) field to indicate the reason of the retransmission request. Some possible reasons include *SYN*, *SYN-ACK* and *FIN*, which are involved in the connection initiation and termination, and *TIMEOUT*, which is useful for the sender to decide the type of loss recovery. The following three steps are taken to establish a connection. Initially, the receiver sends an ACK packet to the sender, with the reason set to *SYN* and the *SeqNo* set to a random value x. Secondly, the sender answers with an PITP

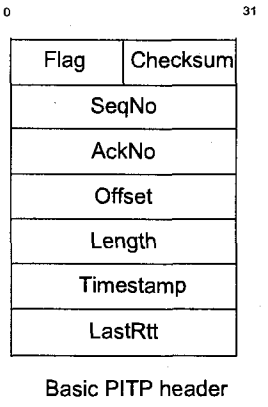

Figure 5.3: The Basic PITP header

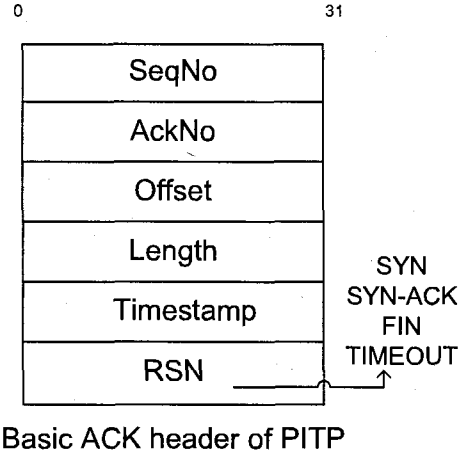

Figure 5.4: The Basic ACK header

packet, with the *AckNo* set to $x + 1$ and the *SeqNo* set to another random value y. Thirdly, in response, the receiver sends another ACK packet, with the *AckNo* set to $y + 1$ and *RSN* field set to *SYN-ACK*. Therefore the connection is established and the sender is able to send data packets to the receiver. Figure 5.5 shows the procedures to initiate a connection according to the discussions above.

- *Connection termination.* Usually, traditional TCP takes the following four steps to terminate a connection [30]: (1) The sender sends a packet with the *FIN* bit in the TCP header set to notify that all data has been sent out. As an example, the

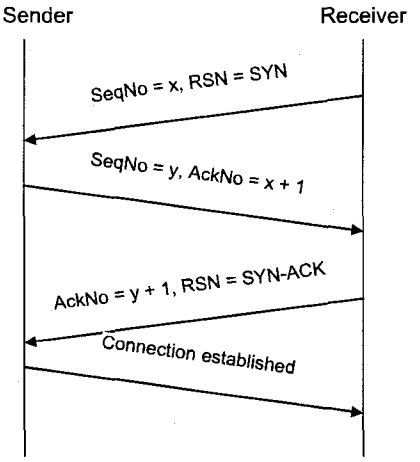

Figure 5.5: Steps to establish a connection in PITP

sequence number of this packet is x. (2) Upon arrival of this packet, the receiver learns that the sender wants to close the connection, and sends an ACK packet in response. The next expected sequence number is set to $x+1$. Meanwhile, the receiver informs the application that the connection has been closed. (3) As TCP is full-duplex, which means data flows exist in both direction, the sender has to wait until the receiver side finishes. When the receiver has no more data to send, it sends an ACK packet to the sender, with the *FIN* bit set. The *AckNo* is set to $x+1$ notifying the next expected packet, and the sequence number is set to a random value y. (4) The sender replies with a packet of *AckNo* set to $y+1$, which implies that the sender has got the receiver's request for connection termination. Finally, the connection is closed. To be brief, a connection cannot be terminated until both the sender and the receiver has got a pair of *FIN* and ACK packets. The connection termination of PITP is almost the same as in TCP. The only difference is that instead of setting the *FIN* bit in the ACK packet header, in PITP the receiver notifies the sender by setting the *RSN* field into *FIN*. As it is difficult for the receiver to know when all data has been sent out by the sender, it is easier if the sender initiates the termination.

- *Loss recovery.* Generally, there are two kinds of packet loss. The first kind is that packets in the middle of the transmission sequence are lost, and the second kind occurs when the last packet in the transmission sequence is lost. These two types of packet loss are detected in different approaches.

    1. The first kind of packet loss can be detected by observing a gap between the sequence numbers and/or between the offset of received packets. The receiver keeps a record of the largest sequence number and the offset of the packets that arrive. When a new packet is received, if its sequence number and/or the offset is continuous with the previously largest ones, it should be accepted and the record should be updated according to the new packet. In another case, if its sequence number and/or offset is smaller than the ones in the record, it might be an out-of-order packet and should be either accepted or discarded according to further determination without updating the record. In the third case, if its sequence number and/or offset is larger than, but not continuous with the ones in the record, then it is probably that the packets in between have been lost. Sequence numbers and offsets of potentially lost packets are added into a *pending_request_list* which keeps the packets that need requesting for retransmission. Figure 5.6 describes how this kind of packet loss is handled.

    2. The second kind of packet loss results from the loss of the last packet in the transmission sequence. In this case, as there are no following packets, the receiver cannot detect the loss by observing a gap between sequence numbers or offsets. A timer – *PitpRqstTimer* – is used to count a timeout session. If there are no new packets that arrive during a timeout session at the receiver, the receiver can assume that the next expected packet is lost during transmission. Therefore, the receiver can request the sender to retransmit that packet. In traditional TCP, the sender is responsible for calculating the *Retransmis-*

```
if (gap_detected)
{
  for (all packets in the gap)
    {
      // create a timer for a lost packet
      timer = new PitpDelayTimer(SeqNo, nodeNo);
      // set expiration of the timer
      setDelayTimer(timer, duration);
      // add the timer into pending_request_list
      pending_request_list.insert(timer);
    }
} // no gap is detected
  else
{
  // a timer for this packet is in pending_request_list
  if (pending_request_list.contains(timer))
    {
      // cancel this timer
      cancelDelayTimer(timer);
      // remove this timer from pending_request_list
      pending_request_list.remove(timer);
    }
}
```

Figure 5.6: Pseudocode for the first kind of loss recovery

sion Timeout (RTO). Karn and Patridge summarized the RTO estimation of the original TCP specification [30] as follows [45]. Firstly, estimates of RTT, SRTT, are calculated by the following formula.

$$SRTT_{i+1} = (\alpha \times SRTT_i) + (1 - \alpha) \times s_i \tag{5.1}$$

$SRTT_{i+1}$is the newly computed SRTT and $SRTT_i$is the current SRTT. s_i represents the current sample of RTT, which is obtained by counting the duration between the time a packet is sent and the time of it being acknowledged. α is a constant of value between 0 and 1. Based on $SRTT_{i+1}$, RTO can be computed according to the following formula.

$$RTO_i = \beta \times SRTT_i \tag{5.2}$$

In the formula, β is a constant greater than 1 so that the estimated RTO is the upper bound of the round-trip time. However, modifications should be made to tailor this mechanism to our PITP for two reasons. Firstly, PITP acknowledges lost packets instead of received packets. Therefore, the sender does not receive an ACK for each packet. Secondly, computation at the sender end should be minimized in order to save energy of sensor nodes. Our solution is that the sender periodically sends empty packets with *AckNo* set to 0 (a normal *AckNo* starts with 1) to the receiver. The receiver acknowledges these packets by setting the *RSN* field of the ACKs to *RTT*, which implies that ACK packets are used for estimating the RTT. Upon arrival of the ACKs, the sender can compute the RTT of this trip by a simple subtraction, and transmit it to the receiver by resetting the *LatestRtt* field of the next packet to send. With the latest RTT obtained from the sender, the receiver is able to calculate SRTT and RTO according to the formulas shown above.

To summarize, in this section, the details about how to implement receiver-controlled mechanism for reliability were introduced. Establishment and termination of connections, together with loss recovery policies, were discussed.

5.2.2 Out-of-Order Packet Delivery

PITP supports out-of-order packet delivery for two reasons. First, retransmissions of out-of-order packets are reduced so that bandwidth is saved and energy consumption of sensors is minimized. Second, immediate processing of received packets is possible in order to give a progressive refining display of images. As described in Chapter 4, there are two aspects of the implementation that should be paid attention to. One is the independence of transmission units, and the other one is the ability to distinguish lost packets from out-of-order packets. These issues will be discussed in detail as follows.

- *Independent data unit.* One major goal of supporting out-of-order packet delivery is to provide immediate processing of received packets. In order for immediate processes of packets, the data that each packet is carrying should be independent from each other, such that each packet can be decoded independently at the receiver. The solution is to transmit images as *Minimum Coded Unit* (MCU) [28], which is a tile of 8×8 pixels of image. In JPEG compression standard, compression/decompression algorithms are applied to each MCU rather than the whole image. This results from the difficulty of DCT transformation. By dividing the whole image into blocks, the calculation becomes easier, meanwhile, less information is kept in memory during encoding and decoding. Whenever an MCU is received, decompression algorithms such as entropy decoding, dequantization and inverse DCT, are applied and the corresponding block of image is displayed. Therefore, MCU is an independent data unit of images that could be encoded and decoded independently from each other. By transmitting images through MCU blocks, even out-of-order packets can be processed immediately upon arrival, so that transmission latency is reduced.

- *Distinguish lost packets from out-of-order packets.* Allowing out-of-order packets introduces the question of how to discriminate lost packets from out-of-order packets, as these two kinds of packets should be treated differently. Out-of-order packets

should be accepted and decoded immediately, whereas lost packets should be requested for retransmission. In traditional TCP, a threshold of 3 duplicate ACKs is counted. When the sender has received 3 ACKs for the same packet, it implies the loss of that packet. This does not work for PITP, since rather than the sender, the receiver is responsible for detecting lost packets. Moreover, negative ACK is used such that only lost packets are acknowledged. Therefore, loss of packets could not be detected by counting the number of ACK. However, the TCP solution reflects the fact that a small delay is always experienced before acknowledging out-of-order packets, which was observed by Paxon et al [65]. ITP proposed a solution to this problem [55] by taking advantages of this observation. PITP incorporates the ITP approach, which calculates a threshold of waiting time instead of counting a threshold of ACK amounts. The receiver is maintaining a *pending_request_list*, which keeps potential lost packets. An *PitpDelayTimer* is attached to each packet in this list to count a duration. The appropriate duration suggested by ITP is $3/r$, where r is the transmission rate of the sender nodes. If packets arrive within this duration, the corresponding request is removed from the *pending_request_list.* In another case, packets relevant to the requests have not been received when the duration expires, a negative ACK for these packets will be sent to request the sender for retransmissions. Therefore, lost packets and bad order packets could be distinguished with the help of *PitpDelayTimer.*

To sum up, implementation issues related with support for out-of-order packet delivery were discussed in this section. Particularly, solutions of how to transmit images with independent data unit, and how to differentiate between lost packets and out-of-order packets, were introduced.

5.2.3 Progressive Output Display

Progressive encoding of JPEG images is supported by PITP such that higher interactivity could be achieved when images are transmitted over low-bandwidth, loss-prone wireless networks. The major differences between progressive encoding and sequential encoding lies in that progressive encoding compresses images through multiple scans, while sequential encoding compresses images in a single scan. PITP uses the *libjpeg* library [58], which is developed by the Independent JPEG Group [22], to realize JPEG image related functionality.

The *libjpeg* library offers full implementation of JPEG encoders and decoders and provide methods such as *read_JPEG_file()* and *write_JPEG_file()* for processing image file as well. Nevertheless, modifications of the original library are needed to tailor it to our implementation. The original read/write methods provided by *libjpeg* directly read data from a JPEG format image file and decompress it into a buffer storing RGB information, or read input data from a RGB buffer and compress data into a *.jpg* file. These methods cannot be simply applied to our protocol, since compressed data, instead of uncompressed RGB data, should be transmitted. Therefore, a *compress_JPEG_mem()* (Compress JPEG image in memory) is implemented to compress RGB image to JPEG image and store the compressed data in memory. Correspondingly, a *decompress_JPEG_mem()* (Decompress JPEG image in memory) is also implemented. The function signature of *compress_JPEG_mem()* is as follows:

compress_JPEG_mem(JSAMPLE image, JSAMPLE* cimage, int quality, int * filelength, bool b_progress)*

where *image* is a pointer to the RGB image buffer, *cimage* points to the compressed image buffer, *quality* represents the level of compression quality (100 for full quality, smaller value for lower quality), *filelength* points to an integer recording the length of compressed image buffer, and *b_progress* is a flag notifying whether progressive encoding is used.

Default *read_JPEG_file()* method reads image files from top to bottom, with sequential encoding mode. An image file is read line by line, until the end of file is reached. When progressive JPEG is applied, a new loop is introduced, which controls the flow of multiple scans. The default number of scans is set to 10 for colored images and 6 for grey-scale images. The number of scans could also be specified externally. Within each scan, input data is retrieved line by line, as in sequential mode. When data for one specific scan has all been consumed, a new scan is invoked. The procedure continues until all input data has been read. The Pseudocode for the described procedures is shown in Figure 5.7. Little modification is required for the compression routine of progressive JPEG images except for invoking the *jpeg_simple_compression()* method to construct a scan list.

The amount of work required to display each scan of an image is the same as that to display a full quality image. Therefore, progressive display is useful under the conditions that the speed of an image being displayed by the decoder is faster than that of the image being transmitted during the network. PITP takes the assumption that sink nodes are configured in computers with spacious memory and fast processing speed. In addition, the networks where images are being transmitted are low-bandwidth wireless links, which are typical of packet loss, out-of-order packets and handoff problems. Thus, in terms of the transmission rate of images, the decoder is fairly fast at displaying those images. Therefore, it makes sense to support progressive display of images in PITP.

In summary, the *libjpeg* library, which is used to implement JPEG image related functionality in PITP, was introduced in this section. Furthermore, modifications made to the library, in order to suit PITP implementations, were presented as well. In addition, discussions of the usefulness of progressive display were carried out and the conclusion that progressive display is helpful for image transmission over WSN was drawn.

```
// input is not completely consumed
while (!jpeg_input_complete())
{
  // start a new scan
  jpeg_start_output(input_scan_number);

  // one scan is not finished
  while (output_scanline < output_height)
  {
    // read one line of input into a temp buffer
    jpeg_read_scanlines(buffer, 1);
    // put data in memory
    put_scanlines_someplace(buffer[0]);
  }
  // one scan is finished
  if (jpeg_finish_output())
  {
    // fragment data into packets
    fragment_jpeg_data();
    // start transmit image data
    send_data();
  }
}
```

Figure 5.7: Pseudocode for progressive encoding multiple scans

5.2.4 Synchronization Control

Motivations for synchronization control mechanism in PITP were introduced in Chapter 4. It has been pointed out that the synchronization should be energy efficient, with reasonable accuracy, and be able to synchronize image transmission. General design guidelines of these features have been proposed in Section 4.3.4, and additional implementations of PITP are going to be discussed as follows.

In order to support synchronization of image transmission, additional fields are added to the basic PITP header, which was introduced in Section5.1.1. Two integers, *NodeNo* (sensor node number) and *QualNo* (quality level), are used to specify the source of image data and the quality level of images respectively. The *NodeNo*, together with the timestamp field, which records the time of a frame being captured, can uniquely identify an image frame. Images from different senders can be considered as the same frame if the timestamps are the same. However, as images are transferred through networks, it is both very difficult and inefficient to achieve that images from various senders can be received at exactly the same time in milliseconds. It is reasonable to allow a range of differences in the timestamp. Most films in theatre are shown at 24fps, as less smoothness is acceptable in PITP, lower frame rate can be used [59]. For example, images are transmitted at 10fps, which means every 100 milliseconds a new frame is transmitted. It can be assumed that timestamps of differences smaller than 50 milliseconds are considered as the same. Experiments could be conducted to find the lowest acceptable frame rate and the appropriate range for timestamp differences.

In addition to the frame level synchronization described above, quality level synchronization is also supported in PITP. The quality level synchronization ensures that a higher quality image from a single sender is not transmitted until either lower quality images from all senders have been received or the receiver cancels the transmission of lower quality images. To be brief, it is designed to guarantee that images from various senders are received at the same quality level during a duration. To achieve this goal, the *QualNo* is attached to each data packet. The receiver checks the *QualNo* whenever

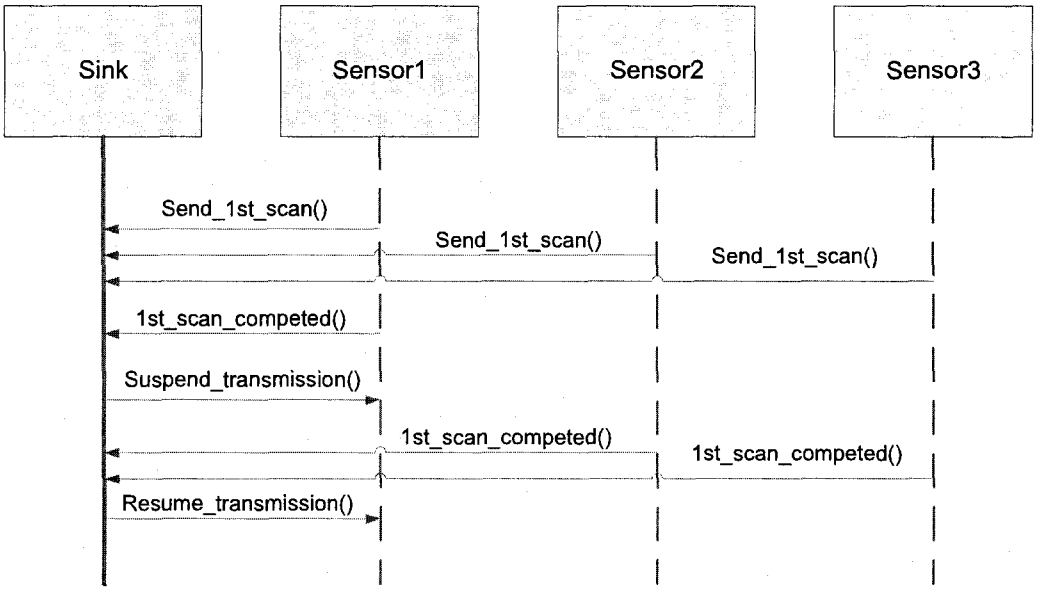

Figure 5.8: Sequence diagram for a synchronization example

a new packet arrives. If the value of packets from one sensor node is greater than that from the others, packets with greater *QualNo* should be discarded. Meanwhile, control packets will be sent by the receiver to tell the sender to suspend the transmission. As soon as lower quality images from all senders are received, a new notification will be sent to the sender to resume its transmission. The described procedure is shown in Figure 5.8. In the other case, if the user is not interested in a set of images according to the lower quality previews, he could ask for the next frame of images directly. Control packets will be sent to all sensor nodes to cancel the transmission of current images and require for following images.

According to our discussions above, two levels of synchronization could be achieved by adding new control fields into the basic packet header of PITP. Most modifications are made at the receiver end to control the transmission rate, and little changes are required at the sender end. Therefore, the synchronization mechanism will not significantly increase the energy consumption of sensor nodes. In terms of accuracy, a range of differences is allowed when comparing timestamps for efficiency concerns.

5.2.5 Congestion Control

PITP incorporates the TCP-ELN mechanism, which was originally proposed by Buchholcz et al. to improve the performance of traditional TCP over error-prone wireless links [19]. The idea of TCP-ELN is that by providing the sender with explicit information about lost packets, two different causes of packet loss – congestion and bit errors – can be differentiated. Corresponding actions are taken for different kinds of packet loss. If packets are lost due to congestion in the traffic, TCP-like congestion control mechanism is invoked. Otherwise, if packet losses result from radio channel errors in the transmission channels, packets should be retransmitted directly without any operation for congestion avoidance.

In order to apply the idea of TCP-ELN to our implementation, modifications are required at both the sender and the receiver end. Firstly, the receiver must be able to separate the two kinds of packet loss: loss due to congestion and loss due to channel errors. Furthermore, the cause of packet loss should be notified by the receiver. Therefore, at the receiver end, a new flag – the ELN bit– is added to the ACK packet header, so that the reason for a packet loss can be explicitly notified. It is assumed that loss information can be obtained from lower layers, for example, the MAC layer. Buchholcz et al. has conducted a set of experiments to show how loss information could be gained to distinguish those two types of packet loss [19]. If a packet is lost due to network congestion, the ELN bit of the ACK packet header for that lost packet is set to "1". Otherwise, for a packet lost resulting from link errors, the ELN bit is set to "0". In this way, the cause of a packet loss can be encapsulated in the ACK header for that packet, and transmitted to the sender.

Simple modifications are made at the sender so that two kinds of packet loss can be treated differently. Upon arrival of an ACK packet, the sender retrieves the packet header and checks the ELN bit. If the ELN bit is set to "1", which implies that the packet loss is due to congestion, then the corresponding packet is resent, and the TCP congestion control actions are taken as well to reduce sending rate. However, if the ELN

bit is set to "0", which means that the packet is lost due to bit errors in the wireless channel, then the sender directly retransmits the corresponding packet to the receiver, and no congestion avoidance technique is used.

In summary, the TCP-ELN technique is used in PITP to tailor the congestion control mechanism of traditional TCP to lossy wireless networks. Modifications made at the sender and the receiver end to support the TCP-ELN idea were introduced. With the modified congestion control mechanism, performance of the protocol can be improved.

Chapter 6

Simulation and Results

The PITP is implemented and simulated with ns2 [64]. In order to evaluate the performance of PITP, experiments have been conducted to compare it with TCP and SITP. By transmitting the same images using different protocols and comparing the percentage of image displayed at various loss rates, we have proved that PITP can lead to faster transmission of images than TCP and smoother progressive display of images than SITP. Another set of experiments has been carried out to test the synchronization mechanism of PITP, and it is proven that PITP can provide synchronized image transmission among multiple senders.

6.1 Simulation

6.1.1 Simulation Introduction

We simulated our PITP with the ns2 network simulator. In ns2, senders and receivers of transport layer protocols are designed as agents, which capture the essence of the protocols such as congestion control, retransmission management, but have no concern with the dynamic window advertisement, connection establishment and termination. Our PITP is implemented as two one-way agents: the *PITPAgent* and the *PITPSink* agent,

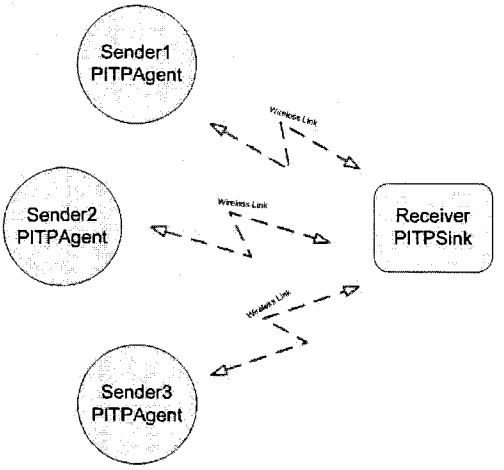

Figure 6.1: PITP simulation topology

which function as the sender and the receiver respectively. As the point-to-multipoint communication should be supported, each *PITPSink* is able to communicate with multiple *PITPAgent*, but *PITPAgent* can not share information with each other. Ns2 has provided the implementation for a basic TCP – the Tahoe TCP, which performs slow start and fast retransmission for congestion control [34]. We program our PITP by using the similar code for round-trip time estimation and retransmission timeout as for the basic TCP in ns2, with additional features such as receiver-based reliability, out-of-order delivery, progressive display, ELN for congestion control, all of which have been introduced in detail in Chapter 4 and Chapter 5.

Due to the lack of hardware equipments such as sensors and cameras, experiments could not be carried out in the real testing environment. In order to simulate the capture of images, we have prepared a variety of images in the hard disk. A set of images can be read into memory and then be transmitted at designated time to imitate images being sent by sensors at a specific frame rate. To enable progressive display at the receiver end, the prepared images at the sender end are converted into progressive JPEG format files in advance.

Time did not permit the implementation of the application for automatically dis-

Table 6.1: Simulation parameters

Parameter	Value
Bandwidth	1Mbps
MAC Layer Protocol	IEEE802.11b
Queuing Policy	DropTail
Interface Queue Length	1000 packets

playing received images on the client screen. Therefore, experiments were conducted by comparing the images outputted into the hard disk. However, it is a possible direction for future work to design and implement the software which can integrate the PITP protocol with the sensors and client computers.

6.1.2 Simulation Topology

The simulation topology of PITP is described in Figure 6.1. Three nodes are set to *PITPAgent* to send image data, and one node is set to PITPSink to receive data from the other nodes. Each node attached with the *PITPAgent* is connected with the node attached with the *PITPSink* through wireless links.

The parameters of this simulation are set as in Table 6.1.

The tcl code to configure these parameters in ns2 is shown in the following:

```
set val(chan)           Channel/WirelessChannel     # channel type
set val(prop)           Propagation/TwoRayGround    # radio propagation model
set val(netif)          Phy/WirelessPhy             # network interface type
set val(mac)            Mac/802_11                  # mac layer protocol
set val(ifq)            Queue/DropTail/PriQueue     # interface queue type
set val(ll)             LL                          # link layer type
set val(ant)            Antenna/OmniAntenna         # antenna model
```

```
set val(ifqlen)         1000                    # maximum packets in
                                                  interface queue
set val(adhodRouting) DSR                       # ad hoc routing protocol
set val(nn)             4                       # number of mobile nodes
```

6.2 Performance Evaluation

6.2.1 Comparison of PITP and TCP at Various Loss Rates

We ran a group of experiments comparing the performance of PITP and TCP. The TCP-Westwood solution (TCPW), which has been proven to significantly improve TCP throughput over lossy wireless links, was compared as well [6, 36, 54].

The experiments cover the transmission of images over (1) a reliable, ordered link; (2) the ordered, unreliable link with a loss rate of 10%, 20% and 30% respectively; (3) an unordered, unreliable link with a loss rate of 10%. Each experiment transmits a 30.1KB, progressive JPEG encoding image over a 802.11b wireless link.

The average percentage of image being displayed at different time for 0% packet loss rate is shown in Figure 6.2. As the figure describes, the same percentage of image can be displayed a little earlier by PITP than by TCP, which implies that the performance of PITP is slightly better than TCP. Our explanation for this difference is as follows. PITP only acknowledges lost packets, thus for the case where no packets are lost, PITP does not need to send ACKs. In contrast, TCP protocols acknowledge received packets, and the sender needs the ACK from the receiver to send the next packet. As a result, PITP is still faster than TCP-like protocols even the image is transmitted over a reliable and ordered link.

Figure 6.3, 6.4 and 6.5 graph the average percentage of displayed image at various time for packet loss at rates of 10%, 20% and 30% respectively. Significant advantage of PITP can be observed as the loss rate increases. The time cost to transmit the same image with TCP at a loss rate of 30% is almost 10 times of the time at 10% loss rate.

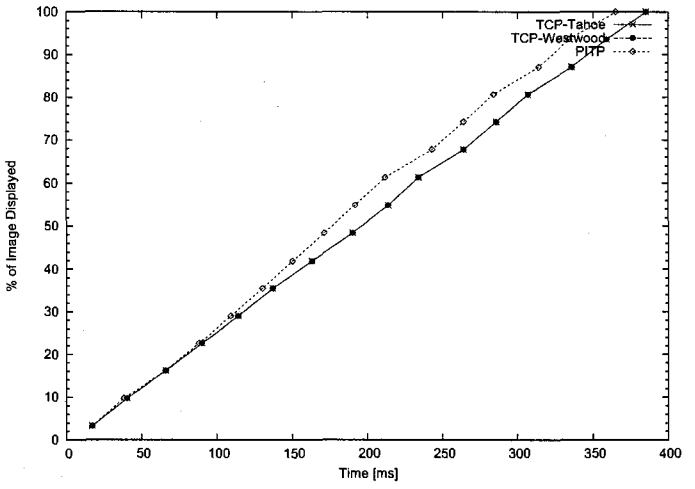

Figure 6.2: Comparison of TCP-like protocols and PITP at 0% loss rate

However, with PITP, the time required at 30% loss rate is about twice of that at 10% loss rate. By comparing the figures, it can be seen that as the loss rate rises, the time used for image transmission with TCP exponentially grows, while with PITP, linear increment of time is required. Generally, two aspects of protocols: loss recovery and congestion control, account for this great difference of PITP and TCP performance. (1) For loss recovery, PITP directly retransmits lost packets upon the arrival of NACKs. The transmission of packets following the lost ones is not affected. In TCP, lost packets are retransmitted either 3 duplicate ACKs have been received or the retransmission timer expires. NACK are only generated for lost packets in PITP, whereas with TCP all received packets are ACKed. Thus, the bandwidth utilization ratio with PITP is higher. (2) For congestion control, PITP does not decrease the transmission rate for non-congestion related packet losses, but TCP does not distinguish packet loss caused by channel errors from those caused by congestion. Once a packet is lost, TCP will take congestion control actions such as to decrease the congestion window size and to backoff the retransmission timeout. As the packet losses in the experiment do not result

from congestion, unnecessary congestion control of TCP leads to its poor performance compared with PITP. TCPW performs slightly better than TCP when the loss rate is low. As the loss rate grows, TCPW performs much better than TCP as it adjusts the congestion window according to dynamic bandwidth estimate.

The average percentage of image displayed at a variety of time for an unordered, lossy link is described in Figure 6.6. The loss rate is 10%, and 10% of packets are out of order. The performance of TCP is worse than that for an ordered link of the same loss rate, but much better than that for an ordered link at 20% loss rate. The reason is that in TCP-like protocols, out-of-order packets may generate duplicate ACKs to require retransmissions. Thus the overall transmission time over unordered link is longer than the time over ordered link. Nevertheless, packets are retransmitted immediately as the threshold of duplicate ACKs has been observed without waiting for the retransmission timer to expire. Therefore, it costs less time to transmit the same data over an unordered link at 10% loss rate than that over an ordered, 20% loss link. The performance of PITP is also between that for an ordered, 10% loss link and an ordered, 20% loss rate. It takes much shorter time to get the same amount of data with PITP than TCP and TCPW, since out-of-order packets are allowed in PITP.

6.2.2 Comparison of Transferring Images With PITP and SITP

We conducted a set of experiments comparing different views of JPEG images encoded in sequential mode and progressive mode respectively. Each experiments transmits a 32.1KB image over a 802.11b wireless link at a loss rate of 0%. Our PITP is used to transmit the progressively encoded image, and an SITP (PITP without support for progressive JPEG) is used for the sequential encoded image. SITP uses the same code as in PITP for all aspects except that the image processing of progressive JPEG images has been modified to cater for sequential JPEG images.

Figure 6.7 graphs the sequence of images being displayed at various time. Four scans of the progressive JPEG image are displayed in the right column. The first scan of image

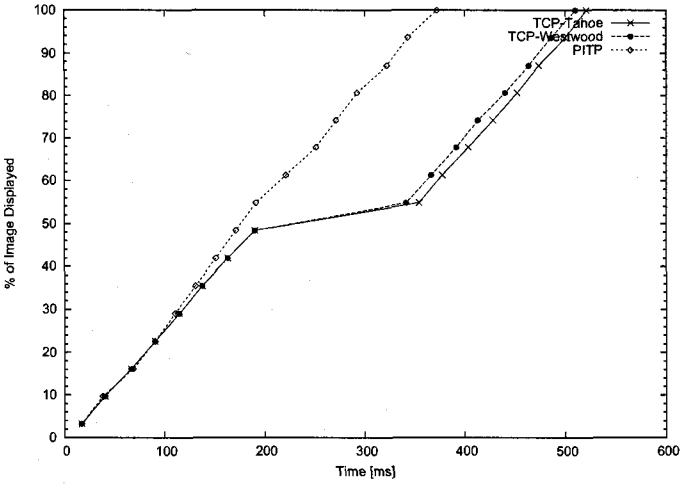

Figure 6.3: Comparison of TCP-like protocols and PITP at 10% loss rate

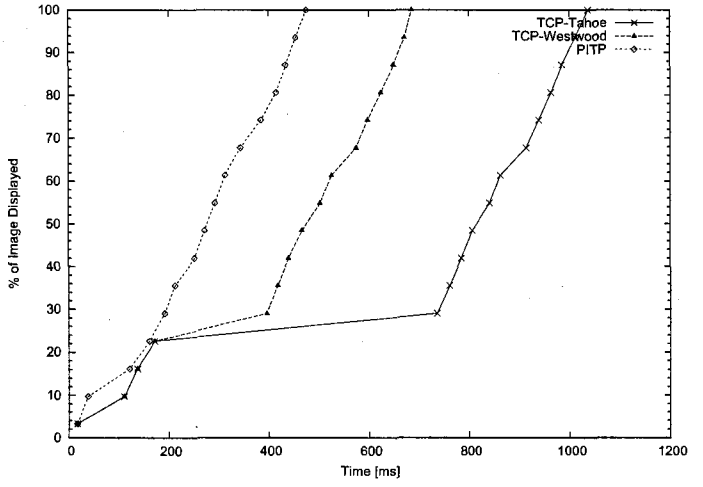

Figure 6.4: Comparison of TCP-like protocols and PITP at 20% loss rate

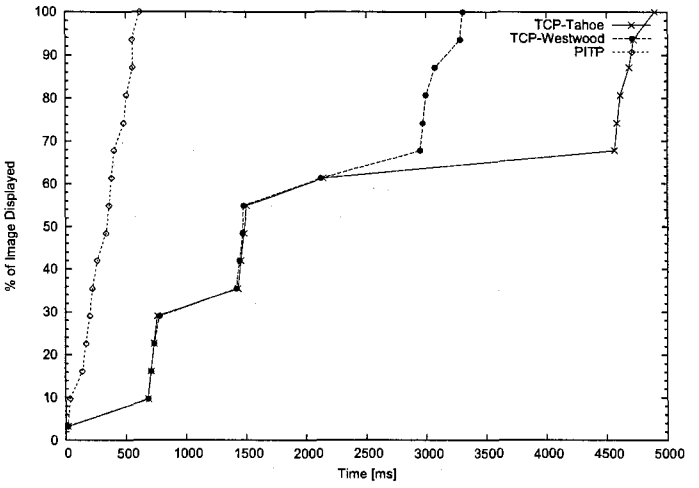

Figure 6.5: Comparison of TCP-like protocols and PITP at 30% loss rate

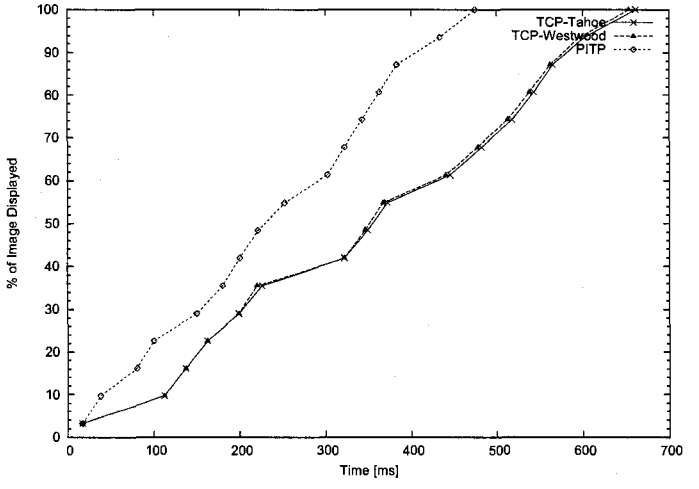

Figure 6.6: Comparison of TCP-like protocols and PITP at 10% loss rate, unordered

Figure 6.7: Sequence of displaying sequential and progressive JPEG images

Figure 6.8: Images for testing synchronization control

is very coarse, and the image quality is increased gradually in the following scans. In the left column, partial views of the sequential JPEG image at the same time of each scan of the progressive one being displayed, are provided. According to our experiment, 7ms after the start of transmission, a rough preview of the whole image can be obtained from the first scan of the progressively encoded image. However, at the same time, little information can be learned from the first stripes of the sequential image. Actually, it is difficult to understand the sequential image until 20ms, or even 37ms have passed, when an integrate and high quality progressive image is already available. Therefore, given the same transmission time, compared with sequential images, partly displayed progressive images can give more useful information for a comprehensive understanding of the whole image. As a result, the support for progressive JPEG provided by PITP is necessary and meaningful.

6.2.3 Synchronization Control of PITP

A set of experiments is conducted to test the synchronization control mechanism of PITP. Three images of various sizes: *flower.jpg* (flower), *lena.jpg* (lena) and *monalisa.jpg* (monalisa), as shown in Figure 6.8, are transmitted in parallel from three senders to the same receiver.

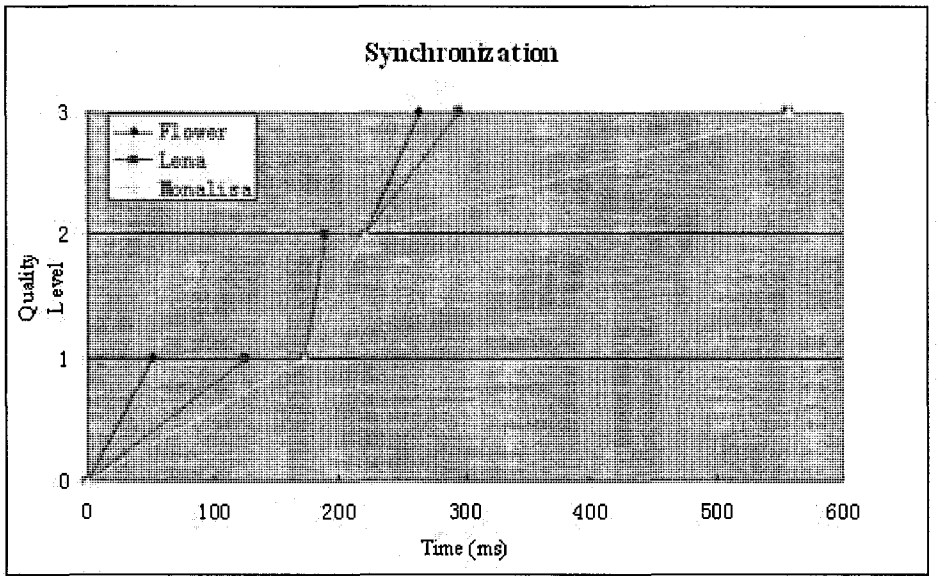

Figure 6.9: Evaluation of synchronization control

Each image is transmitted at three levels of quality. Figure 6.9 graphs the sequence of the images displayed in increasing quality level at varying time. As the file size of monalisa is larger, it takes longer to be transmitted. However, it can be observed that even the transmission of flower and lena at quality level 1 has completed, the transmission at quality level 2 is not started immediately. It is suspended until the transmission of monalisa at the first level of quality has finished. Similarly, flower and lena are not transmitted at quality level 3 until the transmission of monalisa at quality level 2 has completed. This experiment proves that PITP works well at scheduling the transmission among multiple senders and can guarantee that images at high quality level are not transmitted until images from all senders have been received at a lower quality.

6.2.4 Summary

Two groups of experiments are presented in Chapter 6 to evaluate the performance of PITP. The first group of experiments compare PITP with TCP, by transmitting the same images using two protocols over (1) ordered, lossless; (2) ordered, at various loss rate; and

(3) unordered, lossy wireless links. Results show that PITP performs better than TCP for all three cases, and the advantages of PITP become more obvious as the loss rate grows. The second group of experiments compare PITP with SITP, which supports sequential JPEG instead of progressive JPEG. According to the results, at the same transmission time, partial images transmitted with PITP gives more information than sub-images obtained with SITP does. Finally, the synchronization mechanism of PITP is evaluated. Three images of different sizes are transmitted at three quality levels simultaneously. The results prove that transmission among the three senders is synchronized. Therefore, in conclusion, PITP is a reliable, progressive, and synchronous transport protocol for image transmission over lossy, order-changing, low-bandwidth wireless networks.

Chapter 7

Conclusion and Future Work

Due to the inherent characteristics of WSN such as error-prone, order-changing and energy-conserved, traditional transport protocols for wired networks are not suitable for image transmission over WSN. In this thesis, a new transport protocol – PITP – has been proposed to provide reliable, progressive and synchronous transmission of JPEG images over lossy wireless networks. In this chapter, a summary of our thesis work will be provided. Furthermore, possible directions for future research will be pointed out.

7.1 Conclusion

In this thesis, a background discussion of WSN has been presented in Chapter 2, in terms of the characteristics of WSN, general challenges of traditional transport protocols over WSN and the performance metrics for transport protocols over WSN. A great fraction of packet loss is introduced by the high bit error rate of WSN, while in wired network congestion is the only reason for loss of packets. In addition, sensors in WSN are energy-conserved, because they can only carry a limited amount of power and battery, and their storage capacity is fairly small. Therefore, due to the characteristics of WSN, transport protocols running over WSN should be able to provide reliable data transmission in an efficient approach, with appropriate congestion control mechanism. TCP and UDP are

two of the most popular transport protocols over wired networks. However, they are not suitable for WSN for the following reasons. Firstly, TCP assumes that all packet losses are caused by congestion. Whenever a packet loss is detected, a set of actions are taken to avoid congestion, such as to minimize the transmission window, and to decrease transmission rate. Therefore, throughput is significantly degraded if TCP is used directly over WSN, where a variety of packet losses result from link errors. Secondly, UDP, which is not connection-oriented and has neither congestion control nor reliability guarantee, can not provide reliable data transmission over lossy, order-changing wireless networks. As a result, new transport protocols should be designed and implemented to support reliable, energy-efficient data delivery over loss-prone and resource-limited wireless networks.

With the demand of applying surveillance technologies into civilian applications, transmission of images and multimedia stream take up a significant proportion of traffic over the networks. Therefore, proper image compression technique should be used to decrease the size of images and enhance the performance of transport protocols. In Chapter 3, the JPEG still image compression standard has been reviewed, with a detailed discussion of the compression and decompression processes. JPEG is currently the most popular compression standard, and image format as well, for storing and transferring images. JPEG provides high compression ratio without significant degradation of the image quality. Furthermore, JPEG has a progressive encoding mode, which compresses images through multiple scans with increasing quality. Progressive JPEG is useful when images are transferred over low-speed networks, such as WSN, because a rough view can be displayed for the client before all image data has been received. After the introduction of the image compression standard, some existing image transport protocols have been presented in this chapter. ITP, proposed by Raman et al., can support receiver-based, out-of-order packet delivery, and has been proved to be suitable for transferring JPEG and JPEG2000 images over lossy wireless links [55]. NCCI has applied the concept of network-consciousness into transport protocols and can support

progressive display of GIF images [53]. PCTP proposed an adaptive strategy for priority-driven compression and scheduling of progressive JPEG images [4]. Experimental results showed that both ITP and NCCI perform better than TCP in image transmission over low-bandwidth wireless links, and PCTP is also proved to be suitable for transferring images in time/power/bandwidth conserved wireless applications.

Motivated by existing image transport protocols, we proposed a new protocol, PITP, to provide reliable, progressive and synchronized image transmission over WSN, with congestion control facility. PITP supports receiver-controlled reliability, out-of-order packet delivery, as well as progressive display of images. In addition, the TCP-ELN technique, which distinguishes error-related loss from congestion-related loss, is used in the design of congestion control mechanisms. Furthermore, a synchronization control mechanism is proposed to control the transmission priority of different sensors, so that images coming from various sensors are displayed at the same quality level at a certain time. In Chapter 4 and Chapter 5, an extensive discussion of PITP has been presented. The motivation of PITP, together with the proposed environment to apply PITP, has been introduced in Chapter 4. The important characteristics of PITP have been discussed in detail, including the receiver-controlled reliability, out-of-order packet delivery, progressive output display, the congestion control policies and the synchronization control mechanism. Subsequently, implementation issues related with these PITP features have been discussed in Chapter 5.

Experiments have been carried out with the ns2 network simulator to evaluate the performance of PITP. First, a set of experiments are run to compare PITP with TCP, by transmitting the same images with two protocols over a wireless link at different loss rates. Results show that when the link is reliable and packets transmission is ordered, PITP performs slightly better than TCP-like protocols. As the loss rate rises, the superiority of PITP to TCP becomes more significant. Second, a group of experiments are conducted to compare the performance of transmitting images with different encoding modes. Results prove that given the same time during the transmission of the same

image, partly displayed lower quality but integrate progressive image gives more information than several strips of sequential image with full quality. In addition, the third experiment demonstrates the synchronization ability of PITP. Therefore, PITP performs well for transferring images over loss-prone, order-changing wireless links.

7.2 Future Work

Avenues for future research are proposed as follows:

- *Support for JPEG 2000 format.* Currently, the image format supported by our PITP is JPEG. JPEG 2000 is a new image compression standard coming more recently, and can provide higher compression ratio and better image quality. PITP can be extended to provide support for images of JPEG 2000 format.

- *Priority-oriented scheduling.* Progressive JPEG is incorporated in PITP to render a smooth display of images for the client. The interactivity between the client and the application can be further improved by enabling priority-oriented image transmission, which can specify the priority of different objects in one single scan of an image. For example, the data of those objects that the client is most interested in should be transmitted first, whereas the data of the background should be transmitted last. Therefore, most useful information could be received by the client given a short period of time. Further research could be carried out so that priority-oriented scheduling of image transmission can be supported by PITP.

Appendix A

Glossary of Terms

AC Alternating Current.

ACK Acknowledgement.

ADU Application Data Unit.

ALF Application Level Framing.

CACK Cumulative Acknowledgement.

CM Congestion Manager.

DC Direct Current.

DCT Discrete Cosine Transform.

ELN Explicit Loss Notification.

ESRT Event to Sink Reliable Transport.

HTTP Hypertext Transport Protocol.

IBMR Image-based Modelling and Rendering.

ITP ITP: An Image Transport Protocol for the Internet.

JFIF JPEG File Interchange Format.

JPEG Joint Photographic Experts Group.

MCU Minimum Coded Unit.

MJPEG Motion JPEG.

NACK Negative Acknowledgement.

NCCI Network-conscious Compressed Images.

NTP Network Time Protocol.

PITP A Progressively Reliable Image Transport Protocol.

PCTP Priority-driven Coding and Transmission of Progressive JPEG Images for Real-time Applications.

PSFQ Pump Slowly, Fetch Quickly.

QoS Quality of Service.

RLE Run Length Encoding.

RMST Reliable Multi-Segment Transport.

RTO Retransmission Timeout.

RTT Round Trip Time.

TCP Transmission Control Protocol.

UDP User Datagram Protocol.

WMSN Wireless Multimedia Sensor Networks.

WSN Wireless Sensor Networks.

Bibliography

[1] A.Bakre and B.R.Badrinath. I - TCP: Indirect TCP for mobile hosts. In *Proc.15th International Conference on Distributed Computing Systems (ICDCS)*, page 136, May 1995.

[2] A.Bharathidasan and V.A.S.Ponduru. Sensor networks: An overview. *IEEE Potentials*, 22(2):20–23, 2003.

[3] A.Ganz, Z.Ganz, and K.Wongthavarawat. *Multimedia Wireless Networks: Technologies, Standards, and QoS*. Prentice Hall, Upper Saddle River, NJ, 2004.

[4] A.M.Cheng and F.Shang. Priority-driven coding and transmission of progressive jpeg images for real-time applications. *Journal of VLSI Signal Processing Systems*, 47(2):169–182, 2007.

[5] B.Shrader. A survey of tcp performance on wireless links. Technical report, Royal Institute of Technology, Stockholm, Sweden, March 2002.

[6] C.Casetti, M.Gerla, S.Mascolo, M.Y.Sanadidi, and R.Wang. TCP westwood: end-to-end congestion control for wired/wireless networks. *J. of Wireless Net.*, 8(2):467–479, 2002.

[7] C.Intanagonwiwat, R.Govindan, and D.Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking*, pages 56–67, August 2000.

[8] C.L.Heng. Image compression: JPEG. `http://pascalzone.amirmelamed.co.il/Graphics/JPEG/JPEG.htm`.

[9] C.Wang, B.Li, K.Sohraby, M.Daneshmand, and Y.Hu. Upstream congestion control in wireless sensor networks through cross-layer optimization. *IEEE Journal on Selected Areas in Communications*, 25(4):786–795, 2007.

[10] C.Wang, K.Sohraby, B.Li, and M.Daneshmand. A survey of transport protocols for wireless sensor networks. *IEEE Network*, 20(3):34–40, 2006.

[11] C.Y.Wan, A.T.Campbell, and L.Krishnamurthy. PSFQ: a reliable transport protocol for wireless sensor networks. In *Proc. of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 1–11, Septermber 2002.

[12] D.Chen and P.K.Varshney. Qos support in wireless sensor networks: A survey. In *Proc. of International Conference on Wireless Networks.*, pages 227–233, June 2004.

[13] D.Clark and D.Tennenhouse. Architectural considerations for a new generation of protocols. In *ACM SIGCOMM'90*, pages 200–208, September 1990.

[14] D.Forsyth and J.Ponce. *Computer Vision - A Modern Approach*, chapter 26,Application:Image-Based Rendering, pages 780–808. Alan Api, 2002.

[15] D.L.Mills. Internet Time Synchronization: The Network Time Protocol. Technical report, RFC - 1129, October 1989.

[16] C. Douglas. *Internetworking with TCP/IP*, pages 175–229. Pearson Prentice Hall, Upper Saddle River, NJ, 5th edition, 2006.

[17] D.S.Taubman and M.W.Marcellin. *JPEG2000 – Image Compression Fundamentals, Standards and Practice.* Springer, Kluwer, Dordrecht, 2002.

[18] F.Stann and J.Heidemann. Rmst: reliable data transport in sensor networks. In *Proc. of 1st IEEE International Workshop on sensor network protocols and applications*, pages 102–112, May 2003.

[19] G.Buchholcz, T.Ziegler, and T.V.Do. TCP-ELN: On the protocol aspects and performance of explicit loss notification for TCP over wireless networks. In *WICON '05: Proceedings of the First International Conference on Wireless Internet*, pages 172–179, July 2005.

[20] G.K.Wallace. The jpeg still picture compression standard. *Commun. ACM*, 34(4):30–44, 1991.

[21] Independent JPEG Group. Explanation of JPEG progressive mode. `http://wwwicg.informatik.uni-rostock.de/Projekte/MoVi/jpeg/progressive.html`.

[22] Independent JPEG Group. Independent JPEG group. `http://www.ijg.org/`.

[23] H.Balakrishman and S.Seshan. The congestion manager. Technical report, RFC - 3124, June 2001. Internet Engineering Task Force.

[24] H.Balakrishnan, S.Seshan, and R.H.Katz. Improving reliable transport and handoff performance in cellular wireless networks. *ACM Wireless Networks Journal (WINET)*, 1(4):469–481, 1995.

[25] H.Balakrishnan, V.N.Padmanabhan, S.Seshan, and R.H.Katz. A comparison of mechanisms for improving tcp performance over wireless links. *IEEE/ACM Transactions on Networking*, 5(6):756–769, 1997.

[26] H.Salem and N.Mohamed. Middleware challenges and approaches for wireless sensor networks. *IEEE Distributed Systems Online*, 7(3):1, 2006.

[27] H.Y.Shum and S.B.Kang. A review of Image-based rendering techniques. In *IEEE/SPIE Visual Communications and Image Processing (VCIP)*, pages 2–13, May 2000.

[28] ImpulseAdventure. JPEG Minimum Coded Unit. `http://www.impulseadventure.com/photo/jpeg-minimum-coded-unit.html`.

[29] J.B.Postel. User datagram protocol. Technical report, RFC - 768, Aug 1980.

[30] J.B.Postel. Transmission control protocol. Technical report, RFC - 793, sri Int, Menlo Park, CA, Aug 1989.

[31] J.Elson and K.Romer. Wireless sensor networks: A new regime for time syncrhonization. *SIGCOMM Computer Communication Review*, 33(1):149–164, 2003.

[32] J.Jones and M.Atiquzzaman. Transport protocols for wireless sensor networks: State-of-the-art and future directions. *International Journal of Distributed Sensor Networks*, 3(1):119–133, 2007.

[33] K.Brown and S.Singh. M - TCP: TCP for mobile cellular networks. *SIGCOMM Comput. Commun. Rev.*, 27(5):19–43, 1997.

[34] K.Fall and S.Floyd. Simulation-based comparisons of Tahoe, Reno and SACK TCP. *ACM Computer Communication Review*, 26(3):5–21, 1996.

[35] K.R.Rao and P.Yip. *Discrete cosine transform: algorithms, advantages, applications.* Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[36] L.A.Grieco and S.Mascolo. Performance evaluation and comparison of westwood+, new reno, and vegas TCP congestion control. *SIGCOMM Comput. Commun. Rev.*, 34(2):25–38, 2004.

[37] L.Cai, X.Shen, J.Pan, and J.W.Mark. Comparative study of various tcp versions over a wireless link with correlated losses. *IEEE/ACM Transactions on Networking*, 11(3):370–383, 2003.

[38] Elysium Ltd and 2KAN. The jpeg committee home page. `http://www.jpeg.org/index.html`.

[39] M.Allman, V.Paxson, and W.Stevens. TCP Congestion Control. Technical report, RFC - 2581, Apr 1999.

[40] M.Barni. *Document and Image Compression*, pages 89–108. CRC Press, Boca Raton, FL, 2006.

[41] M.Mathis, J.Mahdavi, S.Floyd, and A.Romanow. Selective acknowledgment options. Technical report, RFC - 2018, Oct 1996.

[42] M.Mock, R.Frings, E.Nett, and S.Trikaliotis. Continuous clock synchronization in wireless real-time applications. In *Symposium on Reliability in Distributed Software*, pages 125–132, October 2000.

[43] N.Xu. A Survey of Sensor Network Applications. `http://courses.cs.tamu.edu/rabi/cpsc617/resources/sensor%20nw-survey.pdf`.

[44] P.D.Amer, S.Iren, G.E.Sezen, P.T.Conrad, M.Taube, and A.Caro. Network-conscious GIF image transmission over the Internet. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(7):693–708, 1999.

[45] P.Karn and C.Partridge. Improving round-trip time estimates in reliable transport protocols. *ACM Trans.Comput.Syst.*, 9(4):364–373, 1991.

[46] Q.Li and D.Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *MobiCom 2000*, Boston, USA, August 2000.

[47] R.Caceres and L.Iftode. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE Journal on Selected Areas in Communications*, 13(5):850–857, 2002.

[48] R.Fielding, J.Gettys, J.Mogul, H.Frystyk, and T.Berners-Lee. Hypertext transfer protocol– HTTP/1.1. Technical report, RFC - 2068, IETF, Jan 1997.

[49] R.Kay and F.Mattern. The design space of wireless sensor networks. *IEEE Wireless Communications*, 11(6):54–61, 2004.

[50] R.Kay and F.Mattern. The design space of wireless sensor networks. *IEEE Wireless Communications*, 11(6):54–61, 2004.

[51] R.Szeliski and H.Y.Shum. Creating full view panoramic image mosaics and environment maps. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 251–258, August 1997.

[52] S.C.Cha. Inter-frame prediction method in video coding, video encoder, video decoding method, and video decoder, Oct 2005. United States Patent 20050232359.

[53] S.Iren, P.D.Amer, and P.T.Conrad. Network-conscious compressed images over wireless networks. In *5th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'98)*, Oslo, Norway, September 1998.

[54] S.Mascolo, C.Casetti, M.Gerla, M.Y.Sanadidi, and R.Wang. Tcp westwood: Bandwidth estimation for enhanced transport over wireless links. In *Proc. of the 7th Int. Conf. MobiCom'01*, pages 287–297, 2001.

[55] S.Raman and H.Balakrishnan. Itp: An image transport protocol for the internet. *IEEE/ACM Transactions on Networking*, 10(3):297–307, 2002.

[56] T.Braun, T.Voigt, and A.Dunkels. Energy-efficient tcp operation in wireless sensor networks. *PIK Journal Special Issue on Sensor Networks*, 28(2), 2005.

[57] Surveillance Technology. Surveillance technology. `http://www.surveillancetechnology.com/`.

[58] T.G.Lane. USING THE IJG JPEG LIBRARY. `http://www.jpegcameras.com/libjpeg/libjpeg.html#toc3`.

[59] Wikipedia the free encyclopedia. Frame Rate. `http://en.wikipedia.org/wiki/Frame_rate`.

[60] Wikipedia the free encyclopedia. JPEG. `http://en.wikipedia.org/wiki/Jpeg`.

[61] Wikipedia the free encyclopedia. Motion JPEG. `http://en.wikipedia.org/wiki/Mjpeg`.

[62] Wikipedia the free encyclopedia. MPEG-4. `http://en.wikipedia.org/wiki/MPEG-4`.

[63] Mpeg TV. MPEG.ORG-MPEG Home. `http://www.mpeg.org/`.

[64] VINT. The Network Simulator - ns-2. `http://www.isi.edu/nsnam/ns/`.

[65] V.Paxson. End-to-end internet packet dynamics. *IEEE/ACM Trans. Netw.*, 7(3):277–292, 1999.

[66] W.Dabbous and C.Diot. High performance protocol architecture. In *IFIP Performance of Computer Networks Conference (PCN'95)*, Istanbul, Turkey, October 1995.

[67] Y.Sankarasubramaniam, O.B.Akan, and I.F.Akyildiz. ESRT: event-to-sink reliable transport in wireless sensor networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing*, pages 177–188, June 2003.