A Hybrid Packet Loss Recovery Technique in Wireless Ad Hoc Networks

Hui Yang

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science (Electrical and Computer Engineering) at
Concordia University
Montréal, Québec, Canada

October 2008

Your file Votre référence
ISBN: 978-0-494-45509-8
Our file Notre référence
ISBN: 978-0-494-45509-8

ABSTRACT

A Hybrid Packet Loss Recovery Technique in Wireless Ad Hoc Networks

Hui Yang

TCP utilization in wireless networks poses certain problems due to its inability to distinguish packet losses caused by congestion from those caused by frequent wireless errors, leading to degraded network performance. To avoid these problems and to minimize the effect of intensive channel contention in wireless networks, this work presents a new Hybrid ARQ technique for reliable and efficient packets transfer in static wireless ad hoc network. It is a combination of recent FEC based Raptor coding technique with ARQ based selective retransmission method, which outperforms purely ARQ based method. In contrast to most Hybrid ARQ techniques, which usually employ a byte level FEC, we mostly use packet level FEC in our simulations for the data transfer, on top of less frequent ARQ to recover the residual errors. Existing packet level FEC methods are mostly based on simple parity check codes or Reed Solomon codes with erasure decoding; in this work we use the recent raptor codes. We also introduce the notion of adaptive redundancy which helps to achieve better average network performance and to further improve the redundancy efficiency.

ACKNOWLEDGEMENTS

I would like to take this chance to thank my wife Qi Sun, for the love and for her always being supportive.

I would really like to thank my supervisor Dr. Ahmed K. Elhakeem for giving the directions, the great ideas and the patience. With his care through the whole process, I can finally finish this thesis.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ACK	**Acknowledgment**
AES	**Advanced Encryption Standard**
AIMD	**Additive Increase Multiplicative Decrease**
AP	**Access Point**
ARQ	**Automatic Repeat reQuest**
BCMCS	**Broadcast and Multicast Service**
BEC	**Backward Error Correction**
BER	**Bit Error Rate**
BLR	**Block Loss Rate**
BP	**Belief Propagation**
BS	**Base Station**
BSS	**Basic Service Set**
CDMA	**Code Division Multiple Access**
CRC	**Cyclic Redundancy Check**
CSMA/CA	**Carrier Sense Multiple Access with Collision Avoidance**
DARPA	**Defense Advanced Research Projects Agency**
DoD	**Department of Defense**
DS	**Distribution System**
DSSS	**Direct Sequence Spread Spectrum**
DV	**Distance Vector**
DVB	**Digital Video Broadcasting**
DVB-H	**Digital Video Broadcasting – Handheld**
ECC	**Error Correction Code**
ECN	**Explicit Congestion Notification**
ED	**Error Detection**

ESS	**Extended Service Set**
ETSI	**European Telecommunications Standards Institute**
FCC	**Federal Communications Commission**
FEC	**Forward Error Correction**
FHSS	**Frequency Hopping Spread Spectrum**
FTP	**File Transfer Protocol**
GBN	**Go Back N**
GloMo	**Global Mobile Information Systems**
HARQ	**Hybrid ARQ**
HiperLAN	**High Performance Radio LAN**
IBSS	**Independent Basic Service Set**
IC	**Integrated Circuit**
ID	**Identification**
IEEE	**Institute of Electrical and Electronics Engineers**
IETF	**Internet Engineering Task Force**
IP	**Internet Protocol**
IR	**Infrared**
IS-IS	**Intermediate System to Intermediate System**
ISM	**Industry, Medical and Scientific**
ISN	**Initial Sequence Number**
LAN	**Local Area Network**
LBT	**Listen Before Talk**
LDPC	**Low Density Parity Check**
LLC	**Logical Link Control**
LS	**Link State**
LT	**Luby Transform**
MAC	**Media Access Control**

MANET	**Mobile Ad Hoc Networks**
MBMS	**Multimedia Broadcast Multicast Service**
MDS	**Maximum Distance Separable**
MIMO	**Multiple Input Multiple Output**
MSS	**Maximum Segment Size**
NACK	**Negative Acknowledgement**
NIC	**Network Interface Card**
NTDR	**Near Term Digital Radio**
OFDM	**Orthogonal Frequency Division Multiplexing**
OSPF	**Open Shortest Path First**
PDA	**Personal Digital Assistants**
PDU	**Protocol Data Unit**
PHY	**Physical**
PLCP	**Physical Layer Convergence Procedure**
PLR	**Packet Loss Rate**
PMD	**Physical Medium Dependant**
PRNET	**Packet Radio Networks**
PSK	**Phase Shift Keying**
QAM	**Quadrature Amplitude Modulation**
QoS	**Quality of Service**
RAID	**Redundant Array of Inexpensive Disks**
RF	**Radio Frequency**
RS	**Reed Solomon**
RTO	**Retransmission Time Out**
RTP	**Real time Transport Protocol**
RTS/CTS	**Request to Send / Clear to Send**
RTT	**Round Trip Time**

SACK	**Selective Acknowledgment**
SIFS	**Short Interframe Space**
SR	**Selective Repeat**
SSID	**Service Set Identifier**
SURAN	**Survivable Adaptive Radio Networks**
TCP	**Transmission Control Protocol**
UDP	**User Datagram Protocol**
U-NII	**Unlicensed National Information Infrastructure**
UPCS	**Unlicensed Personal Communications Services**
VoIP	**Voice over IP**
VoWLAN	**Voice over wireless LAN**
WEP	**Wireless Equivalent Privacy**
Wi-Fi	**Wireless Fidelity**
WMN	**Wireless Mesh Networks**
WPA	**Wi-Fi Protected Access**
WSN	**Wireless Sensor Networks**
WWW	**World Wide Web**

Chapter 1

Introduction

There are generally two major types of wireless networks, i.e., the infrastructure networks and the ad hoc networks which will be discussed in the next section. Infrastructured network is predesigned, and has permanent network device deployments. It can be implemented with a fixed or dynamic topology. In the case of fixed topology, a wireless host can be connected via a fixed point, known as an access point (AP) or a base station (BS). An AP is usually connected to the backbone network via a wired link. Cellular networks and most of the wireless LANs work as the static infrastructured networks.

Wireless Local Area Networks (LANs) have been used almost everywhere today, the field of wireless LAN is increasing rapidly as a result of an extensive availability of wireless devices and services, and developments in semiconductor and digital communication technology. In this chapter, a general introduction of wireless LANs and wireless ad hoc networks is presented, and we also talk about some developing techniques at the end. The first section reviews wireless LANs and wireless ad hoc networking is discussed in the second section.

1.1 Overview of Wireless Local Area Networks

Wireless LANs use spread-spectrum or OFDM (Orthogonal Frequency Division

Multiplexing) modulation technology based on radio waves to enable communication between devices in a limited area, also known as the Basic Service Set (BSS). This gives users the mobility to move around within a broad coverage area and still be connected to the network. The popularity of wireless LANs in fact came together with the increase of residential high-speed broadband Internet access. It was and remains the simplest way to share a broadband link between several computers spread over a residence. In addition, the expansion of hotspots and public access points has drastically raised its popularity.

IEEE 802.11, also more popularly known as 'Wi-Fi' (Wireless Fidelity) [1], solves security, mobility, reliability, and the dynamic feature of wireless LANs while keeping compatibility with 802 legacy networks. IEEE 802.11 is the de facto standard in wireless LAN technologies, although there is another wireless LAN standard HiperLAN (High Performance Radio LAN), which is an IEEE 802.11 alternative developed in Europe. The first version of HiperLAN called HiperLAN/1, was originated by the European Telecommunications Standards Institute (ETSI) in 1991, aiming to achieve a data rate higher than 802.11. However, the latest HiperLAN/2 is not doing well in the market, especially since the faster 54 Mbps 802.11a (5 GHz) and 802.11g (2.4 GHz) came out [2]. In this section, we will mostly discuss the IEEE 802.11 based wireless LANs.

In wireless networks, signals transmissions are broadcast and may interfere with each other. A collision will be sensed and transmissions may fail when there are concurrent transmissions within the signal coverage of communicating parties. Consequently, a medium access protocol is necessary to organize the transmission accesses of the wireless

channel so as to achieve a reasonably high throughput and channel utilization. Unlike wired networks, signals transmitted over wireless media may be weakened or twisted because they are propagated over an open and varying medium with irregular boundary. In addition, the same signal may disperse and travel on different paths due to reflection, diffraction, and scattering caused by obstructions before it gets to the destination. The dispersed signals on different paths may take different long times to reach the destination. Therefore, the total signal after summing up all dispersed signals may have been considerably deformed and attenuated compared to the source signal. The receiver may not identify the signal and thus the transmitted data cannot be received. This unpredictable characteristic of wireless medium causes large numbers of packet losses.

1.1.1 Evolution

The first generation of wireless data modems was developed in the early 1980's, some amateur radio enthusiasts added a voice band data communication modem, with data rates below 9600 bps, to an existing short distance radio system, normally in the two meter amateur band.

Later in 1985, the Federal Communications Commission (FCC) released several bands of the wireless spectrum for non-military usage, before that these so-called "garbage bands" were already used in equipments such as microwave ovens that use radio waves to heat food. Right after the announcement of FCC, wireless modems offering data rate on

the order of hundreds of Kbps was developed, and the second generation of wireless modems was conceived.

The third generation products were then produced with data rates higher than 1 Mbps, focusing on the compatibility with the existing LANs with data rates of several Mbps. During the same period, the IEEE 802.11 committee was launched in 1990 to create a standard for wireless LANs. Before the foundation of IEEE 802.11 committee, there were already some early wireless LAN products in the market. Gradually, the technology became more developed and was better applied in various applications. Meanwhile, the Integrated Circuit (IC) technology related to wireless LAN applications and implementations, a main driving technology of fast developing market, was springing up in the market.

Finally, the Institute of Electrical and Electronics Engineers (IEEE) published IEEE Standard 802.11 in 1997, the first wireless LAN standard. This standard, developed by the IEEE LAN/MAN Standards Committee (IEEE 802) in the 5 GHz and 2.4 GHz public spectrum bands, defines the Media Access Control (MAC) and physical (PHY) layers for a LAN with wireless connectivity. It aims at local area networking where the connected devices communicate with other neighbor devices with radio waves. The standard is similar in most aspects to the IEEE 802.3 Ethernet standard. Nevertheless, in particular, the 802.11 standard addresses:

- Data security and user privacy
- Some physical layer signaling techniques and interfaces

- Data delivery services to upper layers and MAC
- Functions to either run in ad hoc mode or integrate with existing wired LANs
- Mobility management between wireless LANs and operation within overlapping wireless LANs

802.11x	Release	Frequ. (GHz)	Typical Thru. (Mbps)	Max Thru. (Mbps)	Modulation	Indoor Range (m)	Outdoor Range (m)
-	1997	2.4	0.9	2	IR/FH/DSSS	~20	~100
a	1999	5	23	54	OFDM	~35	~120
b	1999	2.4	4.3	11	DSSS	~38	~140
g	2003	2.4	19	54	OFDM	~38	~140
n	2009(exp.)	2.4, 5	74	600	OFDM	~50	~5000

Table 1.1: Summary of major IEEE 802.11 standards [3]

Over the next two years, two 802.11 variants were approved, they are 802.11 b which operates in the Industry, Medical and Scientific (ISM) bands of 2.4 GHz and 802.11a which operates in the Unlicensed National Information Infrastructure (U-NII) bands of 5.3 GHz and 5.8 GHz. It is common today in a coffee house or shopping mall, you can take advantage of wireless access while you are enjoying your coffee or having a rest; more and more home users will choose wireless due to the simplicity of installation and mobility when using a laptop. Large wireless network projects are under construction in many big cities, planning to cover the whole city area with wireless access.

The most recent variant was 802.11g, similar to 802.11a, uses a more advanced type of modulation OFDM, but it is used in the 2.4 GHz band. 802.11g can also achieve

speeds of up to 54 Mbps. The IEEE 802.11 standard and its variants and alternatives, such as the wireless LAN interoperability forum, and the European HiperLAN specification had made considerable impact and capacity improvement. The Unlicensed Personal Communications Services (UPCS) and the proposed U-NII bands brought in new chances as well.

1.1.2 Architectures and techniques

Wireless LANs are generally employed as the final link between the wired network and the wireless users in business, giving these users wireless access to the complete services and resources of the corporate network across a building or campus setting. The pervasive acceptance of wireless LANs depends on industry standardization to ensure product reliability and compatibility among a variety of producers.

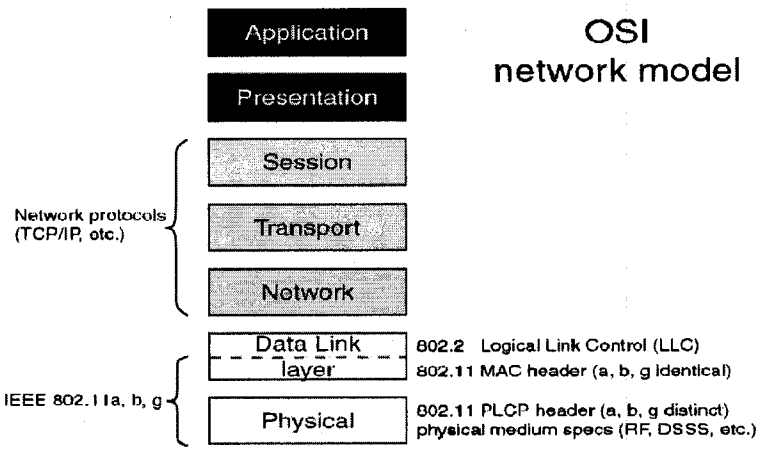

Figure 1.1: IEEE 802.11 standards mapped to OSI reference model

As all IEEE 802 standards, the 802.11 standards concentrate on the bottom two layers of the ISO model, see Figure 1.1, the physical layer and link layer. Any LAN application,

such as protocols including TCP/IP, network operating system like Novell NetWare, can run on a wireless LAN compliant with IEEE 802.11 standard, exactly the same as they are running over Ethernet.

In the architecture of wireless LAN, a service set is a logical grouping of devices. Wireless LANs provide wireless network access by broadcasting signal over wireless Radio Frequency (RF) carrier to all the users within coverage. The receiver can be within range of several transmitters. The transmitter initiates its transmissions with a Service Set Identifier (SSID). The receiver uses the SSID to sort out through the received signals and find the one it wants to listen to.

Wireless LAN Station

The station is the most essential element of a wireless network. A station is also called a node; it has the functionality of the 802.11 protocol, and a component that can connect to a wireless medium. In general 802.11 functions are implemented in hardware and software of wireless Network Interface Card (NIC). A station could be an AP, a client. APs are BS for wireless network. They transmit and receive radio signals for wireless clients to communicate with. Wireless clients can be portable devices such as laptops, Personal Digital Assistants (PDA), or fixed devices such as desktops with a wireless network interface card. Stations may be either still or mobile, and all stations support the 802.11 standard functions, including services of data delivery, privacy, authentication and de-authentication.

Basic Service Set

The basic building block of the wireless LAN network in 802.11 is BSS. The BSS consists of a group of stations, it defines a coverage area where all stations within the BSS maintain completely connected. Each BSS has a SSID which is a 32-byte maximum character string. For instance, "linksys" is the default SSID of Linksys wireless routers.

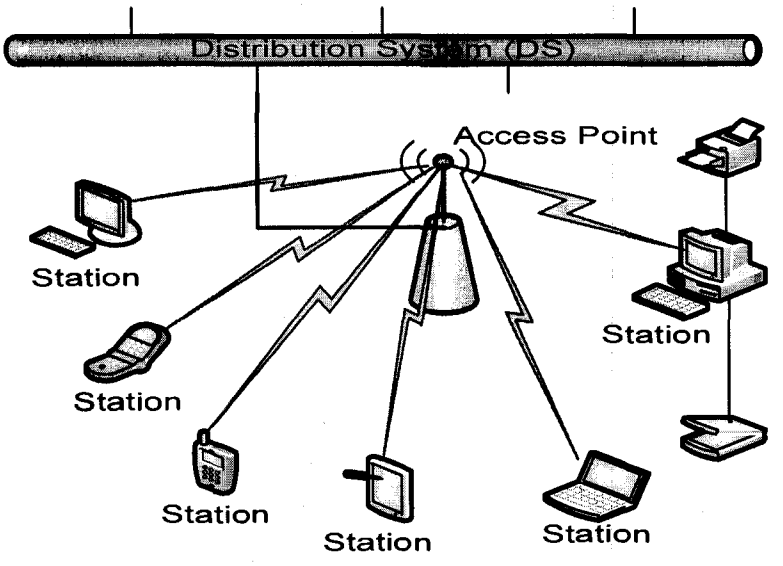

Figure 1.2: Infrastructure Basic Service Set

BSS has a dedicated station known as an AP. The AP is the heart of communications for all stations in the BSS. Every BSS has an identification (ID) called the BSSID, which is the MAC address of the AP servicing this BSS. The client stations do not communicate directly with other client stations, they communicate with the AP, and then the AP forwards the frames to the destination stations. The AP might be installed with an uplink port that connects the BSS to a wired network, e.g., an Ethernet. As a result, BSS can also be referred to as an infrastructure BSS. An infrastructure BSS can communicate with

other stations not in the same BSS by communicating through access points. Figure 1.2 demonstrates a classic infrastructure BSS.

Independent Basic Service Set

An Independent Basic Service Set (IBSS) contains no access points, and they can not connect to any other BSS. In this topology, all stations within the BSS directly communicate with each other via the wireless media in a peer-to-peer fashion. In that case, one station initiates the BSS network and other stations connect to it, every station

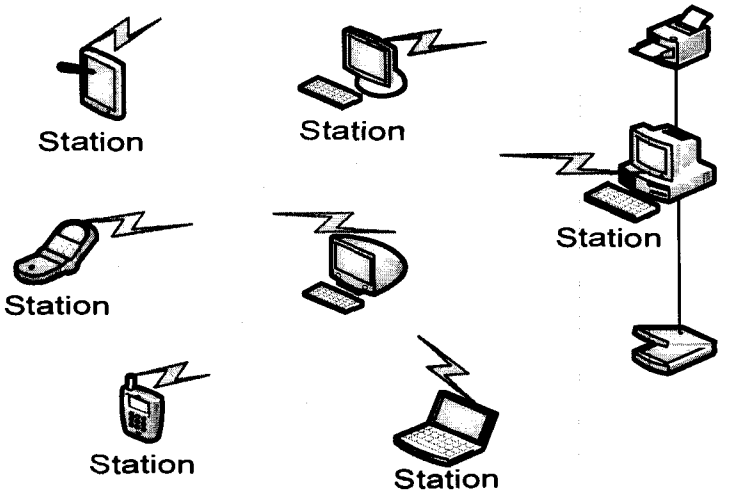

Figure 1.3: Independent Basic Service Set (IBSS)

may not be able to communicate with every other station due to the signal coverage limits. Also known as ad hoc networks, IBSS networks provide limited support for 802.11 privacy and authentication services for BSS network. IBSS network is characteristically limited both temporally and spatially. Figure 1.3 illustrates how the stations equipped with wireless NIC can structure an IBSS and communicate directly with each another.

Distribution System

The Distribution System (DS) is used by AP to communicate with another AP to exchange frames for stations in their own BSSes, forward frames to track mobile stations as they move from one BSS to another, or exchange frames with a wired network. A DS connects APs in an Extended Service Set (ESS), so as to increase network coverage by roaming between BSSes. As IEEE 802.11 describes, the DS is not necessarily a network, the standard sets no constraints on how DS is constructed, only on the services it must provide. Therefore a DS may be a wired network like 803.2 or any equipment that interconnects the APs and provides necessary distribution services.

Extended Service Set

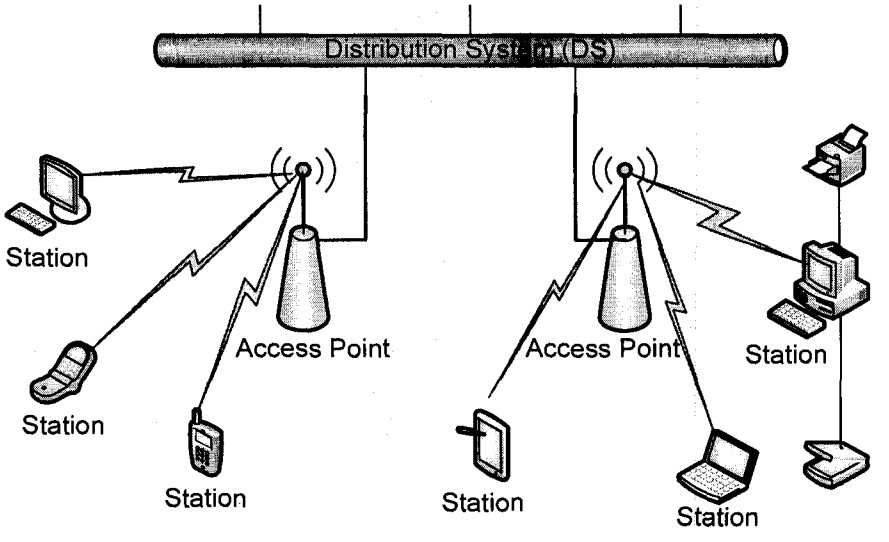

Figure 1.4: Extended Service Set (ESS)

An Extended Service Set (ESS) is a group of infrastructure BSSes interconnected via DS, where the APs communicate with each other to forward frames between BSSes, smoothing the progress of stations' movement across the BSSes. 802.11 uses ESS to

extend the range of BSS. Several infrastructure BSSes can be connected through their uplink interfaces. In 802.11, the uplink interface connects the BSS to the DS. The DS is the backbone of wireless LAN and the uplink to DS is not necessarily wired link, 802.11 specification provides the potential for this link to be wireless. Nonetheless, for now the DS uplinks are typically wired Ethernet.

The DS decides if a frame should be passed back to a destination in the same BSS, forwarded along the DS to another AP, or sent into the wired network to a destination not in this ESS. Network equipment outside of the ESS regards the whole ESS as a single MAC layer network where all stations are physically stationary. As a consequence, the ESS hides the mobility of the stations from the world outside of the ESS. This allows existing network protocols without conception of mobility to run properly with a wireless LAN where there is mobility. Figure 1.4 shows a typical topology of an ESS.

802.11 Physical Layer

The 802.11 PHY has two essential sublayers: Physical Layer Convergence Procedure (PLCP) and Physical Medium Dependant (PMD). The PLCP is actually an interconnecting layer that allows MAC Protocol Data Units (PDUs) to be transferred between MAC stations over the PMD, which is the way of transmitting and receiving data through the wireless medium. To some point, you can think of the PMD as a wireless transmission service function that is interfaced via the PLCP. The PLCP and PMD sublayers vary based on 802.11 types [4].

The IEEE 802.11 standard supports several wireless LAN technologies in the unlicensed bands of 2.4 GHz, and shares the same MAC over two PHY layer specifications: Direct Sequence Spread Spectrum (DSSS) and Frequency Hopping Spread Spectrum (FHSS) technologies. Infrared (IR) technology is also supported, yet it has not been applied by many manufacturers. FHSS and DSSS are essentially different signaling mechanisms and will not interoperate with each other.

In the DSSS technique, the 2.4 GHz band is split into 14 channels of 22 MHz each. DSSS spreads a signal on a bigger frequency band by multiplexing it with a signature or code to reduce localized interference and background noise. To spread the signal, each bit is modulated by a code. In the receiver, the original signal is recovered by receiving the entire spread channel and demodulating with the same code used by the transmitter; while in the FHSS technique, the 2.4 GHz band is split into plentiful of channels. FHSS uses a group of narrow channels and "hops" through all of them in a predestined sequence, e.g., the 2.4 GHz frequency band is split into 70 channels of 1 MHz each. Every 20 to 400 ms the system "hops" to a new channel following a predetermined cyclic pattern.

802.11 Data Link Layer

The data link layer in 802.11 consists of two sublayers: Logical Link Control (LLC) and Media Access Control (MAC). 802.11 uses the same 802.2 LLC and 48-bit addressing as other 802 LANs, allowing for really basic bridging from wireless to IEEE wired networks, but the MAC is exclusive to wireless LANs. The 802.11 MAC is really similar to 802.3 in that it is proposed to support multiple users on a common medium by forcing the

sender to sense the medium before accessing it, that is to say it is a Listen Before Talk (LBT) mechanism.

The 802.11 MAC provides functionality to offer reliable data delivery for the upper layers over the wireless channel. The data delivery itself is based on the connectionless, best-effort, asynchronous delivery of MAC layer data, so there is no guarantee that the frames will be delivered successfully. The 802.11 MAC provides a managed access method to the shared wireless media called Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). CSMA/CA is similar to the collision detection access method employed by 802.3 Ethernet LANs. Another function of the 802.11 MAC is to protect the data being delivered by offering security and privacy services. Security is offered by the authentication services and Wireless Equivalent Privacy (WEP), an encryption service for data delivered on wireless LANs.

RTS/CTS

Basically, 802.11 only uses physical carrier sensing to solve signal interference problem in wireless LANs. However, physical carrier sensing is known to suffer from the hidden node problem. RTS/CTS (Request to Send/Clear to Send), also known as a virtual carrier sensing, is an additional MAC technique used by 802.11 to reduce frame collisions due to the hidden node problem.

RTS/CTS works as follows: At first, the node willing to send data initiates the process by broadcasting a RTS message; once received the RTS message, the destination node replies with a CTS message. Any other node received a RTS or CTS message will

restrain from sending data for a certain period, this solves hidden node problem. The waiting time is included in both RTS and CTS message. If the sender does not received CTS message within a certain period, it retransmits RTS message according to a back-off algorithm. After a successful exchange of RTS and CTS message, data can then be sent by the sender after waiting for a Short Interframe Space (SIFS). This protocol assumes that all nodes have the same signal coverage. Figure 1.5 illustrates the RTC/CTS Four-Way Handshake.

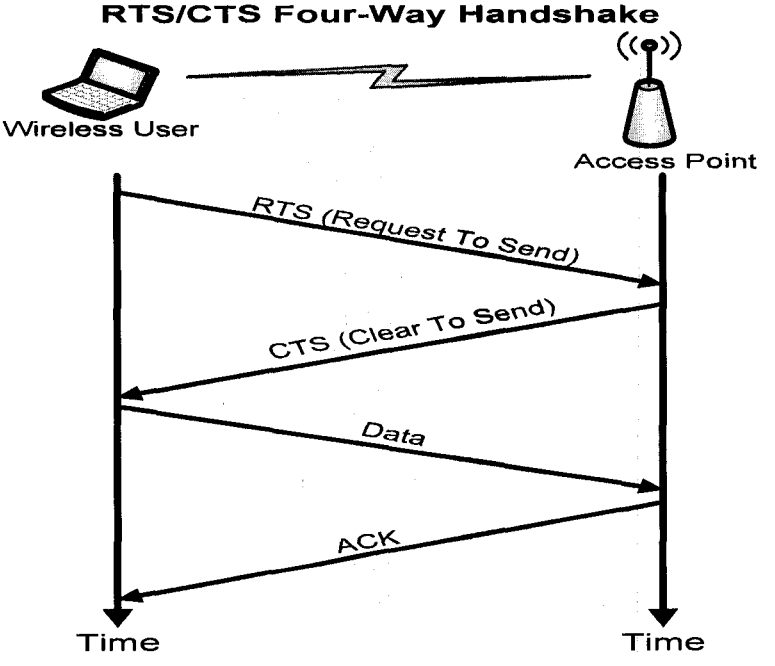

Figure 1.5: RTC/CTS Four-Way Handshake

However, RTS/CTS increases the protocol overhead which may be important for small packets, thus the efficiency of RTS/CTS depends on packet size. Therefore, RTS/CTS is generally used for large packets, where retransmissions are costly from bandwidth standpoint.

Advantages

- Mobility: Released from traditional network connections, network users can move about almost without constraint and connect to LANs from almost everyplace. Wireless LANs free users from dependence on wired access to the network backbone, providing them network access anytime, anywhere.
- Cost effective: Network setup for places hard to wire, such as old buildings and solid-wall structures and cut the cost of owner, especially in dynamic locations requiring frequent alterations, achieve minimal wiring and installation costs.
- Ease of installation: Initial configuration of an infrastructure wireless LAN needs nothing but a single AP. However, for a wired network, there are added costs and complexity of actual cables to be arranged in various places. Furthermore, it can even more difficult for locations that are not easy to get.
- Expandability: Wireless LANs can accommodate extra wireless clients on the fly, without changing the current topology and affecting existing users. On the other hand, it would require additional wiring and may affect other users in a wired network.

Disadvantages

- Security: 802.11-based wireless LANs use RF as the carrier signal for data, the data is broadcast from the sender supposing that the receiver is in the coverage of RF, yet any other station within range of the RF can also receive the data. Hence the users generally make use of various encryption technologies such as Wi-Fi Protected Access (WPA). Some older encryption methods are known to have weaknesses, such

as WEP, which has been widely criticized and considered unsuitable for secure networks. There are numerous security risks relating to the current wireless protocols and encryption methods. Cracking has become much simpler and handier with convenient Windows-based and Linux-based free software appearing on the web. Security is being studied in the IEEE by Task Group I, it is anticipated that a more secure solution will be standardized in the near future. The Advanced Encryption Standard (AES) is expected to be standardized for securing the air interface. The research on security is being done in Task Group I of the IEEE 802.11 experts.

- Stability: RF signals are subject to a broad diversity of interference and compound propagation effects, e.g. multipath, especially in case of Rician fading. Modulation techniques, such as Quadrature Amplitude Modulation (QAM) and various sorts of Phase Shift Keying (PSK) further amplify the propagation effects and interference, thus wireless access is hardly ever used in many vital systems like database servers. Even microwave oven has an effect on the reliability of wireless LANs.
- Coverage: The typical range of a regular 802.11g network is on the order of tens of meters, not as much as necessary in a spacious configuration. For larger coverage, there has to be extra cost to add more APs or repeaters.
- Speed: Wireless devices are almost always slower than the same network using a wired configuration. Relatively speaking, most wireless networks (typically 1 – 108 Mbps) is quite slow compared to the slowest ordinary wired networks (100 Mbps up to several Gbps). TCP and its integrated congestion avoidance mechanism could also

reduce the performance of wireless LANs. In some particular circumstances, wireless throughput could be insufficient.

1.1.3 Future development

Wireless LANs has a promising future with 802.11 leading the way as the standard. We can anticipate that the availability of 802.11 related products will increase dramatically in the near future as businesses realize the increased productivity provided by wireless networks.

Wireless LAN technology has experienced astonishing developments in rate, range, and spectral efficiency, which was originally limited by regulatory policy related to the use of unlicensed spectrum. With the fast deployments of 802.11 based wireless LANs around the world, standards organizations are moving towards more advanced versions for the wireless LAN application. The development of the Wireless LAN as an essential element of the future worldwide seamless wireless service is not restricted to the air interface. The evolution crosses the architecture and seamless integration of wired lines and wireless services; particularly with the introduction of real-time services such as Voice over wireless LAN (VoWLAN) and Video over wireless LAN and the seamless integration of wired lines and wireless services.

IEEE 802.11n

The latest version of IEEE 802.11n draft 4.0 was approved in May 2008 and it is expected to be finalized at the end of 2009. The industry is working aggressively to try to

make sure that existing 802.11n draft products will be able to be software upgraded to the final 802.11n standard. However, there is no guarantee that this will be the case. 802.11n requires entirely new hardware on APs and clients. In some case, the high throughput of 802.11n presents a considerable scalability challenge for products that offer encryption and decryption on the wireless switch, requiring big upgrades. Current 802.11g and 802.11b work in the 2.4 GHz band, and 802.11a works in the 5 GHz band. The 802.11n standard will work in the 2.4 GHz, the 5 GHz radio band, or both bands, providing backward compatibility with existing 802.11a/b/g networks. Most Wi-Fi products and APs hitting the market are dual-band, working in both the 2.4 GHz and 5 GHz frequencies. The final result for business will be a change to greater utilization of the 5 GHz band with 802.11n given the greater available capacity and cleaner frequency.

There are three important features included in the current 802.11n draft are Multiple Input Multiple Output (MIMO), channel bonding and frame aggregation.

Multiple Input Multiple Output

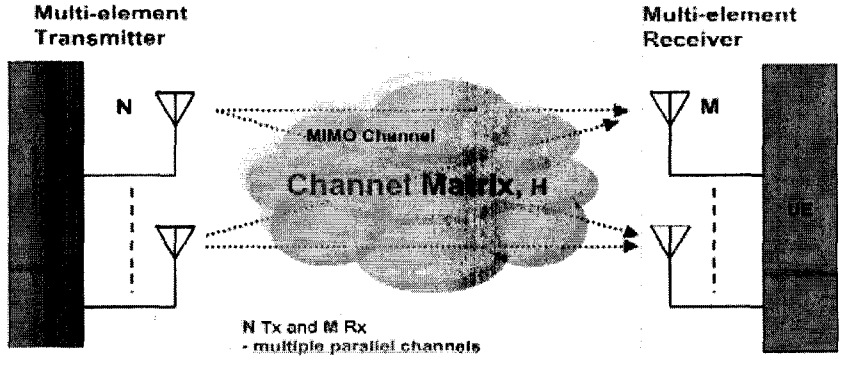

Figure 1.6: The MIMO technology [5]

MIMO is the technique to concurrently transmit multiple radio streams, delivering two or more times the data rate per channel. MIMO improves spectral efficiency by using the same amount of bandwidth to get much higher throughputs. Other than spectral efficiency, MIMO alleviates multipath effects, an extensive source of interference. Figure 1.6 shows a simple sketch for the MIMO technology.

Channel bonding

Channel bonding is a contentious feature in the current 802.11n draft. Conventional 802.11 technologies use a 20MHz-wide channel to transmit and receive. However, 802.11n adopts a technique called channel bonding to merge two neighboring 20 MHz channels into a single 40 MHz channel. Channel bonding is most efficient in the 5 GHz frequency given the much more available channels, while the 2.4 GHz frequency has only 3 non-overlapping 20 MHz channels. For that reason, only two thirds of the total frequency capacity is utilized.

Frame aggregation

802.11 has big inefficiencies in channel acquisition and back-off delays. It is normal that more than half of the time is wasted on the back-offs prior to transmission. 802.11n technologies improve efficiency by aggregating multiple packets of application data into a single transmission frame, so 802.11n networks can send multiple packets with fixed overhead cost of a single frame. Frame aggregation is more helpful for certain applications such as file transfers thanks to the ability to aggregate packet content.

Wireless LAN versus wired LAN

Only a few years ago this was not an issue, there was no doubt that people with regular mind would exchange a dedicated 100 Mbps or even Gigabit wired Ethernet connection for a shared 54Mbps wireless one. However, today with the emergence of 802.11n technique, most students with wireless LAN access use it as their most convenient way to connect to the Internet, labs, campus database system, and libraries. More and more, it is also happening in business world, wireless clients expect to interconnect from everywhere.

More and more people are using wireless networks instead of wired networks, basically there are two reasons: first, with the fast developing software and hardware techniques, manufactories can now integrate wireless LAN functions directly into Ethernet switches. The infrastructure will deal with both wired and wireless access with integrated management and security; second, and more significantly, many famous manufactories such as Cisco are now releasing next generation Wi-Fi gear based on the draft 802.11n standard. People can imagine shared throughput of 150M to 200Mbps to start and over 300Mbps in some high-demand equipments soon. Many experts declare that this wireless technology has a potential data rate of 248 Mbps, which may finally allow consumers to move beyond traditional wired Ethernet LANs. As more companies choose to use new or upgraded corporate wireless LANs, it is being wildly accepted that 802.11n, the next generation high-throughput Wi-Fi, will soon end the era of Ethernet.

A key point in the development of wireless LAN is to provide Voice over wireless LANs as well as other real time services over a real packet switched air interface, so the finalization of 802.11e is an important progress. The 802.11e enables the introduction of real time services into wireless LANs and the 802.11n introduces speeds higher than 100 Mbps, customers can expect a QoS comparable to that of a wired link with the combination of centralized topology and dense deployment.

Another noteworthy development in wireless LAN technology is the emergence of wireless mesh networks. Mesh networks have the potential to dramatically increase the area served by a wireless network. Mesh networks even have the potential, with sufficiently intelligent routing algorithms, to improve overall spectral efficiencies attained by selecting multiple hops over high capacity links rather than single hops over low capacity links.

1.2 Wireless ad hoc network

An ad hoc network is a network where all the nodes communicate in peer-to-peer mode. There are no APs and no one gives approval to communicate. It can also be called infrastructureless network or IBSS, Figure 1.3. Generally these networks are spontaneous and can be quickly set up. It is ad hoc for the reason that each node is prepared to forward packets for other nodes, so compared to a wired network in which the router carries out the task of routing, in an ad hoc network it is the node itself that dynamically makes routing decision according to the network connectivity. An important characteristic of ad

hoc networks is that the network connectivity and link quality may vary fast due to node mobility and power control scheme. Ad hoc networks can be constructed through any wireless technology, including RF and IR [6].

The decentralized nature of wireless ad hoc networks makes them appropriate for conditions where infrastructure is either not available or not trustable, so ad hoc networks are not dependable in the case of emergency. It may improve the scalability compared to infrastructure wireless networks. Some examples include an infrastructureless network of laptops in a conference or campus, temporary offices, and soldiers in the military field.

1.2.1 History

The first generation of ad hoc networks can be traced back to 1972, when they were called Packet Radio Networks (PRNET), sponsored by the Department of Defense (DoD) of United States. Together with ALOHA, CSMA and a sort of Distance Vector (DV) routing technique, PRNET were used in experiment to provide different networking capabilities in military environment.

The second generation of ad-hoc networks emerged in 1980s, the ad hoc network systems were further developed and put into practice in the Survivable Adaptive Radio Networks (SURAN) program by the Defense Advanced Research Projects Agency (DARPA). It provided a packet-switched network to the battlefield in an environment of no infrastructure. Later in 1990s, the idea of commercial ad hoc networks emerged with laptops and other portable devices became popular. In the meantime, the thought of a

collection of mobile nodes was brought up at quite a few research conferences. The IEEE 802.11 committee adopted the term 'ad hoc network' and experts started to study the possibility of applying ad hoc networks into other fields. The ad hoc network technology continued to develop in the interim. Some of the outcomes included the Global Mobile Information Systems (GloMo) and the Near Term Digital Radio (NTDR). The GloMo offered an office environment with Ethernet-type multimedia connectivity anywhere, anytime for handheld devices.

Ad Hoc Networks and the Internet

The spreading out of the Internet in early 1990's, in conjoint with lower priced wireless products such as 802.11 wireless LANs and Bluetooth devices, led to a rising focus on ad hoc networks. In 1997, the Internet Engineering Task Force (IETF) launched the Mobile Ad Hoc Networks (MANET) Working Group to study new routing protocols that deal with the multi-hop paths and dynamics of ad hoc networks, more than 70 protocols has been proposed for various circumstances. With the development of various routing protocols and new laptops and handheld devices being preinstalled with wireless component, businesses are starting to recognize the potential of commercial ad hoc network applications.

1.2.2 Classification

Wireless ad hoc networks can be further classified by their application into three categories: mobile ad hoc networks (MANETs), wireless mesh networks (WMN) and

wireless sensor networks (WSN).

Mobile ad hoc network

A mobile ad hoc network (MANET) [7] is a self-configuring wireless network with a random topology, consisting of mobile wireless stations with routing function. The stations are free to move at random and organize themselves randomly; therefore the topology of MANET may change quickly and unpredictably. The design of routing protocols is a difficult issue. In spite of the application, MANET requires effectual distributed algorithms to decide network organization, routing, and link scheduling. Nevertheless, it is still a complicated problem today for us to determine practical routing paths and transport packets in a decentralized environment where network topology varies frequently.

The set of applications for MANETs is diverse, ranging from small, static networks that are restricted by power sources, to large, dynamic networks. MANET can work in a stand-alone mode, or can connect to the Internet backbone. It became a popular topic for study as laptops and 802.11 based wireless LANs became prevalent in late 1990s.

Wireless Mesh Network

Wireless Mesh Networks (WMNs) [8] consist of mesh routers and mesh clients, where mesh routers have minimal mobility and form the backbone of WMNs. WMNs provide network access for both mesh and conventional clients, can be implemented in full mesh topology or partial mesh topology. In full mesh topology, each station is connected directly to every other station; while in partial mesh topology, stations are connected to

only some, of the other stations, not all of them. Mesh clients can be either stationary or mobile, and can form a client mesh network among themselves and with mesh routers.

A WMN is self-organizing and does not require manual configuration. It is also self-healing because it is unnecessary to manually reroute the packets. It is reliable and offers redundancy, the degree of redundancy is basically a function of node density. Therefore, if one node fails, all the rest can still communicate with each other, directly or through one or more intermediate nodes. WMNs work well when the nodes are located at scattered points that do not lie close to the same line.

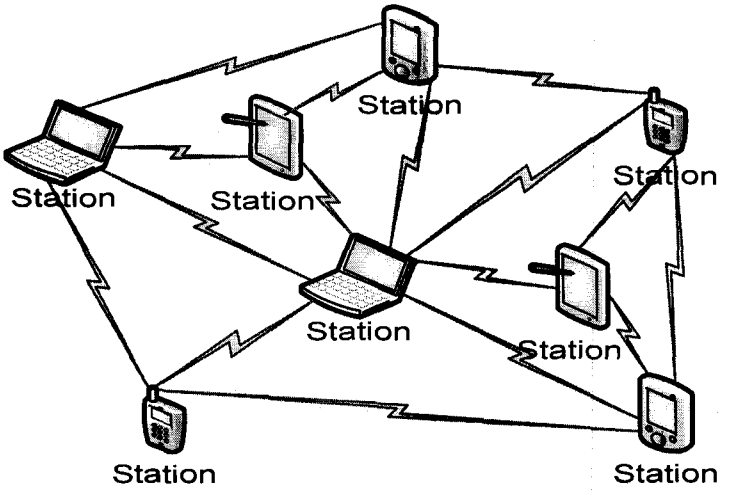

Figure 1.7: Wireless Mesh Network

WMNs are a promising technology for next generation wireless networking technology. A lot of applications are stimulating its fast growth. The integration of WMNs with other networks such as cellular, 802.11, 802.15 and 802.16, can be achieved through the gateway and bridging functions in the mesh routers. Figure 1.7 shows a simple topology of a WMN.

Wireless Sensor Network

A Wireless Sensor Network (WSN) [9] is a set of nodes organized into a cooperative network. It generally consists of a data acquisition network and a data distribution network, monitored and managed by a management center. Each node has certain processing capability, may have a RF transceiver, typically with a single omnidirectional antenna, several kinds of memory, a power source, and contain a range of sensors and actuators. The nodes usually self-organize once being deployed in ad hoc mode.

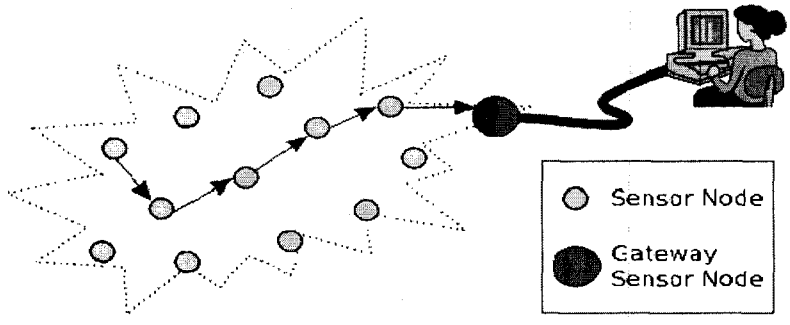

Figure 1.8: Wireless Sensor Network [10]

Unlike conventional wireless networks, WSN is usually characterized by denser node deployment, asymmetric data transmission, higher unpredictability of sensor nodes, and strict power, computation and memory limitations. These distinctive characteristics and restrictions lead to loads of new challenges for the development and final realization of WSNs. The development of WSN was initially motivated by military applications, but WSNs are currently used in many civilian application areas, such as home automation, environment monitoring and traffic control. Figure 1.8 shows a simple illustration of it.

1.2.3 Challenges

Ad hoc networking faces all kinds of challenges from many layers. The PHY layer have to adapt to rapid channel variations; the MAC layer should minimize collisions, maintain fair media access, and transfer data over the shared wireless media in the presence of fast changing situations and hidden or exposed nodes; the network layer needs to perform the routing and maintains efficiency while channel varies frequently. It also needs to incorporate efficiently with conventional networks and carry out duties such as self-configuration in the varying situation; the transport layer has to do the statistics of packet loss and delay, which is quite different than wired networks. Finally, applications need to be designed to deal with frequent disconnection and reconnection with peer applications.

In most wireless ad hoc networks the nodes compete to access the shared wireless medium, leading to lots of collisions, so wireless ad hoc networks are intrinsically limited. A main task in ad hoc network is to increase the efficiency of data transfer in severe circumstances such as power limited and very dynamic topology. Moreover, routing and transport protocols (e.g. TCP/IP) must be modified so as to improve the efficiency. Another challenge is increasing the practicability to support commercial applications. Security is possibly the most difficult problem. Real time voice and video streaming applications will only be practicable if QoS is well developed. Finally, it is important to develop middleware services that hide the complexities from high layer applications.

Chapter 2

TCP in wireless environments

TCP is the de facto standard transport layer protocol used in most applications. It was originally designed for wired networks, where random Bit Error Rate (BER) is negligible and congestion is the major source of packet loss. The popularity of various wireless network applications especially wireless Internet and high speed multimedia services, demand appropriate modifications of TCP to improve the network performance. TCP assumes that all of the packet losses are indications of network congestion, then the additive increase multiplicative decrease standard TCP congestion control gradually gets to the steady state, which represents the protocol's efficiency in terms of throughput and bandwidth utilization.

However, wireless networks suffer from high BER and user mobility. Mobile equipments experience unpredictable and momentary disruption of network connectivity when they move around. The mobility causes unstable, higher end to end delays and packet losses while the network learns how to deliver packets to the node's new location. Unfortunately, TCP mistakenly believes these delay variation and packets losses are signals of network congestion, and then suppresses its transmission rate, leading to degraded network performance. Therefore, that assumption is not suitable for situations when the end to end path contains wireless links, since many factors such as high BER, user mobility and varying channel quality may all contribute to packet losses. Numerous

researches show that the standard TCP works badly in wireless environment because it cannot differentiate the reason of packet losses.

In this chapter, section one provides a little overview of TCP fundamentals, section two roughly talks about some basic problems when TCP is used in wireless environments, and some proposed improvements focusing on the above problems are discussed in that last section.

2.1 TCP fundamentals

The Transmission Control Protocol (TCP) [11], specified in RFC 793, is one of the core protocols of the Internet protocol suite. The Internet protocol suite TCP/IP gets its name since TCP is so important that most applications and protocols are based on it, which is in turn based on Internet Protocol (IP). It compensates for IP's weakness by providing reliable, connection oriented connections that hide most limitations of IP.

2.1.1 TCP introduction

Sources of Packet Loss

In a data network, generally packet loss may happen for two reasons: first, packets discarding in physical channel; second, data corruption, in the case that any bit level error correction code used by physical or link layer cannot recover the whole packet, and it is discarded by receiver, so it is effectively lost. Data corruption can cause a packet to be discarded at the destination. Different channel access and data distribution technologies are subject to different kinds of corruptions:

- Various cable connections can be influenced by low quality wiring and connectors, crosstalk or electro-magnetic interference.
- Wireless networks such as wireless LANs, cellular networks, and satellite networks can be disrupted by radio frequency interference, signal attenuation due to line of sight obstacles, poor weather conditions, multipath fading, antenna pointing, polarization, or alignment errors.
- Optical fibers are vulnerable to physical vibration like temperature variation, or low quality splices and temporary connectors.

Packets might also be discarded by intermediate routers on purpose due to the congestion control algorithms employed by protocols such as TCP. In addition, Buffer overflows caused by unexpected heavy network traffic load can force the intermediate routers to discard packets.

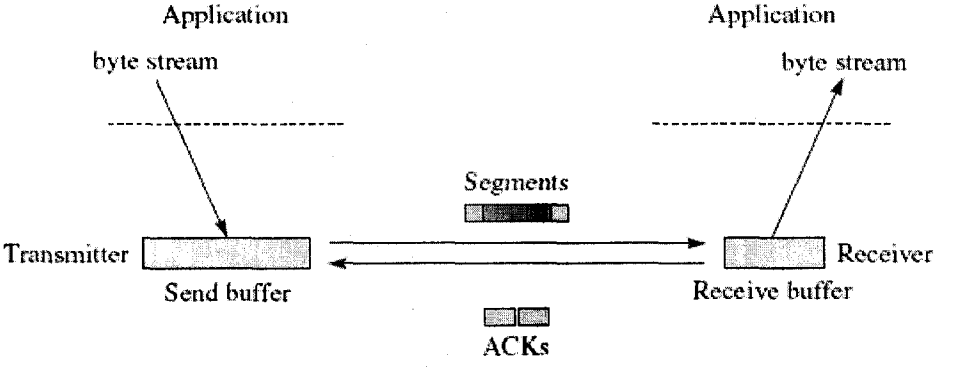

Figure 2.1: TCP preview [12]

TCP is a connection oriented transport protocol that sends data as a stream of bytes. Every TCP packet is assigned a sequence number, and will be acknowledged by the receiver only if it is received successfully and in order, if so, the receiver will send a

corresponding Acknowledgment (ACK) with sequence numbers of the next expected packet. Otherwise, there will be failures. By using sequence numbers and ACKs, TCP make it possible for the sender to know the delivery information. In case of packet loss in channel, TCP can retransmit the packet until either the packet is successfully delivered or until a timeout is reached. With sequence number, TCP can also identify duplicate packets and throw them away. Figure 2.1 shows a simple preview of TCP.

A few key features that set TCP apart from UDP include: flow control, congestion control, retransmission, ordered data transfer - the destination rearranges data according to sequence number, error free and discarding duplicate packets. TCP perfectly supplements the underlying IP service with such functionality as:

- Flow control. TCP controls the traffic speed so the buffers will never overflow. Fast users will lower their transmission rate to keep up with others.
- Reliability. TCP adopts sequence numbers to manage the data that has been transmitted and received, and make sure the data is properly delivered by retransmission in case of a transmission failure.
- Self-adjustability. TCP can dynamically learn the congestion level of the network and adapt its operation to maximize the throughput, thus avoiding either underutilization or overload of the network.
- Streaming. TCP data is organized as a stream of bytes, much like a file. The detailed technique behind the network is hidden from end users.

Round Trip Time Estimation

When a sender transmits a packet, it waits a period of time for an ACK. If it does not receive the ACK within an expected period, the packet is assumed to be lost and is then retransmitted. The problem is how long we should wait, well it depends. For Ethernet, only a few microseconds are enough; if it is wide area Internet, a few seconds might be reasonable during peak hours; in satellite networks, it may take minutes. All the modern implementations of TCP try to solve this problem by observing the regular end to end transmissions and developing a proper estimation of regular round trip duration. This process is called Round Trip Time (RTT) estimation, one of the most important parameters in a TCP exchange. If it is too low, packets are retransmitted unnecessarily; if it is too high, the channel stays idle while the user waits timeout. Both cases are a waste of network resources.

TCP Processes

TCP offers connection-oriented service over packet switched networks, which means that there is a virtual connection between source and destination. In contrast to its traditional counterpart User Datagram Protocol (UDP), where users directly begin to send packets whenever they want, TCP offers connections that have to be pre-established. There are three phases in any virtual connection: connection establishment, data transfer and connection termination, which are explained as follows.

Connection Establishment

Before a user tries to connect with another user, the receiver must first pick a port and make it available for connections, known as a passive open. Once the passive open is set

up, the sender can then create an active open. In order for two users to communicate using TCP they must first establish a connection by exchanging messages in a serial of processes known as the three-way handshake. Figure 2.2 above illustrates the processes of the three-way handshake. In Figure 2.2, it can be seen that there are three TCP

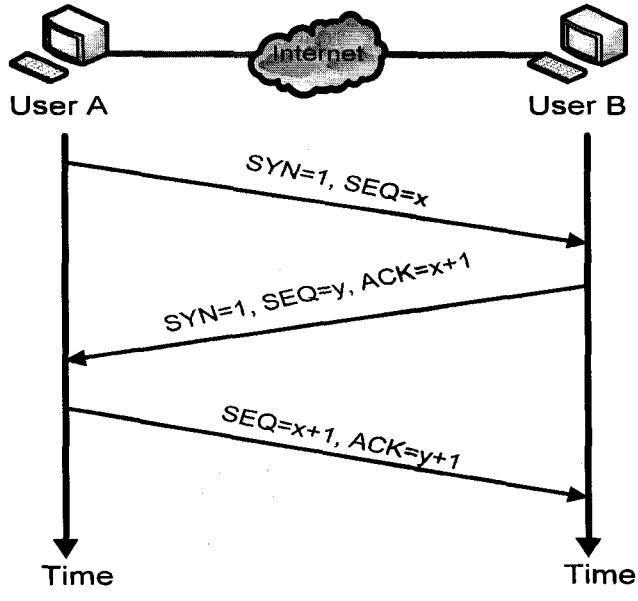

Figure 2.2: TCP Connection establishment

segments exchanged between two users, user A and user B. From top to bottom, the figure shows the events in time sequences.

1. To start, user A initiates the connection by sending a TCP segment with the SYN control bit set and an Initial Sequence Number (ISN) represented as the variable x in the sequence number field.

2. A moment later, user B receives this SYN segment, processes it and replies with a TCP segment of its own. The reply from user B includes the SYN control bit set and its own ISN represented as variable y. User B also sets the ACK control bit to

indicate the next expected byte from user A should have data starting with sequence number x+1.

3. Once user A receives user B's ISN and ACK, it completes the connection establishment phase by sending a final ACK segment to user B. In that case, user A sets the ACK control bit and indicates the next expected byte from user B by placing ACK number y+1 in the ACK field, and also it sends its own data with the sequence number x+1. At this time, both users have received an ACK of the connection and consequently have finished the connection establishment.

In addition to the information shown in the Figure 2.2 above, an exchange of source and destination ports used for this connection are also included in both users' segments.

Data Transfer

Having finished the connection establishment and exchanged the ISNs, users at both ends can exchange data. Without touching much technical details, we only roughly describe a few key ideas here. In a simple TCP implementation, a sender keeps sending data to the receiver given that there is data to send and that the sender does not exceed the transmission window advertised by the receiver. After the receiver accepts and processes TCP segments, it sends back positive ACKs, indicating the location of next data in the byte stream. These ACKs also include the window which shows how many bytes the receiver is presently willing to accept. If some data is lost or duplicated, a gap may be present in the stream, in which case all or part of the packets in the window will be retransmitted, depending on the different sliding window schemes used. Sliding window schemes will be presented in next 'Flow Control' subsection. A receiver will continue to

acknowledge the latest reception in the byte stream. If there is no data to send, TCP will just stay idly by waiting for the application to put data into the byte stream or to receive data from the other end of the connection. If the data queued by the sender exceeds the receiver's advertised window size, the sender must stop transmission and wait for further ACKs before continuing on the transmission.

Connection Termination

Generally, to release a connection, a four-way handshake process is needed to totally shut down a connection. Four steps are required because TCP is a full duplex protocol, which means both ends must shut down independently. The connection termination phase is shown in Figure 2.3 below. Note that instead of SYN control bit fields, the connection termination phase uses the FIN control bit fields to indicate the closing of a connection. To terminate the connection in our example:

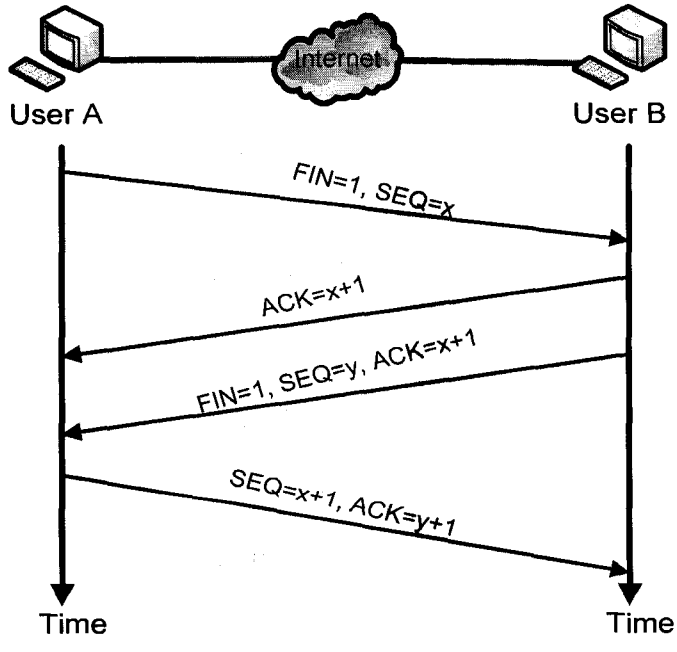

Figure 2.3: TCP Connection Termination

1. User A tells TCP to close the connection. So it sends the first FIN segment and its sequence number x to user B.

2. When user B receives the initial FIN, it immediately acknowledges the segment and reminds user A the next segment should be x+1. Meanwhile it warns its application of the termination request.

3. If the application on user B agrees to shut down the connection, it then sends its own FIN segment, sequence number y, together with the same ACK.

4. Finally, user A will finish the termination and respond with an ACK.

It is also possible to terminate the connection by a 3-way handshake, when user A sends a FIN and user B replies with a FIN & ACK, it is like combining step 2 and 3 together. And then user A replies with an ACK.

2.1.2 TCP Flow Control

TCP realizes flow control through a sliding window protocol. In each TCP packet, the receiver indicates the space it can provide to buffer the data in the receive window field of TCP header. The sender can only send at most that amount of data before it has to wait for an ACK and window update from receiver. If the sender transmits too fast for the receiver, TCP starts flow control to slow down the transfer speed. TCP also reports delivery information to high layer protocols and applications it supports. All these features make TCP a reliable end-to-end transport protocol. Figure 2.4 below is a typical illustration of TCP sliding window mechanism, in which w is the advertised window size indicated by the receiver at the beginning. Suppose the source knows that, based on

ACKs received, Byte *x* is the last data byte received by destination. The source can send data of size up to Byte x + w.

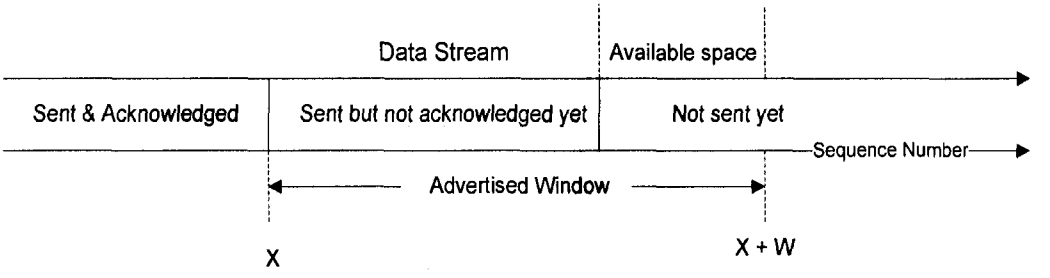

Figure 2.4: Sliding window

Go Back N

Go Back N (GBN) is an example of sliding window protocol where the receive window size is one. In a GBN protocol, the sender is allowed to transmit multiple packets without waiting for an ACK, but there is a limit that the number of unacknowledged packets must be within a certain range. An ACK for a packet is considered to be a cumulative acknowledgement, which means that all packets with a sequence number up to and including this one have been correctly received at receiver. The receiver only accepts in order packet, so if an error occurs when receiving a packet, the receiver will simply discards all the following packets and then the sender will have to retransmit all the following packets in its window, which decreases the performance.

There is a timer for the oldest packet that has already been transmitted but not yet acknowledged. Whenever there is such an unacknowledged packet, the timer is restarted; if there are no outstanding unacknowledged packets, the timer is reset. If a timeout occurs,

the sender retransmits all packets that have been previously sent but have not yet been acknowledged, which also decreases the performance.

Selective Repeat

To avoid unnecessary retransmissions in GBN, in Selective Repeat (SR) scheme, the receiver has a larger window size, so it can store out-of-order but error-free packets. Therefore, it acknowledges any correctly received packet whether or not it is in order. Out-of-order packets are stored until all the lost packets are received, and then the packets can be delivered to higher layer in order. On the other hand, the sender only retransmits the packets that have not been acknowledged. If there is an error, it simply sends a Negative Acknowledgement (NACK) to ask for a retransmission. In particularly, in case of packet loss, the receiver records the sequence number of the earliest lost packet, continues to accept the subsequent packets and replies each with an ACK piggybacking the sequence number of the earliest lost packet. The sender continues to send subsequent packets until it reaches its limit of send window. Once the sender has sent all the packets in its window, it retransmits the packet whose sequence number is given by the ACKs, and then continues from the place it left off.

SR scheme is mainly based on a method called Selective Acknowledgment (SACK), a modification to TCP proposed in RFC 2108, it is an option that allows the receiver to acknowledge discontinuous packets that were received correctly. The use of SACK is optional and it needs support from both ends to work properly, which is negotiated in the Connection Establishment phase. SACK is usually indicated in the optional field of TCP header. TCP employs a form of SR scheme to provide reliable end-to-end data transfer

over communication networks, and the SACK is widely supported in all popular TCP stacks.

2.1.3 TCP congestion control

TCP congestion control [RFC 2581] and Internet traffic management related subject matters generally have been active areas of study and experiment. Current standard TCP implementations usually contain four intertwined algorithms: slow start, congestion avoidance, fast retransmit and fast recovery.

Slow Start

In TCP congestion control, Slow Start (SS) (Figure 2.6) is a scheme used by the source end to control the transmission rate. This is achieved by the return rate of ACKs from the receiver. To be exact, the rate of ACKs returned by the receiver decides the rate at which the sender can transmit data.

At the beginning of a TCP connection, the slow start algorithm initializes a congestion window to one segment, which is the Maximum Segment Size (MSS) set by the receiver during the connection establishment phase. Every time an ACK is received by the sender, the congestion window is then increased by one segment. In fact, as long as the network condition is good, slow start is not very slow because the windows size would increase exponentially in such case. Suppose the first transmission succeeds, it increases the window to two segments; after successful transmission of these two segments, the window size goes to four segments; then eight segments, then sixteen

segments and so forth until it reaches the maximum window size advertised by the receiver or until congestion happens. Figure 2.5 shows the TCP timing during Slow Start.

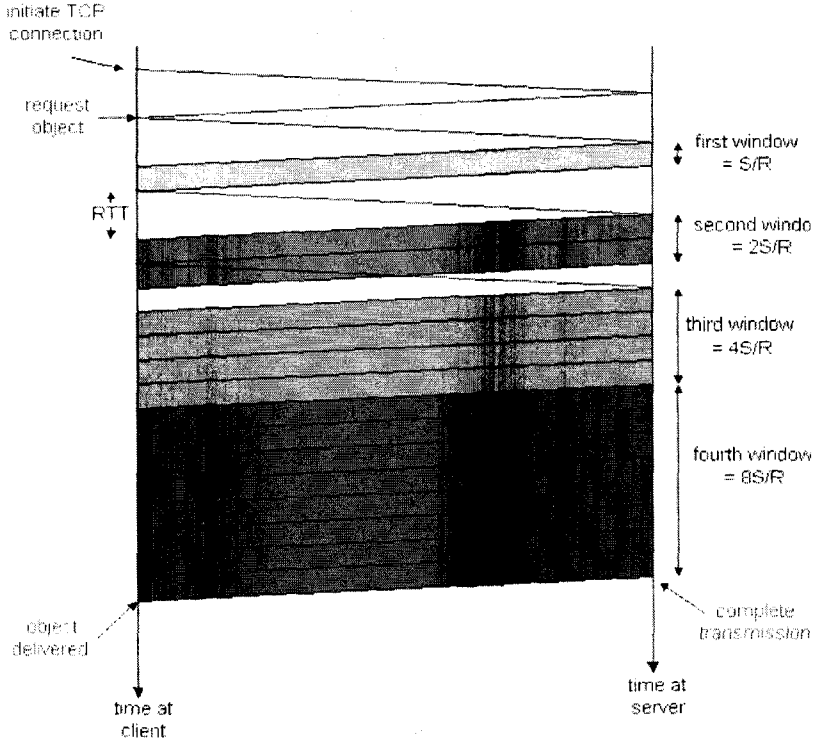

Figure 2.5: TCP timing during Slow Start [13]

However, sometime network quality may vary or the congestion window may be too large for the network such that packets may be dropped. Packet loss will cause timeout at the sender, in which case the sender will enter congestion avoidance process as follows.

Congestion Avoidance

Congestion avoidance (CA) (Figure 2.6) is used to slow the transmission rate when a packet loss happens. Congestion avoidance and slow start are independent algorithms with different objectives. But in practice they are usually used together to better control the data transfer so it does not stay slow.

Basically, congestion avoidance is also thought to be an Additive Increase Multiplicative Decrease (AIMD) algorithm [RFC 2001] because it employs a method of linear growth of the congestion window, combined with an exponential reduction when congestion takes place. In congestion avoidance, network congestion could be indicated by either a timeout or the reception of three duplicate ACKs. If it is caused by three duplicate ACKs, fast recovery algorithm will be activated, where the sender immediately halves its current transmission window, which is so called 'multiplicative decrease'; If congestion was caused by a timeout, the congestion window is reset to one MSS, which means the sender directly goes back to the slow start phase.

Nonetheless, slow start only runs until the halfway to where congestion last took place. After this halfway point, the TCP enters congestion avoidance phase, where the congestion window is increased by one segment every RTT, no matter how many ACKs were received in that RTT, until a packet loss happens. A typical way to do that is for the TCP sender to increase its congestion window by $(MSS/CongWin)*MSS$ bytes for every new ACK, where *CongWin* denotes the current congestion window size. That is so called 'additive increase', a linear growth of congestion window, compared to slow start's exponential growth. This method makes the sender more carefully increase its transmission rate as it comes close to the place where the congestion happened last time.

Fast Retransmit

Fast retransmit is an improvement to TCP which saves the time a sender waits before retransmitting a lost segment. When a duplicate ACK for the same packet is received, TCP does not know whether it is caused by a segment loss or simply that a segment was

delayed and received out of order at the receiver. It is assumed by a TCP sender that, if more than two duplicate ACKs are received by the sender, it is a strong sign that at least one segment has been lost. Because the TCP sender assumes that, enough time has elapsed for the receiver to properly reorder all the segments and send a new ACK, by the fact that the receiver had already sent three duplicate ACKs. Therefore, if a TCP sender receives three duplicate ACKs with the same acknowledge number, that is a total of four ACKs with the same ACK number, the sender can then confidently assume that the segment with the next higher sequence number was lost and will not arrive out of order. The sender will then retransmit the packet that was believed to be lost without waiting for the retransmission timeout to expire. The fast retransmit algorithm first appeared in the 4.3BSD Tahoe release, and it was followed by slow start [RFC 2001].

Fast Recovery

Fast Recovery (Figure 2.6) is an algorithm employed by TCP Reno to improve the performance of TCP Tahoe, especially for large windows, that allows high throughput under moderate congestion. Because duplicate ACKs can only be generated when a segment is received, this is a strong signal that there may not be severe network congestion since at least there is still data flowing through, so TCP does not have to reduce the flow rapidly by resetting the congestion window to 1 MSS. Therefore, after the fast retransmit of the lost segment, the sender resumes its transmission with a larger window, compared to that in slow start it is only one MSS, the sender then enters congestion avoidance phase and linearly increases its window. The fast recovery algorithm appeared in the 4.3BSD Reno release [RFC 2001].

Finally, Figure 2.6 illustrates an outline of two typical TCP congestion control algorithms: TCP Tahoe and TCP Reno, with the periods of exponential increase, additive increase and multiplicative decrease. Each scenario shows a reaction of sender to

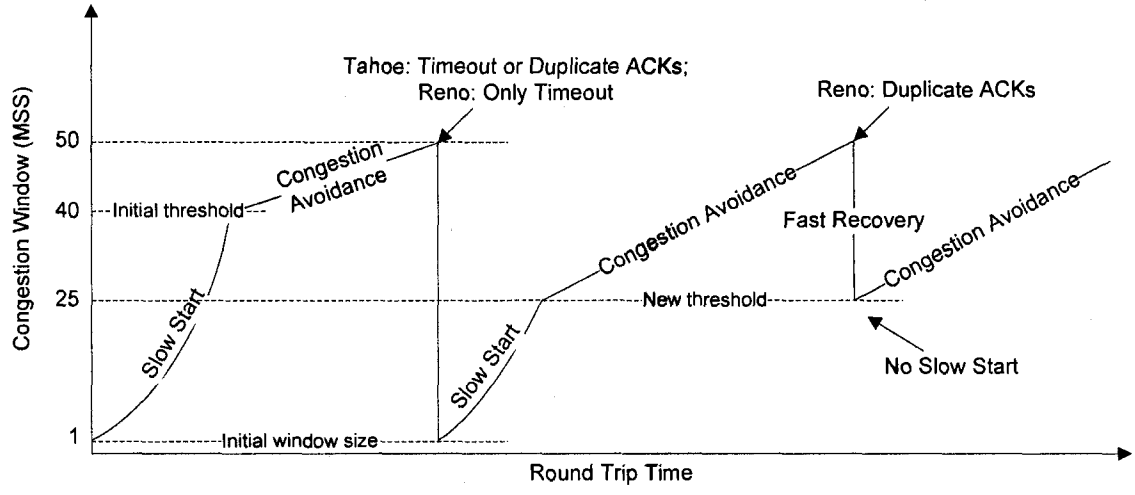

Figure 2.6: TCP congestion control in Tahoe and Reno

different network conditions. From Figure 2.6 one can clearly understand the difference between the two TCP versions: Tahoe unconditionally resets congestion window to the initial value of 1 MSS when a loss event occurs, no matter it is caused by a Timeout or three Duplicate ACKs; while Reno does that only if it is a Timeout, and for the case of Duplicate ACKs, it uses the Fast Recovery algorithm to maintain high speed.

TCP is a complicated protocol that deals with all kinds of mechanisms for data communication in a packet switched network such as the Internet. With the increasing development of Internet, our dependence on TCP keeps increasing. Supporting the reliable delivery of data on a packet switched network is not an easy job. Even after so many years of standardization, the researches have never been stopped, it is still an area of great activity and there remain many problems to be solved. For example, fairness is

now a topic of great concern. Its objective is to realize 'fair play' for all network users, and to limit the greedy TCP senders so as to make room for users with a low bandwidth connection. Current congestion control techniques have been working wonderfully so far, it only remains to be seen how far they can carry on the Internet in various transmission circumstances such as in wireless environment. TCP in wireless will be the main subject of next section.

2.2 Problems and enhancements in wireless

Since TCP has been primarily designed and developed for wired networks, any packet loss is considered to be caused by congestion and then leads to huge precautionary decrease of the congestion window size. Nevertheless, as we know wireless channels experience random packet losses caused by various effects such as multipath fading, hand off, shadowing and other radio effects that should not be considered congestion. In case of packets loss due to wireless channel quality, the congestion window size will be reset, and then there will be a congestion avoidance phase, leading to considerable underutilization of channel resources.

A lot of studies show that the standard TCP performs inefficiently in a wireless environment due to its inability to distinguish packet losses caused by congestion from those caused by errors. Unfortunately, today most wireless data applications such as World Wide Web (WWW), File Transfer Protocol (FTP), multicasting and Telnet, use TCP as the standard transport layer protocol since they need reliable data delivery. Lots of research has been done to conquer this problem and to improve the network performance over TCP based wireless networks.

2.2.1 TCP problems in wireless

High Bit Error Rate

The conventional congestion control methods for TCP have been generally designed for wired networks, where packet loss caused by bit corruption and link errors is almost negligible, in other words, almost all packet losses are caused by congestion. However, the BER in wireless networks is much higher than that in traditional wired networks, comparing the BER of wired networks on the order of 10^{-9} to 10^{-6} to that of wireless networks on the order of 10^{-4} to 10^{-1}. For example, it is common to have a BER of 2% or so for Code Division Multiple Access (CDMA) based wireless networks. Therefore, in wireless network, packet loss does not necessarily mean congestion because it may be lost due to interference, signal fading and so on. But TCP incorrectly considers it as congestion loss and launches congestion control process, where it keeps the sending rate relatively low, leading to a degraded performance.

Burst loss

A burst loss may be caused by signal fading. Channel interferences can lead to correlated packet losses, usually a burst loss of many consecutive packets within a very short duration. In wireless networks, since the connection is unreliable, wireless users usually suffers from unpredictable disconnection when they move about or when there is a power failure. Transmission in this period experiences huge packet losses, causing unnecessary suppression on transmission rate by TCP and then degraded performance. Burst loss can cause several consecutive timer expirations and retransmissions of the same packet within a short period. The value of retransmission timer is doubled every time a failure

occurs until it reaches some threshold. Hence, several consecutive retransmission failures can lead to very long idleness of the connection even though the network condition resumes immediately.

In infrastructured networks, all traffic is routed via access point (AP) or base station (BS). Such as in cellular systems, when wireless users move out from the coverage area of the current BS, they have to register at another BS which they are moving into. As a result, a handoff takes place and all the following traffic is then routed via the new BS. Even though it usually takes only a few seconds, many packets being transmitted may be lost because they are transferred to the old BS when a handoff is going on; a wireless user may also lose the connection to BS, and any data transmitted to or from it will be lost, both leading to a burst loss. The handoff frequency depends on the mobility of some particular users and the coverage area of BS.

In ad hoc networks, where users move from one place to another, the network connectivity varies, so the network topology may change frequently. Therefore, the original routing may not work anymore, and it takes time to recalculate a new routing plan. As a result, some packets using the original routing may be lost during this process, so a burst loss takes place. The cause of a burst loss in ad hoc networks is affected by the coverage area and mobility of every user in the network, compared to infrastructured networks where it is only some particular users and the coverage area of BS that matter.

Unpredictable delay

A highly variable RTT may also cause high RTO (Retransmission Time Out), which is based on both estimates and variance of RTT. Since wireless user move around randomly,

their distances from a BS change all the time, causing temporally varying delay. This unpredictable delay is not easy for TCP to adjust to. TCP reacts slowly to data loss when RTO is high. Variations in the RTT may be caused by link layer retransmissions of wireless channel. If the link layer frames that contain a TCP packet must be retransmitted due to bad channel quality, the packet is delayed. RTT variations can also be caused by channel access method, handover and queuing in routers, BSs and other intermediate nodes. A long RTT causes low network efficiency and underutilization, as it takes several RTTs for TCP to gradually achieve the network capacity.

Packet disorder

Packet disorder means the case that some packets are received out of order. This is found out to be not unusual so that most fast retransmissions in fact are unnecessary, leading to significant performance degradation. In infrastructured networks, handoff may cause disordered packets. During handoff, packets travel through different paths may take different time to get to the destination. For an ad hoc network, there is no fixed infrastructure and every user can act as a router. Therefore, dynamic routing causes packets of the same source-destination to be forwarded through different paths and then to be received out of order. Besides, in some proposed link layer error correction techniques, retransmission is performed regardless of the semantics of the underlying transport protocol, which can also lead to disordered packets.

As a result, TCP is not suitable for many applications. Since the application will be stuck after a lost packet until the retransmission is successfully finished. This is a problem for real time applications such as streaming multimedia, multiplayer on-line

games and Voice over IP (VoIP) where it is usually more helpful to get most of the data in time than to get all of the data in order.

Limited power

Wireless equipments usually use battery as power source, so unlike electrically powered equipments, they cannot afford too many retransmissions. In other words, TCP is not designed as an energy-efficient protocol.

Small bandwidth

Compare to wired networks, wireless networks have a much smaller bandwidth. For example, the Ethernet can now achieve a bandwidth of up to 1 Gbps, while the current IEEE 802.11g has a bandwidth of only 54 Mbps. Therefore, it is a major problem for TCP to efficiently utilize the wireless bandwidth in some cases such as real time applications.

TCP packets may be lost due to unreliable link layer protocol. After trying retransmission for a few times, link layer protocol gives up and leaves further responsibility of error recovery to higher layer. TCP may also mistake a sudden increase of RTT, which is common in wireless networks, as a data loss. If the delay is long enough for retransmission timer to expire before an ACK is received, TCP will mistake the delay as a signal of data loss due to congestion. Therefore TCP will unnecessarily retransmit the data that it thinks to be lost and then goes to slow start.

2.2.2 TCP enhancements in wireless

The promising wireless applications, especially high speed multimedia services and the emergence of wireless IP communications running over the Internet, demand effective enhancements or modifications of TCP for better performance. The standard Reno [RFC 2581] halves the window size when experiencing a packet loss whatever the reason was. If it is due to network congestion, this improves network congestion. On the other hand, it would degrade the performance for random loss. The features of wireless networks vary with access technologies, so a universal solution for all kinds of wireless networks is unpractical. Each wireless TCP solution tries to take care of some specific problems.

TCP has been continually evolving ever since its first specification RFC 675 in 1974. Even though many developments have been made over the years, the primary foundation behind TCP remains almost the same. TCP congestion control, specified in RFC 2581, is one of the most important TCP improvements in last decade; it introduces efficient algorithms to avoid unnecessary congestion. After that, a signaling scheme called Explicit Congestion Notification (ECN) was proposed for congestion avoidance. There are lots of implementations of TCP, some of which are thought to be standard TCP implementations. The original TCP congestion avoidance algorithm was known as TCP Tahoe, including the very basic congestion control scheme, namely the slow start and congestion avoidance algorithms described before. However a lot of other algorithms have been proposed since then, such as Reno, New Reno, Veno, Westwood, BIC and so on. TCP Reno adds the fast retransmit and fast recovery algorithms to TCP Tahoe. Furthermore, TCP New Reno improves upon TCP Reno by changing some thresholds in fast recovery algorithm and avoiding the occurrence of multiple retransmissions after

timeout [RFC 2582]. TCP New Reno is for now the most widely used TCP congestion control scheme in practice, while most others competing proposals still need further investigation.

TCP New Reno

TCP New Reno [RFC3782] is a small modification over TCP Reno. It can discover multiple packet losses, so it is much more efficient than Reno in case of multiple packet losses. Like Reno, New Reno also performs fast retransmit when it receives multiple duplicate packets, yet it differs from Reno in that it does not quit fast recovery until all the unacknowledged packets at the time it started fast recovery are successfully received. Therefore it avoids reducing the congestion window many times as in Reno. However, New Reno suffers from the fact that it takes one RTT to identify each packet loss. Only when the ACK for the first retransmitted packet is received, can it realize which other packet was lost. Even though, New Reno works much better than Reno at high BER.

ATCP

In ad hoc networks, it is usual to have a high BER and the route changes a lot so network topology changes, leading to more packet loss in addition to network congestion. ATCP [14] is a cross layer method proposed to provide end-to-end solution to improve TCP throughput for mobile ad hoc networks. It works between the standard TCP and IP layers, and it is based on ECN message to detect network congestion and distinguish congestion loss from error loss, and uses the ICMP 'Destination Unreachable' message to detect the routing and topology status of the network. According to the feedback messages, ATCP sets TCP sender to an appropriate state: persist, retransmit or congestion control. ATCP

also reorders the packets so that TCP would not generate duplicate ACKs. ATCP deals with high BER, route failure, network congestion, and packet reorder, which makes it more suitable for TCP in mobile ad hoc networks. In addition, ATCP does not generate or regenerate ACK packets nor modify the TCP, so it maintains the standard end-to-end TCP semantic. However, it is not always possible to have a stable node to provide feedback messages in ad hoc networks, so sometime ATCP might be unreliable.

Link Layer proposals

This approach tries to hide wireless packet losses from higher layers by using link layer level retransmissions instead of end-to-end retransmissions, so the packet loss is localized and the probability of packet loss due to wireless channel is decreased. These proposals employ intermediate routers to store all unacknowledged packets and retransmit them whenever a packet loss is detected. As the propagation delay of radio signal is much smaller than end-to-end delay, so this approach knows immediately about the packet loss and can then react faster than higher layers. All the proposals maintain the end-to-end TCP semantics.

However, TCP has its own end-to-end retransmission mechanisms, and it has been shown that independent retransmission protocols can decrease the performance, especially under high BER. In addition, with the network security being more important, encryption is broadly adopted, therefore if the data is encrypted, this approach may not work. This kind of proposals includes Snoop [15], SNACK-New Snoop (SNACK-NS) [16], Delayed Duplicate Acknowledgments (DDA) [17] and so on.

TCP Vegas

TCP Vegas [18] detects congestion based on the increasing RTT values of the packets in the connection, in contrast to other versions of TCP, which do so only after the congestion has actually happened. In TCP Vegas, timeouts were set and round-trip delays were measured for every packet in the transmit buffer. The RTT of the connection and the window size are used to compute the number of packets in the network buffers. Vegas takes delay as a signal of congestion and then reduces its throughput, and uses additive increases and additive decreases in the congestion window. Vegas decreases the window size when it exceeds some threshold and increases it when it is below certain threshold. It tries to stabilize the network congestion state around the optimal point by proactively adjusting the congestion window without significant change in the congestion window. However, Vegas detects congestion based on RTT measurements, which may inaccurately reflect congestion level on forward path.

TCP Veno

TCP Veno [19], as its name says, is a combination of Vegas and Reno. It uses the same method as Vegas to estimate the accumulated packets in the network, but it proposes an approach to distinguish the causes of packet loss by a threshold. If the number of accumulated packets is less than the threshold, the packet loss is random, where Veno increases the congestion window in a conservative way, that is to send only one packet for every other ACK received; otherwise the loss is caused by congestion, where it performs the standard TCP Reno. However, Veno works poorly under high BER, and it does not deal with disconnection. Figure 2.7 shows a result from [19].

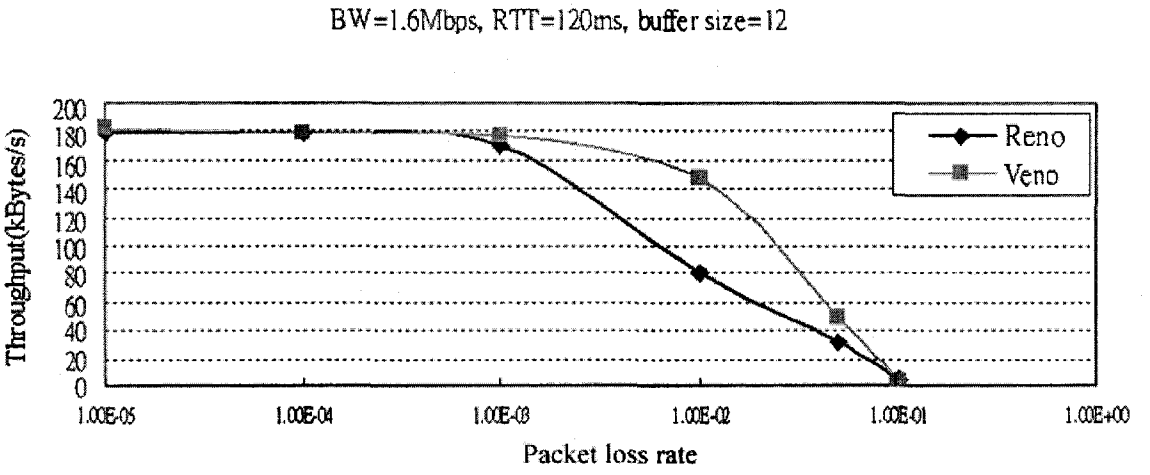

Figure 2.7: Throughput versus packet loss rate for Reno and Veno [19]

TCP Westwood & Westwood+

TCP Westwood [20], which can be considered as an extension of TCP Reno, is a sender-side only modification of the TCP congestion window algorithm. It adjusts the size of the congestion window based on the rate of the ACKs. Westwood sets up a mechanism to measure bandwidth at the sender side, based on the interval of returning ACKs. It evaluates available bandwidth to control the sending rate. When it receives three duplicate ACKs, it sets the slow start threshold to reflect its estimated bandwidth-delay product. This method preserves the end-to-end TCP semantics, and needs slight modification at end users and in some case the routers.

Westwood+ is a development of Westwood, whose bandwidth estimation algorithm was soon discovered worked badly with reverse traffic due to ACK compression. Westwood+ improves the accuracy of the estimation of the available bandwidth and it is implemented in the kernel of Linux. Figure 2.8 shows a result from [20].

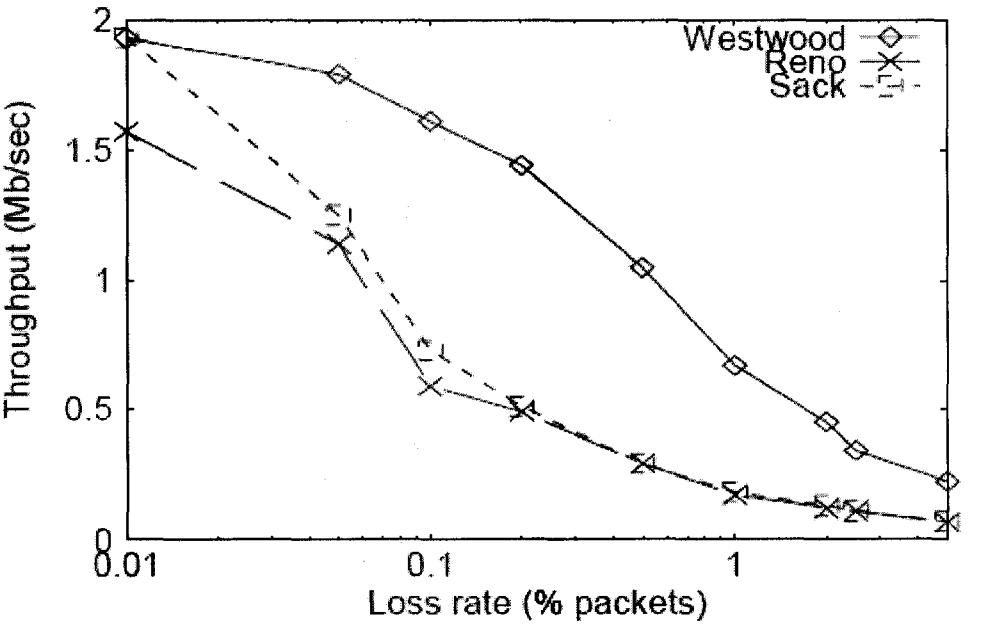

Figure 2.8: Throughput vs. error rate of the wireless link [20]

However, large packet loss does not necessarily mean network congestion, which is especially true for wireless networks since wireless links are known to have high error rate. This scheme will have poor performance when the sender mistakenly estimates the bandwidth due to the random delay in wireless network, or the random packet loss rate suddenly increases. In addition, it still needs to modify the standard TCP at the sender side.

Numerous methods have been proposed to improve TCP performance over wireless networks, the proposals presented above are just a small part of them. However, due to the basic instinct of TCP protocol, there are always some limitations for these methods. Each of them only deals with one or a few problems that TCP experiences in wireless environment, there is no almighty method for now. Performance improvements are always important, but in many cases with the cost of violating the end-to-end semantics of TCP is not always comfortable.

In general, UDP is used as an alternative where TCP does not work well. It offers the application multiplexing and checksums like TCP, but does not deal with building streams or retransmission, making it possible for the application developer to program it in a way more suitable for a certain situation or to substitute them with other methods like Forward Error Correction (FEC), which will be discussed specifically in the next chapter.

Chapter 3

A new hybrid ARQ technique

In data communication networks, data could be corrupted and data packets could be lost in the process of transmission. There are two fundamental error correction techniques treating this problem: Forward Error Correction (FEC) and Backward Error Correction (BEC). Although, in some point, BEC is efficient and easy, but it needs feed-back channels and it is inappropriate for situations in which many clients have to wait for a retransmission, such as wireless network.

Forward Error Correction (FEC) and Automatic Repeat reQuest (ARQ) have been the two basic loss recovery techniques generally used for data transfer in communication networks. Both FEC and ARQ use Error Detection (ED) techniques such as Cyclic Redundancy Check (CRC) to detect errors and erasures, but FEC adds more redundant data to the source information to enable correction. According to coding theory, an error is defined as a corrupted symbol in an unknown position, while an erasure is a corrupted symbol in a known position. In case an error or erasure is detected, FEC may be able to correct it, while ARQ just simply asks for a retransmission. The advantage of FEC is that no retransmission is needed, so feed-back channels are not necessary. FEC is therefore applied in situations where retransmissions are comparatively impractical or costly. Generally, in data communication networks, ARQ schemes are preferred over FEC for error control, when there is a high-quality channel, such as wired networks. However,

FEC is better when feed-back channels are not available or retransmission is not suitable for some situations like there is a long Round Trip Time (RRT) or a very competitive multiple media access environment, such as wireless networks.

A hybrid ARQ (HARQ) scheme combines the advantages of FEC and ARQ, and offers much better throughput performance and reliability, especially for wireless networks. There are mainly two kinds of HARQ approaches so far, namely type I and type II. Basically, for type I HARQ, it acts like standard FEC unless the channel quality is poor, and not all transmission errors can be corrected, the receiver will detect this situation using the ED code, then it discards the received coded data and a retransmission will be requested, similar to ARQ; while type II HARQ is more dynamic and complicated, it transmits redundant data only if there is a transmission failure, in which case the sender launches FEC process, redundancy is transmitted instead of repeating the same packet and the previously received packet will also be used to improve the decoding capability [21]-[24]. HARQ technology has also been considered in the link layer function of IEEE 802.16 (WiMax) [25] and the third generation of mobile networks (3G).

In this chapter, we introduce a new HARQ technique for reliable and efficient packets transfer in wireless environment. In contrast to most HARQ techniques proposed so far, which usually employ a byte level FEC combined with ARQ, in our system, we mostly use packet level FEC for the data transfer, in conjunction with ARQ to compensate for the little inefficiency. It is similar to type I HARQ, except the FEC is applied in higher layer, the application layer. The first section provides some reviews of FEC; the second section gives a brief description of digital fountain and raptor codes, state-of-the-art

concepts and the most advanced FEC techniques; detailed system implementation and simulation results are presented in the last section.

3.1 Forward Error Correction

Forward Error Correction (FEC), also known as Error Correction Code (ECC), is a technique generally used to deal with errors and erasures in real time communication networks. FEC techniques allow a receiver to correct errors or erasures without further communication with the sender. The error correction is "forward" in the sense that no feedback from the receiver or further transmission by the sender is required. FEC is

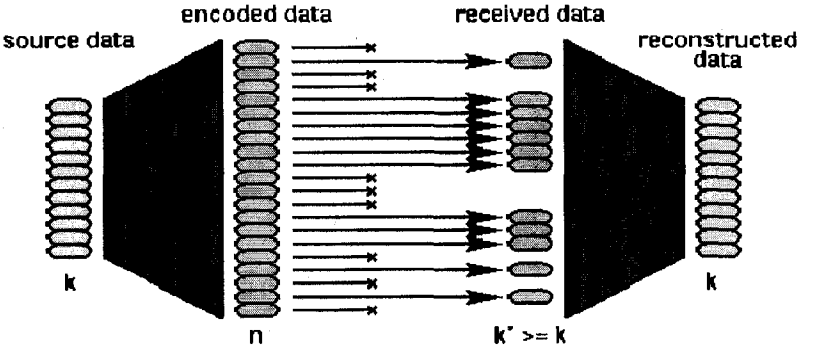

Figure 3.1: Forward Error Correction

realized by attaching redundancy to the transmitted source information using a predesigned algorithm such as Raptor or Reed Solomon, to realize the errors and correct them. Every redundant bit is often a complex function of lots of source information bits. The source information may or may not be included in the encoded codes, so FEC codes can be divided into two subcategories: Codes that contain the source data after encoding are called systematic codes, and those that do not are nonsystematic codes [26].

3.1.1 FEC principles and applications

In a communication system that employs FEC, the sender sends the source data to the encoder. The encoder adds redundant bits to source data to get a longer sequence of encoded data, called a codeword. Then the codeword can be transmitted to the receiver, which uses a corresponding decoder to obtain the source data. Codewords with a large number of redundant bits reduce the information carried by each individual code bit, which is helpful because it reduces the probability that all of the source information will be lost in a single transmission. However, more redundant bits may require more transmission bandwidth and cause packets delivery delay.

FEC can be classified as error detection codes, error correction codes, and erasure correction codes:

- Error Detection (ED) codes only check whether or not the received data is correct, but do not give the methods to locate and correct the errors. It is very important to maintain data integrity across lossy channels and undependable storage media, so it is widely implemented in all kinds of ARQ schemes to perform Error Detection function. Some popular ED codes are: parity check, checksum and CRC. For instance, a 1's complement checksum ED code is used in IP data packets to help receiver check the integrity of IP header, and in TCP and UDP data packets to check the integrity of header and payload data.

- Error Correction codes not only detect an error but also correct it. For instance, block codes or convolutional codes are widely used in various kinds of data storage systems and the physical layers of IEEE 802.11a/b/g wireless LANs.

- Erasure Correction codes can correct a certain amount of lost data where the positions of the lost data are already known. For instance, raptor codes can recover lost packets because they were not received, or errors were detected and the packets were discarded.

Bit Error and Packet Loss

In digital communication networks, transmitted data may be distorted by the noisy channel. At the receiver, each bit is compared with a threshold. If the instantaneous amplitude of the noise is high enough at the sampling instant, the bit may be mistakenly received, causing a bit error. Bit Error Rate (BER) is the ratio of the number of bit errors to the total number of bits sent in a particular time.

In some unreliable networks, such as IP based networks, there is no guarantee that transmitted data will get to the destination. Real time data like encoded video has to be decoded and provided to users at a constant speed. If a packet is delayed too long, it is considered as a packet loss. The ratio of the number of packet loss to the total number of transmitted packets is called Packet Loss Rate (PLR).

The ratio n/k is usually referred to as the stretch factor of an erasure code. The stretch factor represents the proportion of redundancy, where k is the number of only source bits and n is the number of total bits per packet, including redundant bits. The extra redundancy added by FEC means that more than just the original data is transmitted, calling for either a longer transmission time, if the data rate remains the same, or a faster data rate thus higher bandwidth, if the transmission time remains the same. However, what is interesting is, the extra redundancy can at last save transmission time and

bandwidth when compared to the retransmissions that would be required if there is no FEC. The primary trade-offs in FEC are the degree of error protection provided by a specific algorithm, including the level of latency, encoding and decoding processing, and bandwidth extension or extra overhead required to improve resistance to error or loss. Because of the complexity of FEC codes, when designing a reliable communication systems using FEC, we must pay attention to the processing capability of both of the source and destination end.

FEC applications

FEC is the crucial element in an incredible range of applications. For example, latency-sensitive applications like video conferences and applications where the transmitter forgets the data right after it is sent, such as most television cameras, as in case of an error, the source data is no longer accessible, so FEC is used in data storage systems like RAID (Redundant Array of Inexpensive Disks) and distributed data store.

In conventional packet switching computer networks where full duplex communication is available, error correction is performed by using an ARQ protocol such as TCP. TCP makes sure that any data which does not get to the final destination is retransmitted right away. For real time applications like video, the encoder needs to process data stream, so a steady end-to-end delay is required to keep this continuous stream of data. In networks with a high bandwidth-delay product, retransmissions of packets cause big variation in end-to-end delay, this requires a large buffer at the receiver to compensate for the delay effect jitter. However, large buffers not only are pricey but also introduce extra delay, which may be a problem for sensitive real time

communication like video conferencing. In satellite communication, retransmission is also inappropriate since there is little capacity of feed back channel or no feed back channel at all. So retransmission mechanism adopted by ARQ has drawbacks such as high cost and high delay. In contrast to TCP, UDP adds no loss recovery, reliability and flow control features to the lower IP layer. Thanks to its simplicity, UDP headers are shorter and use less network resources. UDP is more practical in conditions where the reliability is unnecessary, or error and flow control can be further supplemented by a higher layer protocol.

FEC also plays an important role in cellular networks. The 3G cellular network is gradually dominating personal communication today. Different cellular networks have different channel capability, RRT, frequency allocation and transport protocol configuration. Cellular networks are much more affected by data corruption owing to various environmental conditions such as the whether conditions and the interference. 3G cellular networks provide multiple services and support real-time multimedia services. The environments and diversity of wireless situations may significantly have an effect on end user performance, so most cellular networks employ FEC techniques in their physical layers and there are increasing applications in higher layer as well.

FEC stands for the most efficient and economical way of improving the reliability of data transmission or storage. As bandwidth efficiency and spectrum management are drawing increasing attentions, it is getting even more significant to get the most out of channel capacity without giving up transmission reliability.

3.1.2 The two fundamental FEC codes

There are basically two fundamental FEC coding techniques: block coding and convolutional coding. We briefly review them as follows, without great technical details.

Block coding

Block coding used to be the first type of channel coding used in earlier mobile communication systems, in which the encoder spreads parity bits into the source data sequence with a specific algebraic algorithm, so it can also be called algebraic code. On the other end, the decoder uses an inverse of the algebraic algorithm to recognize and correct any errors or erasures caused by the poor link quality. In contrast to source coding methods like Huffman coding, and channel coding techniques such as convolutional encoding, the key feature of block code is that it is a fixed length channel code [27]-[32].

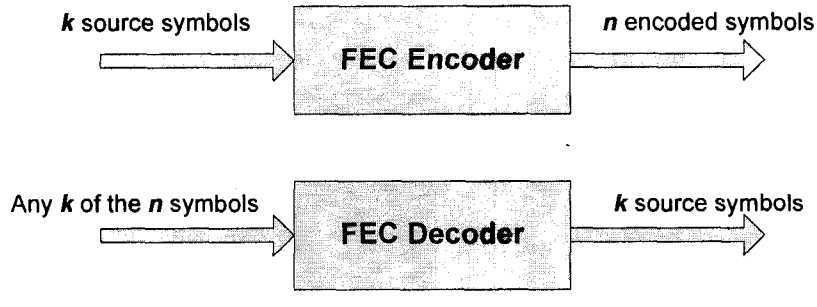

Figure 3.2: FEC Block Coding

As shown on the above figure 3.2, an (n, k) block erasure code converts k source symbols into a set of n coded symbols, such that any k of the n encoded symbols can be used to reconstruct the source symbols. Generally, the first k symbols in each set are the same to the original k source symbols; the remaining $n–k$ symbols are the parity. Usually, FEC codes are able to correct both errors and erasures in a block of n symbols. For the

transmission of streamed multimedia packets, loss detection is carried out based on the sequence numbers in Real time Transport Protocol (RTP) packets, in such case it is known as erasure codes.

Convolutional coding

Convolutional coding, first introduced in 1955, operates the input bits in streams rather than in blocks. Its most important characteristic is that the encoding of any bit is seriously affected by the bits that have preceded it, that is to say, the memory of previous bits. The decoder takes into consideration the memory when attempting to guess the most likely sequence of data that generated the received sequence of code bits. The first type of

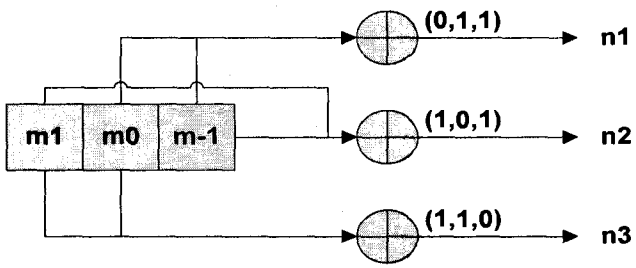

Figure 3.3: A rate 1/3 non-recursive, non-systematic convolutional encoder with constraint length 3.

convolutional decoding in history was sequential decoding, which used a systematic process to look for a nice estimation of the source data sequence; but such processes require a lot of memory, and usually lead to buffer overflow [33]-[37].

The above figure 3.3 shows a rate 1/3 (m/n) convolutional encoder with constraint length (k) of 3. Generator polynomials are G1 = (0, 1, 1), G2 = (1, 0, 1), and G3 = (1, 1,

0). Therefore, output bits are calculated (modulo 2) as follows: n1 = m0 + m-1, n2 = m1 + m-1, n3 = m1 + m0.

Today with the tremendous development of FEC technique, LDPC and Reed Solomon codes are the most popular FEC codes proposed in the literature and are often recommended in Internet Engineering Task Force (IETF) RTP profiles. The following is a very simple review of the two most widely used erasure codes.

Low Density Parity Check Codes

Low Density Parity Check (LDPC) codes, invented in the early 60's and had been forgotten for about 30 years, was the first code achieving a data transmission rates close to the theoretical limit, i.e. the Shannon Limit. Many recent research works have considered adopting LDPC code. LDPC codes are defined by a sparse parity check matrix. The sparse matrix is usually randomly generated, subject to the sparsity constraints. These codes have two major benefits: first, the simplicity of XOR operations makes high speed encoding and decoding possible, which is perfect for handheld devices; second, LDPC handles very large source blocks. LDPC codes are thought to become a standard in the developing market for highly efficient data transmission systems, such as cellular networks and interplanetary communication [38].

Reed Solomon Codes

Reed Solomon (RS) codes are a special class of linear non-binary block codes with the ability to correct both errors and erasures, and they have been used for almost half a century. An RS code provides perfect error protection against packet loss given that it is a Maximum Distance Separable (MDS) code, that is to say, no other coding scheme can

recover the lost source symbols with fewer received code symbols. According to a comparison carried out by the IETF, RS codes are more suitable for small block size and real time streams, while LDPC codes work better for large blocks over unidirectional channels. In addition to block size constraint, RS codes also suffer from computational complexity [39].

Even though with the incredible development in recent years, some FEC codes are more than ever approaching the Shannon Limit, RS codes are still playing a very important rule now, especially for high rate systems with relatively small data packets. RS codes have outstanding burst correction capability, so they are broadly used in various commercial applications, the ability to correct both random and burst errors is ideal for applications such as magnetic tape and disk storage like CDs, DVDs and Blu-ray Discs, where the defects in storage media sometimes may cause burst errors. RS codes are also deployed in data transmission technologies like DSL and WiMAX, and broadcast systems like Digital Video Broadcasting (DVB), so RS codes are perhaps the most widely used code.

3.1.3 Packet level FEC

Erasure channel

In an erasure channel, a packet is either lost in the channel, or is perfectly transmitted without any corruption. The receiver either gets a packet error free, or loses it in total. The Internet is a typical example of erasure channel, where files are transmitted in packets, each of which is either lost or perfectly received.

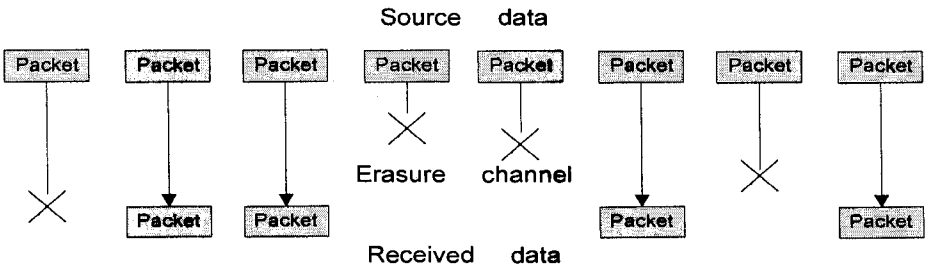

Figure 3.4: Erasure channel

Packet level FEC is a packet loss recovery technique that can recover a certain amount of packets after erasure channel. Therefore, no further retransmission is needed in systems equipped with packet level FEC, which makes it useful in networks running real time applications like video and audio streaming. Packet level FEC techniques are generally based on the use of error detection and erasure correction codes, which can detect and recover both erasures and bit level errors [40].

Existing packet level FEC methods are mostly based on simple parity check codes or Reed Solomon codes with erasure decoding, but in this paper we will use the most advanced FEC codes called raptor codes. In the standard packet level FEC, k information packets are grouped into a block and then attached with r parity packets to create a coded block with $k + r$ packets. The parity packets are constructed in a similar way as the parity bits in the linear block codes except that bits in the encoding process are from different packets. If a packet is lost during transmission, the receiver knows the position of the lost packet according to the packet sequence numbers. Even though most wireless networks use integrated physical layer adaptive coding and modulation schemes, packet level FEC protocols are usually necessary at application level. Wireless communication suffers from both short term fast fading and white Gaussian noise, which is solved by the integrated

physical layer coding; long term slow fading when entering a tunnel, which is solved by packet level FEC coding.

Packet level versus byte level

FEC can be implemented at many levels from byte level up to packet level. In byte level FEC, a symbol is a byte; while in packet level FEC, a symbol is a packet. Byte level FEC is implemented at the physical layer of almost all wireless networks. Packet level FEC is generally based on erasure coding, it has the advantages such as: byte level FEC is unable to recover an entirely lost or delayed packet; a single parity packet can be used to correct different single packet losses in a group of packets; in byte level FEC, a corrupted packet is already detected and discarded at the link layer by CRC, or at the transport layer by checksum, and in IP based network, the network layer will detect corrupted packets due to erroneous bits and then discard them, so they will not be available at application level.

3.2 Digital Fountain and Raptor codes

Fountain codes, first introduced by Michael Luby [41], are sparse-graph codes for erasure channels such as the Internet. Standard file transfer protocols simply divide a file into some packets, and then repetitively transmit each packet until it is successfully received. There has to be a feed-back channel for the transmitter to know which packets need retransmission. In contrast, fountain codes generate potentially limitless packets based on some random functions of the entire file, and the transmitter keeps sending packets to the receiver without any acknowledgement. Fountain codes are known to have efficient encoding and decoding algorithms thanks to the small computational costs, and are able to recover the original k source packets from any k' of the encoding packets with high

probability, regardless of the order, where k' is just slightly greater than k. A fountain code can be considered as an optimal code if the original k source packets can be recovered from any k encoded packets.

Fountain codes were firstly designed for erasure channels, but their performance on other channels such as noisy channels and fading channels has since been studied and proved to be excellent [42]-[44]. They have also been proved in broadcast and multicast in wireless networks [45] and wired local Ethernet networks.

3.2.1 The Fountain concept and its applications

The Digital Fountain

A digital fountain is an abstraction of erasure coding for network communication. Imagine you are in front of a water fountain spouting an unlimited stream of water drops, any of which can be used to fill your glass, you do not care which drops of water fall in as long as you get enough water. Similarly, in communication networks with a digital fountain, a user receives encoded packets from one or more servers, once enough encoded packets are received, the receiver can rebuild the source data, and which packets are used does not matter.

Digital fountains in effect change the traditional model of communication, where a user receives an ordered sequence of packets to get the original data. Without that restriction, digital fountains truly improve the efficiency and simplify the data delivery, especially when the data is large or is to be sent to many users, making them suitable to more kinds of networks than early techniques. Fountain codes can also be called rateless codes in the sense that the number of encoded packets that can be generated from the

source is potentially infinite. Regardless of the channel quality, we can send as many encoded packets as needed in order for the decoder to recover the source data, there is no fixed code rate.

Developments of the Digital Fountain

As we know in RS codes [39], a data of k symbols can be recovered with any k received encoding symbols, so theoretically RS codes can be used to build a practical materialization of a digital fountain. Nonetheless, there are quite a few difficulties in practice. As an alternative, many implementations of digital fountains are derived from variations of LDPC codes [38]. For example, Tornado codes, a type of LDPC codes designed for erasure channels [46], was a big step forward towards the development of fountain codes. The problem is that the number of encoded packets that will be generated must be decided in advance, since the encoding is based on the graph that corresponds to the Tornado code.

LT codes

LT (Luby Transform) codes [47], near optimal erasure codes invented by Michael Luby, are the first full realization and practical implementation of the 'digital fountain' concept. LT codes distinctively adopt a simple algorithm based on exclusive-or operations for encoding and decoding. Similar to some other fountain codes, LT codes depend on sparse bigraphs to achieve high performance in the cost of a little redundancy overhead.

LT encoding

Suppose the file to be transferred consists of k packets, the LT encoding process for a source file is simply as follows:

1. Randomly choose the degree d of the packet from a degree distribution, which is the

key of LT coding and is carefully designed according to the source file size k.

2. Choose uniformly at random d distinct input packets, and set the encoding packet to be the bitwise exclusive-or of those d packets.
3. Repeat the above two steps until desired number of encoded packets have been generated.

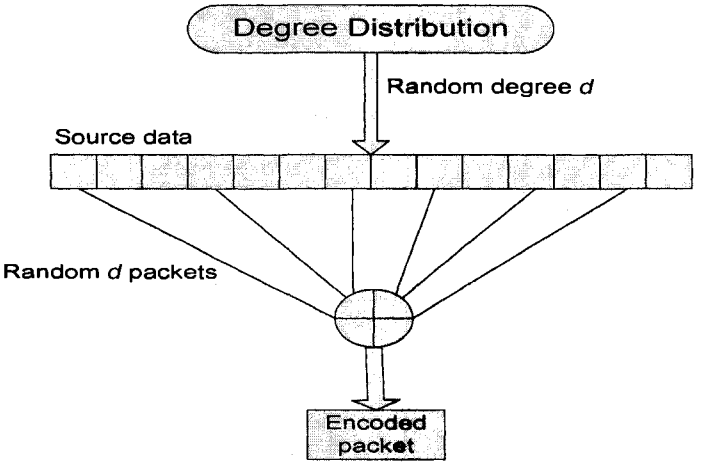

Figure 3.5: LT encoding

The above figure 3.5 roughly shows the process. The encoding process describes a graph connecting the encoded packets to source packets. If the average degree of this degree distribution is much smaller than the file size k, then the graph is sparse. Each encoded packet is independently and randomly generated as the exclusive-or of a particular subset of the source packets and is transmitted together with the information specifying the selected degree and source packets. By this information, the decoder knows which specific d source packets were used to generate each encoded packet, though the values of the source packets remain unknown until decoding is finished.

LT decoding

At the receiver, when using the received encoded packets to recover the packets of the

source data, the decoder has to know the degree d and the combination of source packets for every encoded packet. There are various methods to pass this information to decoder, depending on the applications. For example, it can be passed to receiver straightforwardly, or there can be an agreement in advance, where a key is assigned to each encoded packet.

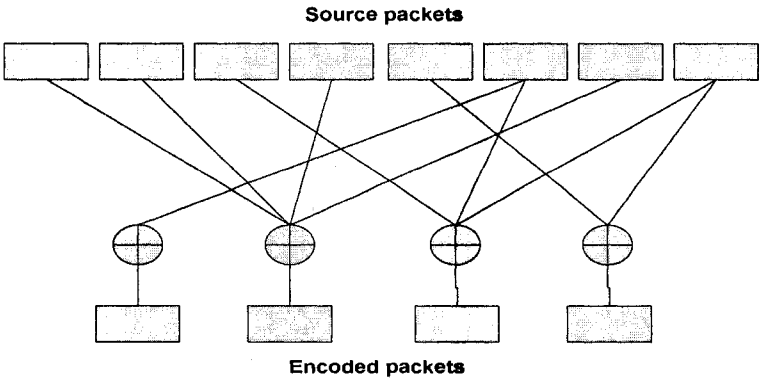

Figure 3.6: The degree distribution of encoded packets

A key could be the seed of a pseudorandom generator, in that case, the key is randomly chosen by the encoder to generate the degree d and the list of source packets used to generate the encoded packet, and then the key could be transmitted along with the encoded packets. At the receiver, the decoder applies the same key to obtain the degree and the list.

Here we assume that the decoder in some way knows the necessary information to complete decoding process, e.g. see figure 3.6. LT decoding can be effectively accomplished by so called 'Belief Propagation' (BP) method. A simple description of the LT decoding process is shown as follows:

1. Find an encoded packet t that is connected to only one source packet, say s, i.e., an encoded packet with degree one. In case that there is no such packet t, this decoding fails.

2. Set $s=t$. Since s is the only source that produces t, so t is sure that s has the same value. This 'belief' starts to propagate.
3. Add the value of s to all the encoded packets that are connected to s, so as to restore their previous values. So s continues to propagate the 'belief' to other packets.
4. Remove all the connections connected to s.
5. Go back to step 1 until all the source packets are determined.

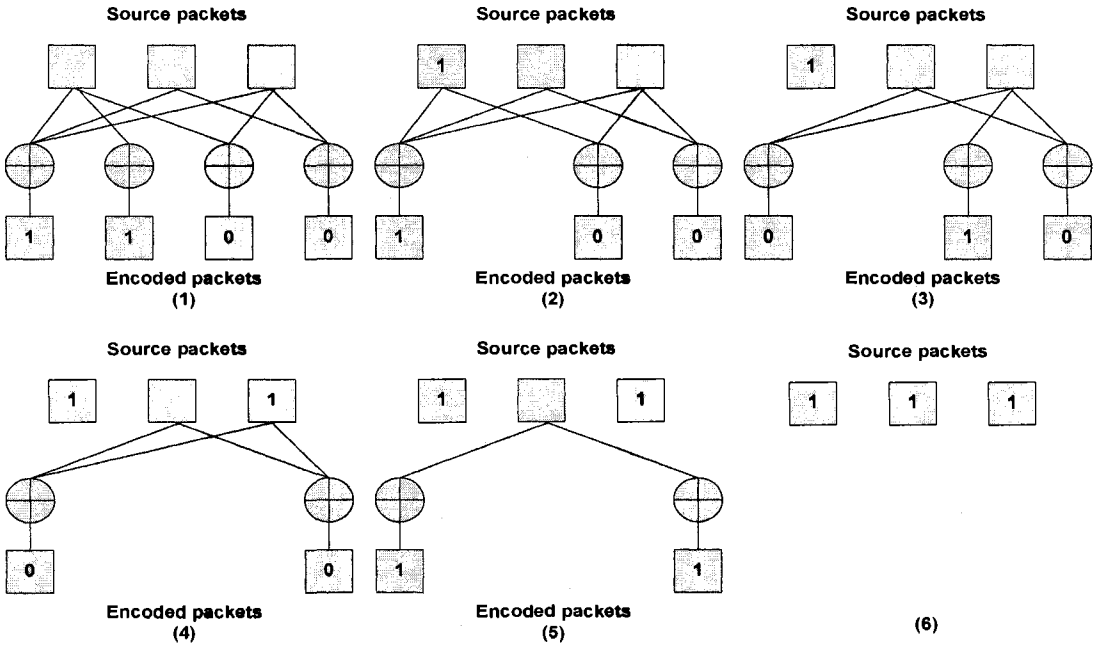

Figure 3.7: An example of LT decoding process

Here we give a very simple example of the LT decoding process, given that a packet contains only one bit. Suppose we received 4 packets, each has the value 1, 1, 0, 0 respectively, and there are 3 source packets. The decoding process is illustrated in figure 3.7 as above.

Through the algorithm described above, we know the basic idea of LT encoding and decoding process. Each encoded packet is generated randomly and independently using the same degree distribution, thus all received encoded packets are equally important in

decoding process. Also because encoded packets can be generated as many as necessary simply by repeating the encoding algorithm, LT codes are really fountain codes. The complexity of LT codes is directly related to the degree distribution. The smaller the average degree is, the less number of exclusive-or operations are performed in each encoding and decoding process, and obviously the simpler it becomes. Meanwhile, the degree distribution should let the decoder completely recover the whole source block with just a little bit more received encoded packets than the total number of source packets.

LT codes design

The design of the degree distribution is the most crucial part of LT code design. On one hand, some packets should have low degree so as to make the decoding process begin and to keep it running, and to minimize the computing cost of encoding and decoding processes. On the other hand, some rare encoded packet should have high degree, which means its degree is close to the total number of packets to make certain that there are no source packets that are not connected to any encoded packets. In an LT code, to decode successfully, every source packet must be connected to at least one encode packet. At each iteration, to be more efficient, generally we would hope that there is only one encoded packet that has a single connection with source packet. Also, we would hope that the end of each iteration would lead to the come-out of another new packet with single connection. This objective is perfectly achieved by Michael Luby with a carefully designed degree distribution called robust soliton distribution. The details of this distribution design is beyond the scope of this thesis, for more details please refer to [47].

3.2.2 Raptor codes

Raptor code [48] is an excellent development of LT codes. Raptor is a fountain code, so there is no limit to the number of encoded packets that can be generated from a given source block. As long as sufficient encoded packets are successfully received, no matter which specific encoded packets are received or in what order they are received, they can be used to recover the source data. Raptor codes are also universal in the sense that they operate close to capacity for any erasure channel with erasure probability less than one.

Rateless erasure code

Conventional FEC codes generate a fixed number of repair symbols. For example, an (220, 190) RS code has a code rate of 190/220 and can generates 220-190=30 repair or parity symbols from k=190 source symbols. Comparatively, Fountain codes can generate unlimited number of repair symbols without repetition, so generally no fixed code rate is applied. Raptor codes are almost an ideal erasure code because they can recover the source block from any set of k received symbols.

Raptor can be employed at transport layer or higher to provide packet level loss protection for communications networks. It is a packet level erasure code that has been designed and optimized for effective packet loss recovery, full flexibility and low complexity. Raptor provides full flexibility in the sense that, on one hand, Raptor can generate as many encoded packets from a source block as required to completely eliminate the effects of packet loss; on the other hand, Raptor can generate as few encoded packets as desired to control the latency or bandwidth occupation but still provide a desired level of packet loss protection, it is up to the requirements of the

particular application, namely Raptor codes are able to provide any level of protection against any level of packet loss.

Raptor encoding and decoding

The core component of Raptor is an LT code, which has relatively good performance but the complexity is not linear. Raptor codes use an LT code with average degree about 3. It is such a low average degree that some part of the source packets will be unassociated and will not be recovered. Therefore in order to get linear complexity, Raptor codes add a pre-coding stage prior to LT encoding to produce a little redundancy. A fixed length systematic code is used in the pre-coding stage to improve code performance with a little

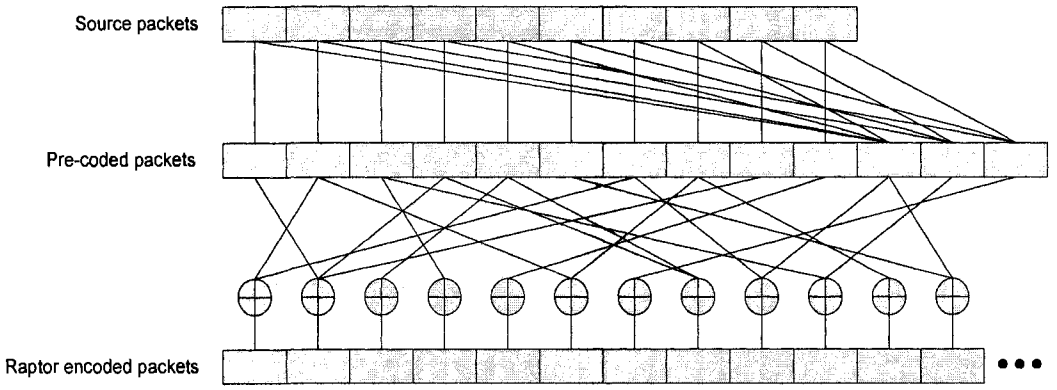

Figure 3.8: Raptor coding

bit extra redundancy. In particular, a low complexity pre-coding algorithm is applied to the source packets to create a pre-coded block, and then an LT code is applied on the pre-coded block to generate unlimited encoded packets. At the receiver, the decoder simply applies the inverse procedures: decodes the received encoded packets in the same way as LT code, and then uses the recovered pre-coded packets to get the source block.

The performance of Raptor has been optimized by careful design of the pre-coding algorithm and of the degree distribution used in the LT code, so its complexity increases linearly with the source block size and it can completely recover the source block with the reception of encoded packets a bit more than the source block size. Raptor is usually implemented as software running on a general purpose processor without the need for special hardware. Raptor code can achieve very high encoding and decoding speeds on a variety of platforms with relatively low processing powers, making it practical for various applications and services. Especially, Raptor code is ideal for consumer electronics such as cellular phones and PDAs, and no additional supporting hardware is needed.

Raptor versus Reed Solomon

When used as packet level FEC codes, RS codes become more inefficient and restrictive, its quadratic running times are too large for many applications. The major difference between Raptor and RS codes is that Raptor codes do not have restraints on source block size. Unlike RS codes, the packet size of Raptor codes can be less than or equal to the source packet size, it can be as small as necessary. The processing requirement of the Raptor code increases linearly according to the source block size, while it increases quadratically in RS code. RS codes require to be implemented in hardware at decoder due to the relatively high processing requirements. So Raptor code has much faster processing speed and needs much less processing requirement than RS code, offers more efficient packet loss protection than RS codes.

Raptor Codes provide full flexibility, offering the option to balance between the degree of packet loss protection, processing speed and bandwidth occupation; while RS codes suffer from restrictions that decrease their performance and limit their effectiveness, optimization in one dimension causes poor trade-offs in other dimensions.

Raptor and Reed Solomon codes in mobile standards

Application layer FEC uses FEC codes to protect against IP packet loss, it is a kind of erasure code not error correcting code. When used in application layer FEC, an FEC code must be strongly specified according to the specific application. So application layer FEC often shows pros and cons of different FEC coding technologies.

Raptor and RS codes are both widely accepted as application layer FEC by various mobile standards. In 3GPP MBMS (Multimedia Broadcast Multicast Service), they both have been evaluated extensively for both file delivery and streaming, Raptor has been selected for both streaming and file delivery; in DVB-H (Digital Video Broadcasting – Handheld) IP datacast, RS are already used at link layer, Raptor has been evaluated extensively and selected for file delivery; in 3GPP2 BCMCS (Broadcast and Multicast Service), Raptor will be proposed for both streaming and file delivery.

3.2.3 Systematic Raptor codes

Systematic codes are better for some applications. As in real time applications like streaming video, when the receiver cannot recover the whole source block because no enough encoded packets have been received, but could still show part of the information if the code is systematic. First, in order for Raptor coding to be systematic, it needs to be

sure that the encoding process is invertible. Raptor codes can then be made systematic by first multiplying the input source data with the inverse of the first k columns of the generator matrix, which is like the inverse of the encoding process, then applying the normal encoding process to the resulting packets, which lets the original source packets to be regenerated as the first k output packets of the code. Systematic Raptor code has been standardized as the application layer FEC code for the 3GPP MBMS.

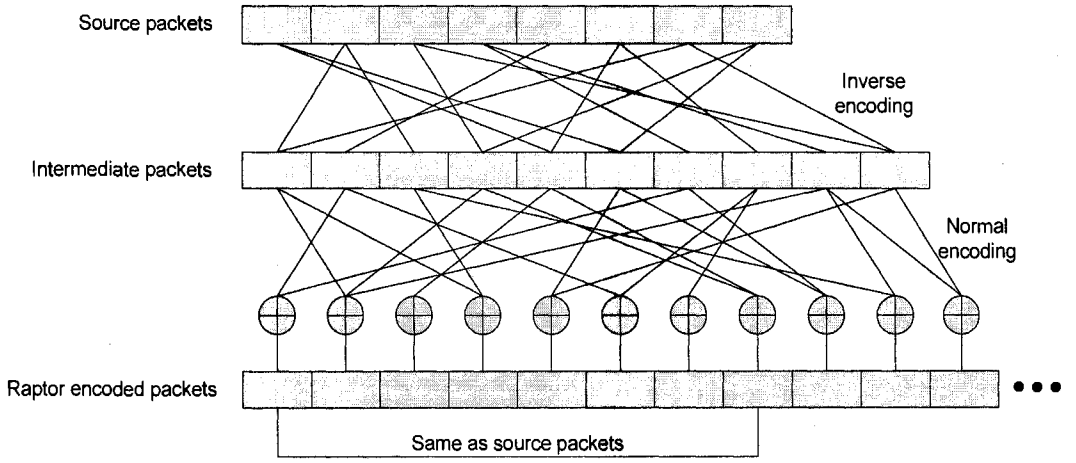

Figure 3.9: Systematic Raptor encoding

3.3 System level simulations for various networks

According to Michael Luby [49], generally, there is a simple way to estimate the performance of raptor codes, that is, if a block consists of more than 200 packets, the little inefficiency of the raptor code can be well modeled by the equation:

$$P_f(m,k) = \begin{cases} 1 & if\ m < k, \\ 0.85 \times 0.567^{m-k} & if\ m \geq k. \end{cases} \qquad \text{Equation 3.1}$$

whereby $P_f(m, k)$ represents the failure probability of the code with k source packets if m packets have been received. Although this equation is quite simple, however, it is the foundation of simulation of raptor code, which is an important component of the whole simulation system. In [49], some research had been done to prove the accuracy of the above equation. Figure 3.10 shows the simulation results from [49] for some selected cases and also a graph of the formula. In the figure, the strait line represents the formula; it can be shown that for different k, the equation almost perfectly emulates the raptor code performance.

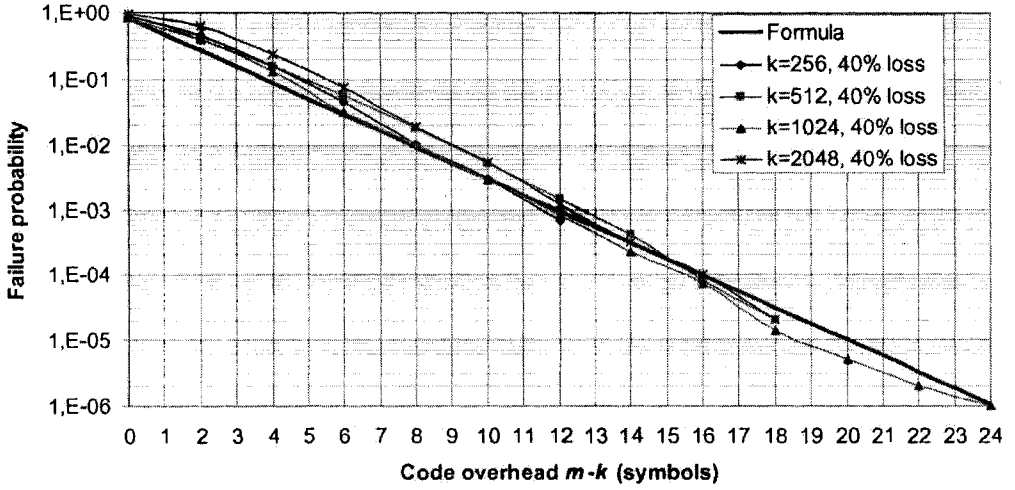

Figure 3.10: Raptor block loss rate for different received overhead m-k, different block size k, and channel packet loss rate of 40%. [49]

Figure 3.11 shows how the probability of successfully decoding a block is affected by the number of redundant packets received by the receiver after the erasure channel. It can be seen that the success probability increases exponentially with the increasing number of received packets. Figure 3.11 shows that for $m=k$, with no extra redundant packets received, it still has a chance of 15% to completely recover the source block; with the

number of received redundant packets increasing to 8, that is, for $m-k=8$, it has a success probability of 98.4%; when received 16 redundant packets, that probability goes to 99.98%, very close to 100%. In the last case, suppose we have a source block size of 1024 packets, to get a Block Loss Rate (BLR) of 10^{-3}, the received overhead needs to be 16/(1024+16), which is only about 1.56%.

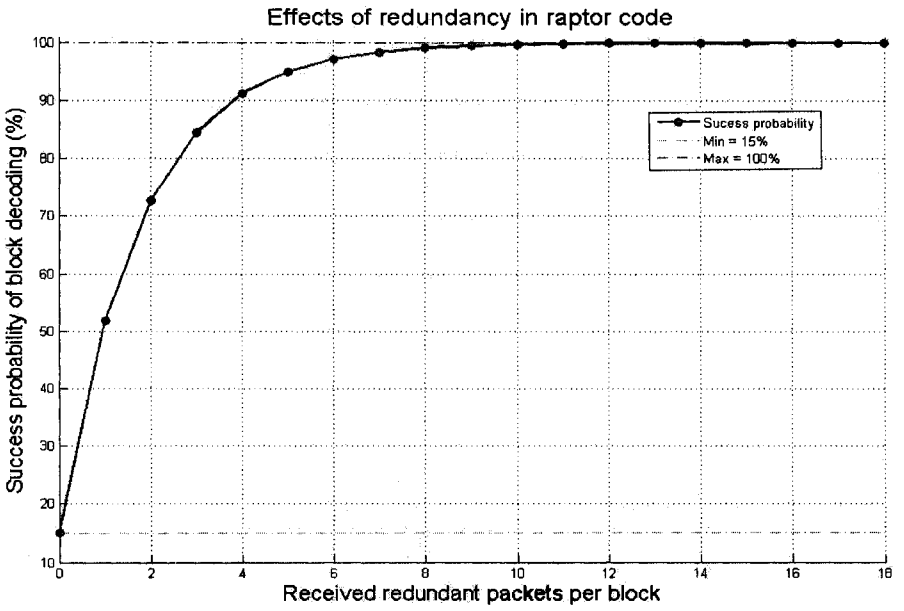

Figure 3.11: Effect of received redundancy (m-k) of raptor coding

3.3.1 Simulation set-up

The simulation system (Fig. 3.12) is constructed and coded in Matlab. It simulates a virtual wireless ad hoc network with static routing scheme. The network is created by generating a certain number of random node locations in a limited area, then applying a shortest path algorithm to the nodes to get a specific routing scheme. The file to be transferred is first split into one or more blocks with the same size, and block is the basic processing unit of the error recovery system. The channel access method then simulates a

virtual wireless channel access mechanism as if there is an effective Media Access Control (MAC) technique like IEEE 802.11 RTS/CTS. Then according to the systematic raptor coding procedures described above, the sender encodes the entire source block to be

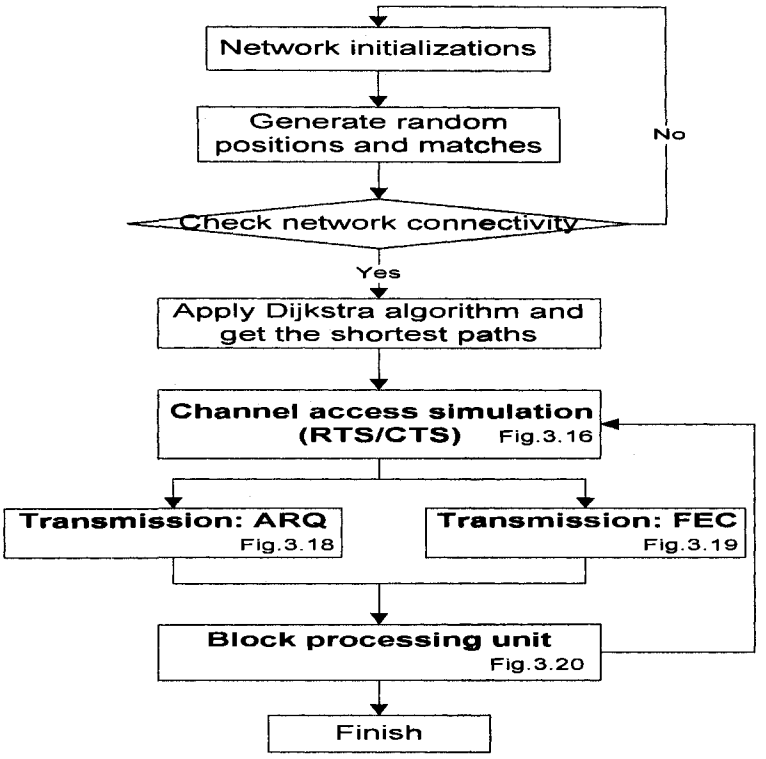

Figure 3.12: General flowchart of the simulation system. Note some flowcharts are presented in order to show a general structure of the simulation program, so they may not be necessarily organized in natural time sequence.

transferred, generating another set of source block plus a certain amount of redundant packets. The number of redundant packets generated to achieve the best performance depends on various conditions, such as the Packet Loss Rate (PLR) of the physical

channel, the desired degree of loss protection, number of intermediate hops all the way to destination and the level of Quality of Service (QoS).

In this simulation, the default mode of packet transfer is application layer FEC based on UDP, there is no need of acknowledgements (ACKs), and the sender just keeps sending packets whenever it is sure that the channel is free. This saves a great deal of media resources, which is extremely important for improving the efficiency of network, especially for wireless network with a very limited channel resource shared by many wireless users. The sender counts the number of packets it transmits, and stops transmitting when a predetermined number of packets for the block have been transmitted. However, these packets are probably not all delivered, as some of them might be lost due to the lossy channels. Suppose that each packet is error detected, for example, by CRC, whose tiny probability of malfunction is ignored. After all of these packets either go through the entire path and get to the destination or get lost en route, in a few cases, no data packet is lost, so the program will skip over the decoding process and continue on next block; however, in most of the cases, some data packets will be lost. Therefore, the receiver initiates the Raptor decoding process. If $m < k$, decoding will fail; otherwise it succeeds with a certain probability, depending on the value of $m - k$. In case of decoding failure, we are able to get only a part of the data packets, the program will check the sequence gap between the original data packets, record them, and transmit only the lost data packets with ARQ.

For the sake of simulation simplicity and to avoid extensive encoding and decoding costs for every end user, our system level simulator employs this equation-based (Eq. 3.1) model for Raptor code as an alternative, which significantly accelerates the simulations

and saves a lot of simulation time and resources, without losing generality or accuracy of simulation. More particularly, to find out that if the raptor decoder would decode a given amount m of received packets, a uniformly distributed random variable is sampled and compared with P_f in Eq. 3.1. If it is less than P_f, then raptor decoding fails, otherwise it succeeds. Detailed description of encoding and decoding processes of raptor codes has been presented in previous sections.

Each packet has a sequence number, the receiver records the sequence numbers of all packets that have been lost or received in error. If the block decoding succeeds, the process continues working on the next block; in case of failure, the system switches to ARQ transfer mode, in which case, only the lost source packets will be delivered by ARQ, so together with the already received packets, the receiver will have all the source packets in this block at the end, no further decoding is needed. Figure 3.12 shows a general system flowchart of the whole idea described above. Detailed description of simulation system implementation will be presented in next section.

To be more efficient, some concerns are ignored in this simulation. For ARQ, we ignore the little transmission time and tiny failure probability of ACKs as a convention; for Raptor codes, we ignore the processing time of encoding and decoding. As a convention, we also ignore the signal propagation delay in all cases. In addition, queuing delay is ignored, i.e. the nodes have infinite cache to store packets. However, we instead focus on the most important problem in static wireless ad hoc networks – channel contention, so we simulate the channel access control with respect of both hidden node problem and exposed node problem; also we employ the shortest path algorithm and implement routing scheme for the network.

3.3.2 Detailed system implementations

Network Initializations

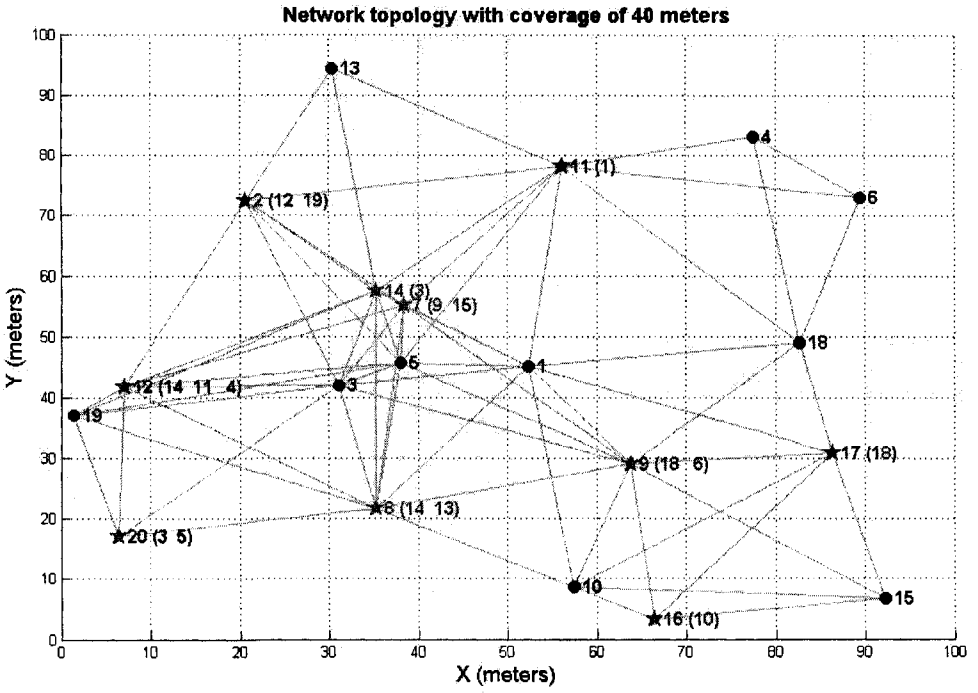

Figure 3.13: An example, a network of 20 nodes with a node coverage of 40 meters. The stars denote the source nodes, with the intermediate stops in the bracket, and the dots denote the destination nodes. A line between two nodes means that they are in the coverage of each other. The same to all of the following figures.

As in figure 3.13, to create a network at the beginning, first a certain number of nodes is chosen, let us say, *2*U*, so there are *U* source-and-destination pairs. Suppose these nodes are located in a certain range of area, say a *100*100* m^2 area, and each has a certain coverage radius (the value may vary from 30 to 70 meters), generate a random position for all of these *2*U* nodes within this area, record the exact locations for each user. Then

randomly match them up to U pairs of source-and-destination nodes. According to the predetermined radius, connect all the nodes in its coverage for each user, record these relations, and make sure every source-to-destination path is connected by either single hop or multiple hops. In case that the path cannot be realized, the simulation program will modify the network topology. By using the location and connection records, the program calculates the shortest path for every pair of source-to-destination nodes according to the Dijkstra's algorithm, which is usually the basis of Link State (LS) routing protocols, with Open Shortest Path first (OSPF) and Intermediate System to Intermediate System (IS-IS) being the most basic ones. A basic principle of Dijkstra's algorithm is given in the appendix.

Channel access method

This subsystem builds a virtual scheme to simulate Request to send / Clear to send mechanism (IEEE 802.11 RTS/CTS), which is used by the 802.11 wireless networking protocols to reduce frame collisions introduced by the hidden terminal problem. Both hidden terminal problem and exposed terminal problem are taken into account in this virtual scheme. Furthermore, the fairness of channel access among wireless users is also considered and well implemented.

In wireless networks, the hidden node problem, see figure 3.14, arises when a node A is not in the signal coverage of node B, yet it is in the coverage of the third node C which is communicating with the B. This leads to difficulties in media access control. To conquer this trouble, handshaking is implemented in conjunction with the Carrier sense

multiple access with collision avoidance scheme (CSMA/CA). The same trouble exists in a Mobile Ad-hoc Network (MANET).

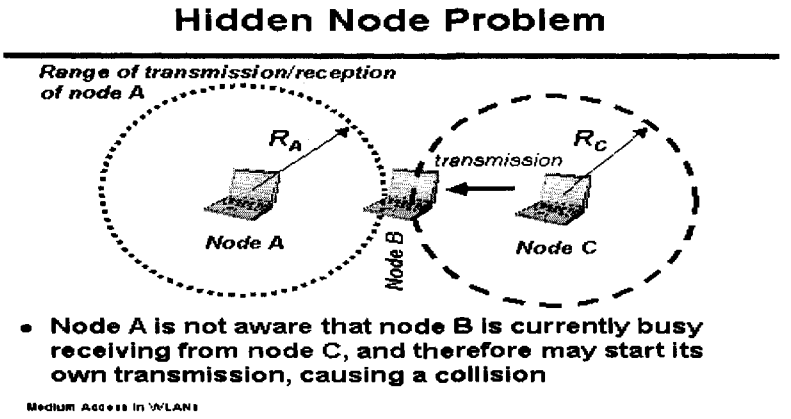

Figure 3.14: Hidden node problem

Nevertheless, the exposed node problem, see figure 3.15, occurs when a node is prevented from sending packets to other nodes due to an adjacent transmitter. Both of these two situations may cause problems, the first one could result in conflicts and the latter can lead to an inefficient utilization of wireless channel.

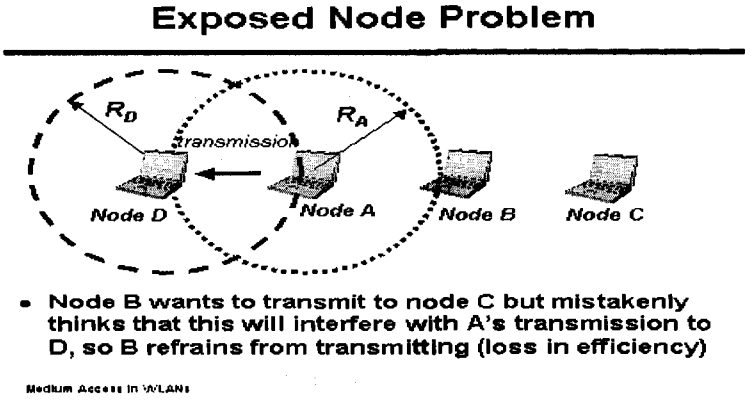

Figure 3.15: Exposed node problem

During the simulation, the duration of each system iteration is one packet transmission time, which can be obtained by dividing packet size with channel capacity. The counter of iteration is updated at the beginning of every system iteration, and then followed by the reset of node status indicators. There are three node status indicators in the simulation system, they are *transfer*, *receive* and *listen*, respectively. A true value of *transfer* means this node is transmitting packets to another node; *receive* indicates this node is receiving packets from another node; *listen* means that in the range of this node, there is another node in the status of *'transfer'*. It should be noticed that, a node in the status of *'listen'* is not necessarily *'receive'*; however, if a node is *'receive'*, it must be *'listen'* as well.

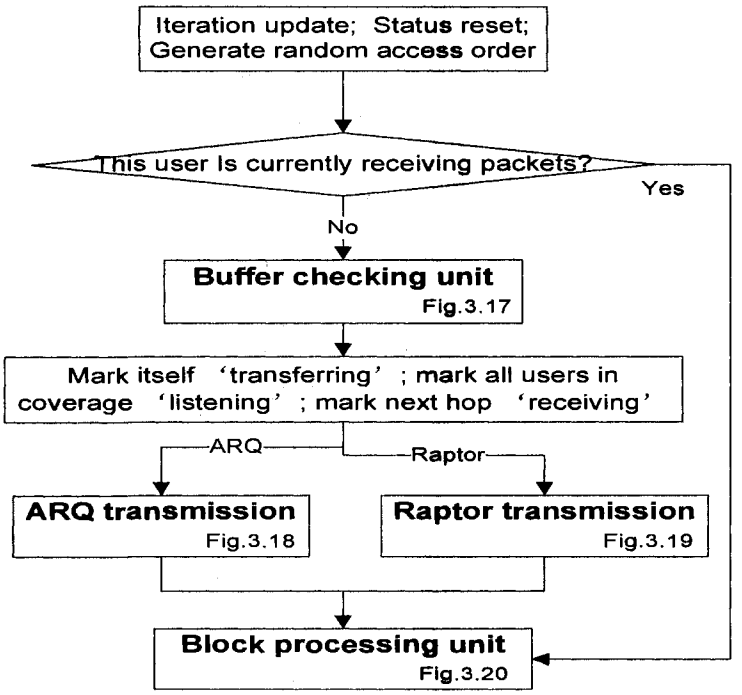

Figure 3.16: Channel access method

To realize channel access control, in this channel access method, we create a virtual mechanism as an effective RTS/CTS scheme. To achieve the fairness of channel access,

at the beginning of each system iteration, a uniformly distributed random access order of these nodes is generated, and according to this order, all the nodes will have a chance to access the channel and will be asked if they have the intention to transfer data and then provide related information. If this user is in the process of receiving packets from another user, i.e. the value of '*receive*' is true, then it will not sense the channel and certainly will not compete for channel access. Otherwise, it competes for a chance to use the channel to transmit packets. Figure 3.16 is a rough demonstration of channel access processes.

Buffer checking unit

Before we continue on the channel access process, let us take a moment to understand the buffer checking unit, figure 3.17 roughly shows the buffer checking process. The major task of this unit is to get a list (let us call it 'checking list') of available data stored in the buffer of the node. To achieve that, it checks the data buffers from every source and makes sure the following:

1. This path must have data to transfer, i.e. its buffer is not empty. In particular, the corresponding counter is not zero.

2. The next hop of the data of this path must not be listening to any other node, so there would be no conflicts. In particular, the *'listen'* of next hop is zero. This overcomes the hidden node problem in an effective way, although does not actually deal with it.

3. Of course, the next hop must not be transferring data. In particular, the *'transfer'* of next hop is zero.

4. Any node in this node's coverage is not receiving from any other nodes. In particular, the *'receive'* of any node in this node's coverage is zero. This overcomes the exposed node problem in an effective way, because this node does not care if its neighbouring nodes are 'transfer', but only cares if they are 'receive', as long as they are not 'receive', there would be no interference.

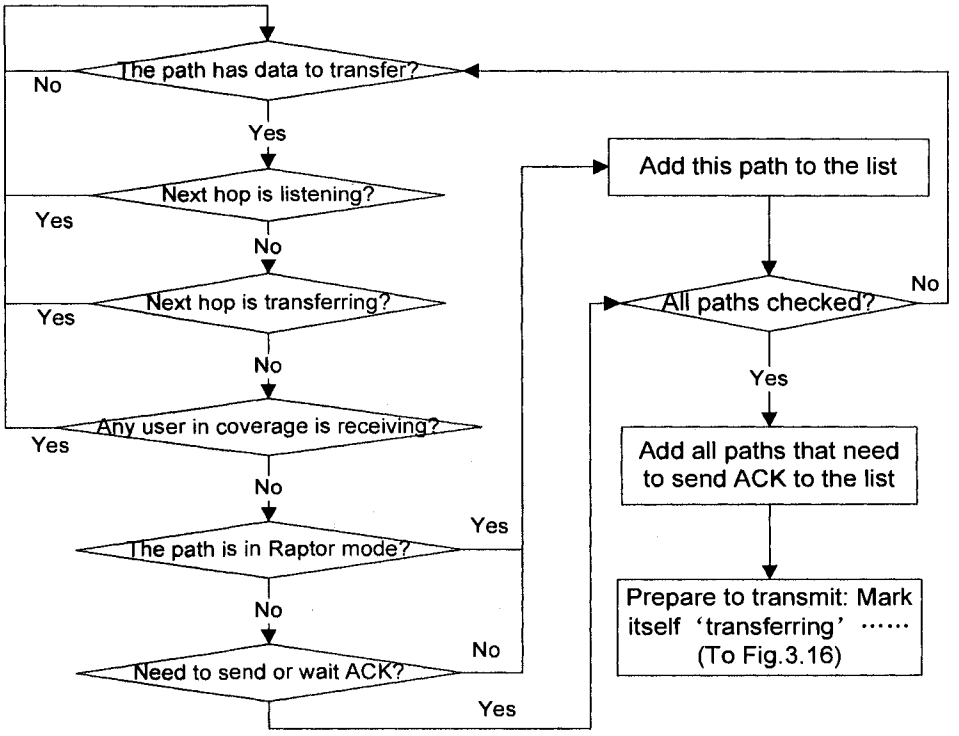

Figure 3.17: Buffer checking unit

After checking the above conditions, in Raptor mode, the information will be directly added to the 'checking list'; while in ARQ mode, it will be added only if this path does not need to send or wait an ACK, otherwise the information will be disregarded. After checking data buffer from all sources, the program then checks all the sources if they need to send an ACK, records this information and adds it to the checking list. The

format of the entry is basically like [Source-Destination ID, Next hop ID]. In case that the check list is empty, then the system directly goes to the next node in the above mentioned random order. Other than that, the real transmit procedure will start. First, randomly choose one entry from the checking list and get the Path ID (Source-Destination ID) and the next hop ID. Then followed by a serial of status updates: marking itself *'transfer'*; marking all the neighbouring nodes *'listen'*; marking next hop *'receive'*. Then by checking the value of ARQ indicator *'arq'*, goes to '*ARQ transmission*' or '*Raptor transmission*', which are explained as follows.

ARQ transmission

Figure 3.18 shows the rough procedures of ARQ transmission, the fundamental mechanism behind is based on the traditional Stop-and-Wait scheme. When it comes to

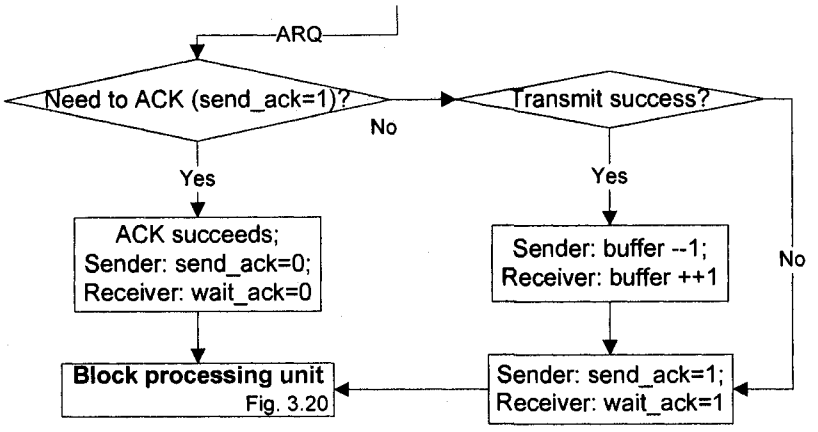

Figure 3.18: ARQ transmission

ARQ transmission, it first needs to know if it needs to acknowledge the last transmission by checking a variable *send_ack* indicating the status of ACK. If it needs to ACK, we assume the failure probability of all ACK transmissions are zero all the time, so the ACK

always succeeds. Then it turns off the ACK status indicator of both sender end and receiver end, and goes to the block processing unit. If it does not need to ACK, it directly transmits the packet, if success, it updates the buffers of both sender and receiver. To determine if a transmission succeeds or not, a uniform random variable is sampled and compared with the PLR, if it is greater than PLR, then the transmission succeeds, otherwise it fails. No matter whether transmission succeeds or fails at the end, the ACK status indicator of both sender and receiver will always be set to 'on' to remind the receiver that it has to ACK or NAK first, and the sender needs to wait for the results. Then it goes to the block processing unit in the following.

Raptor transmission

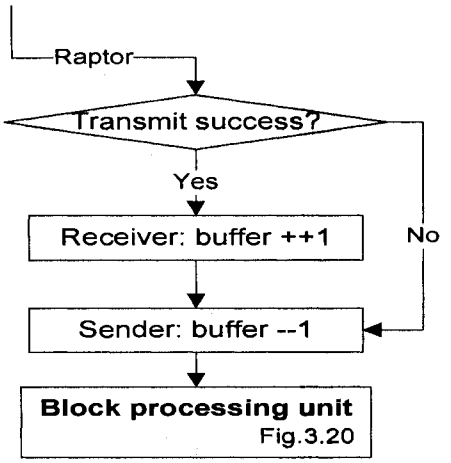

Figure 3.19: Raptor transmission

Raptor transmission is quite simple; figure 3.19 shows its rough procedures. If the transmission succeeds, it simply updates the receiver's buffer. Because it is Raptor transmission, there is no retransmission, if you fail to transmit a packet, you lose it, so it

then updates the sender's buffer no matter it is a fail or success. After that, it goes to the block processing unit in the following.

Block processing unit

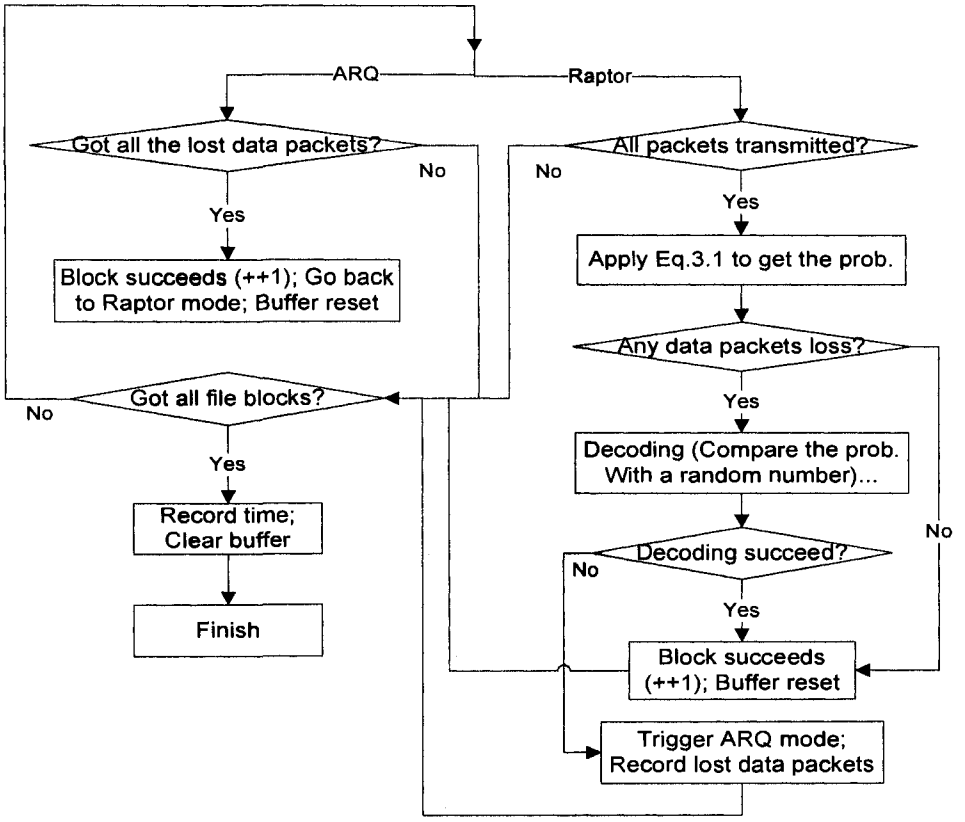

Figure 3.20: Block processing unit

This unit is mainly for virtual carrier sensing, which is divided into two parts, respectively concerning ARQ and Raptor coding. For the ARQ part, if the receiver has got all the lost original data packets, this block is finished; it then updates the block counter and turns off the ARQ indicator to go back to default Raptor transmission mode. It also resets the buffers of both source and destination end, so as to get ready for the next block transmission.

The Raptor part is more complicated. Given that all the predetermined amount of packets has already been transmitted, the system applies the equation-based model (Eq. 3.1) to get a failure probability for the decoding. If, fortunately, there is no original data packets loss, then the program will skip over the decoding process, and the block directly succeeds, no need to decode. That is efficient and advisable since decoding might end up with a failure. However, generally there are some source packet losses, so here comes the decoding. This is rather simple, the program just calls a uniform random number and compares it with the probability for this decoding to decide whether it succeeds or fails. When Raptor decoding succeeds, it updates the block counter and clears the buffer; when Raptor decoding fails, it triggers the ARQ transmission mode, records all the lost data packets. After the above processes, if all the file blocks have been processed for the subject source-to-destination path, then the system records the total file transmission time on this source-to-destination path. Figure 3.20 shows the rough processes of the block processing unit.

3.3.3 Results and contributions

This section shows a summary of the major contributions and simulation results for various sizes of networks.

The parameters and statistics

The file size typical value is 512 Kbytes, which is 512*1024*8 bits. Packet size is 512 bytes, so there are 1024 packets in total. If we consider the whole file as a source block, the block size is then 1024 packets.

Input

Input parameters include the size of the network, Packet Loss Rate (PLR) and redundancy R. The sizes of networks we simulate are 4, 8 and 12nodes. The PLR we used in the simulation has the typical values of 10^{-4}, 10^{-3}, $10^{-2}$and 10^{-1}.

The design of the redundancy is a tricky problem because the required redundancy crucially depends on the number of hops throughout the source-to-destination path. Suppose the block size is B, the minimum required redundancy ratio is R_m, and the packet loss rate is P. It is obvious that the expected number of packets received at the destination should at least equal to source block size B to make the Raptor decoding work. The expected number of packets received by the destination node is $B \times (1 + R_m) \times (1 - P)^n$, where n is the number of intermediate hops. In order to make the raptor code works, the minimum redundancy required is when the equation $B \times (1 + R_m) \times (1 - P)^n = B$ holds, which gives us:

$$R_m = \frac{1}{(1-P)^n} - 1 \qquad \text{Equation 3.2}$$

Corresponding to this equation, table 3.1 gives the approximate values of the minimum required redundancy for different combinations of PLR and number of hops.

Now, let us investigate a little bit about the actual redundancy R, for a certain path with its corresponding R_m, if $R < R_m$, Raptor code will have no contribution to the performance at all, in fact, it even degrades the performance because of useless redundancy; on the other hand, if $R > R_m$, then Raptor code starts working, but when it is too high, it becomes meaningless because it has been already enough to make sure the

decoding will succeed. From the beginning of this section, according to Eq. 3.1, we can get that when Raptor decoder receives 16 extra packets on top of the already existing minimum required redundancy, the decoding will succeed with a very high probability of

R_m & R	1 hop	2 hops	3 hops	4 hops	R (For all)	R (For all)	R (For all)
	Pc_i=15%	Pc_i=15%	Pc_i=15%	Pc_i=15%	P_c=91.21%	P_c=99.09%	P_c=99.99%
$P=10^{-4}$	~ 0	~ 0	~ 0	~ 0	R_m+0.39%	R_m+0.78%	R_m+1.56%
$P=10^{-3}$	0.1%	0.2%	0.3%	0.4%	R_m+0.39%	R_m+0.78%	R_m+1.56%
$P=10^{-2}$	1.0%	2.0%	3.1%	4.1%	R_m+0.39%	R_m+0.78%	R_m+1.56%
$P=10^{-1}$	11.1%	23.5%	37.2%	52.4%	R_m+0.39%	R_m+0.78%	R_m+1.56%

Table 3.1: The minimum required redundancy R_m and actual redundancy R for different situations. P_c is the success probability of Raptor decoding, and the column of actual redundancy R is for all lengths of paths, it is the corresponding R_m plus a certain value.

P_c = 99.99%. So we have $B \times R - B \times R_m = 16$, giving us $R = R_m + \frac{16}{B}$, for B = 1024, $R \approx R_m + 1.56\%$, one can see the difference is very tiny related to the value of R_m especially when PLR is high, as in the Table 3.1 above.

Output

The efficiency of the i^{th} source-to-destination path can be derived from this equation: $\eta_i = \frac{B}{T}$, where T denotes the total transmission time (in packet transfer time unit) used to successfully deliver all the packets in the block, which includes all kinds of factors such as channel access, packet loss, redundancy, decoding failure, retransmission and waiting for ACK. Thus, the average network efficiency can be calculated as the sum of all the single source-to-destination path efficiencies divided by the total number of paths, which

yields us $\overline{\eta} = \frac{\sum_{i=1}^{U} \eta_i}{U}$, where U denotes the total number of source-destination paths in this network.

Contribution 1: The recent FEC-based Raptor coding technique combined with ARQ-based selective retransmission method constructs a new hybrid ARQ method, which outperforms purely ARQ-based method.

With the employment of the recent FEC technique, i.e. Raptor code, it is possible to offer potentially unlimited redundancy, that is, it can offer any desired amount of redundant packets. Without having to acknowledge for every transmission, hybrid ARQ can avoid the extremely strong channel contention among wireless users, so it brings great improvement. In addition, with the selective retransmission, the sender only needs to retransmit some of the lost packets, the lost data packets, without wasting time on the lost repair packets. This effect becomes more noticeable in larger networks, because the channel contention becomes even stronger and the advantage of no need to acknowledge is greater, see Figure 3.22 – 3.25, 3.27 – 3.30 and 3.32 – 3.35.

Here are some selected results from lots of simulations to show the effects mentioned above. Figure 3.21 shows the network topology of a small 4-user network, with the node coverage radius of 50 meters. It has two source-to-destination paths: 2 – 4 and 3 – 4 – 1. Figure 3.22 – 3.25 show the performance comparisons between this hybrid ARQ and pure ARQ under different channel conditions, it can be seen that the improvement is huge, almost twice as the pure ARQ.

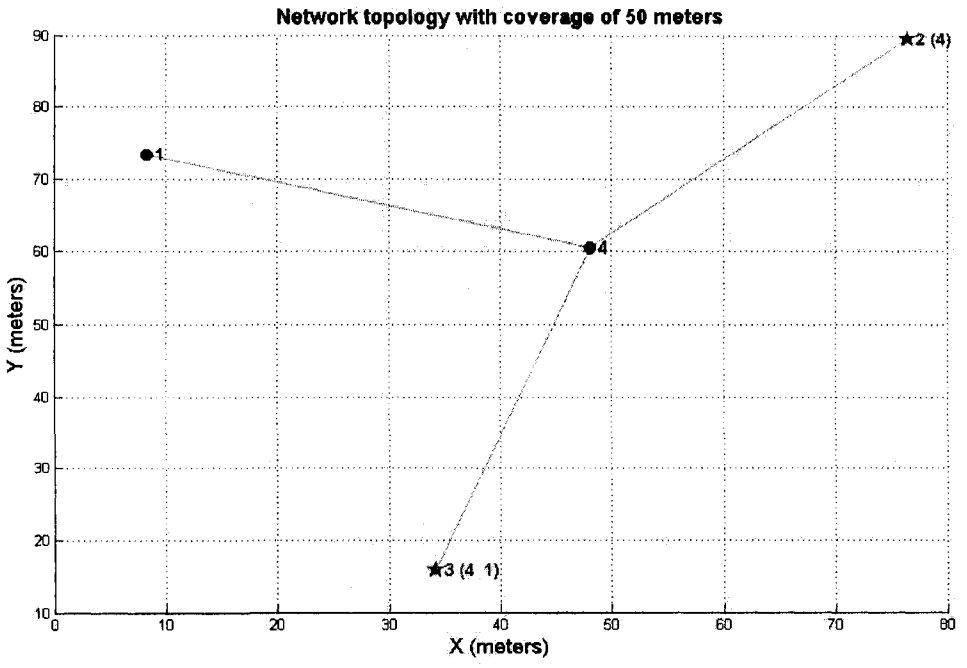

Figure 3.21: A small 4-user network

It should be noted that for the Raptor coding system used in hybrid ARQ, the redundancy has to reach a certain minimum required redundancy value R_m as listed in Table 3.1, so as to make the decoding work. Before reaching this value, the redundancy does not make any contributions to the performance. On the contrary, it actually decreases the performance because of the extra transmissions and waiting for channel access. This can be shown in many figures like Figure 3.22 – 3.25, 3.27 – 3.30 and 3.32 – 3.35. It is more obvious when PLR is high, such as in Figure 3.25, 3.30 and 3.35, one can see that all the curves go down at first, start to go up at a certain point and then reach a maximum at some point.

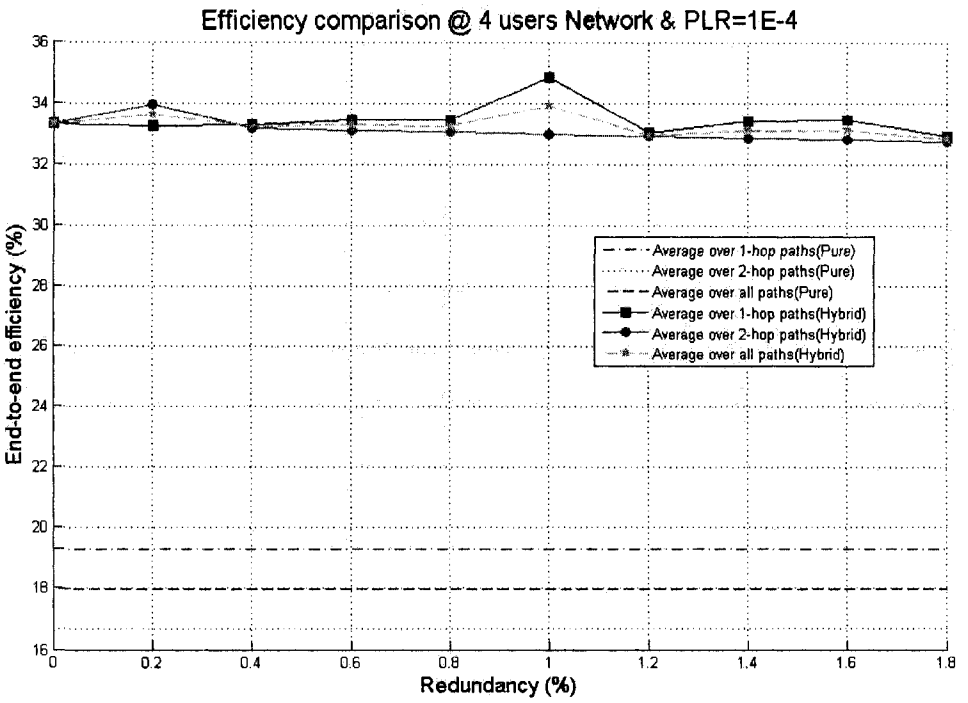

Figure 3.22: Efficiency comparison between hybrid ARQ and pure ARQ

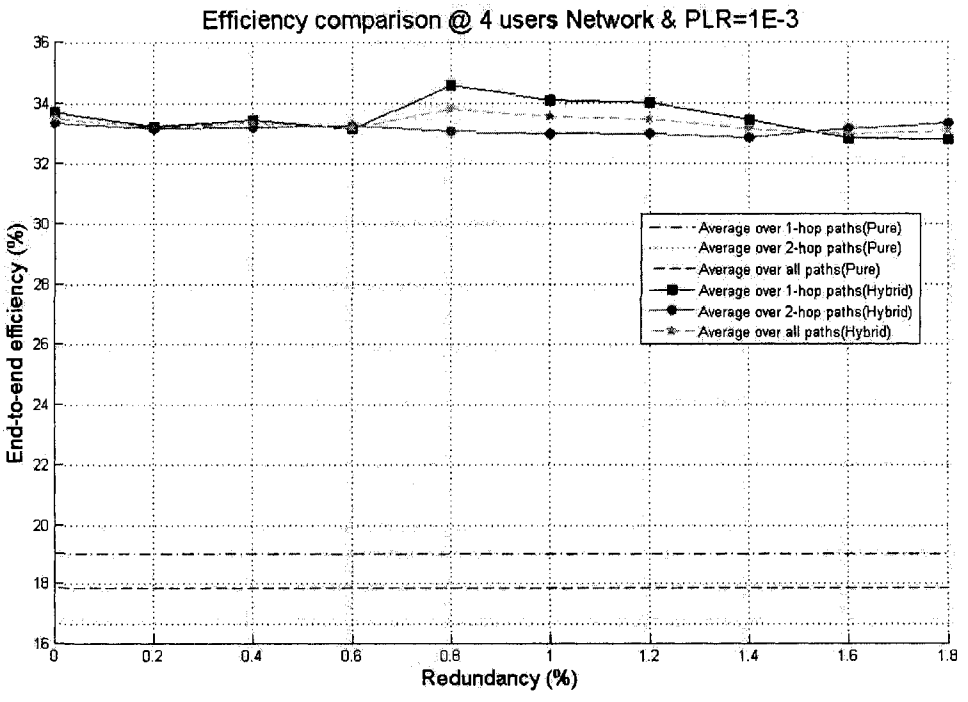

Figure 3.23: Efficiency comparison between hybrid ARQ and pure ARQ

In the hybrid ARQ, when the redundancy becomes very high, the success probability of Raptor decoding reaches a very high value which is very close to 100%, and keeps approaching it unlimitedly. At certain point, it will be high enough that the extra redundancy becomes useless, so the performance will gradually decrease. This can be shown in many figures like Figure 3.22 – 3.25, 3.27 – 3.30 and 3.32 – 3.35.

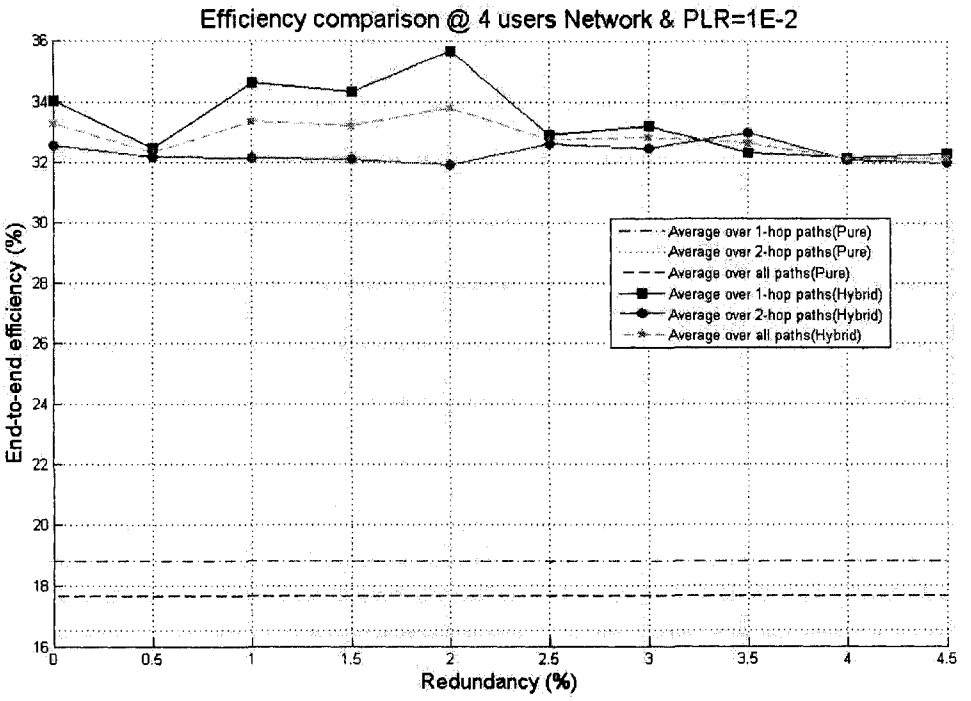

Figure 3.24: Efficiency comparison between hybrid ARQ and pure ARQ

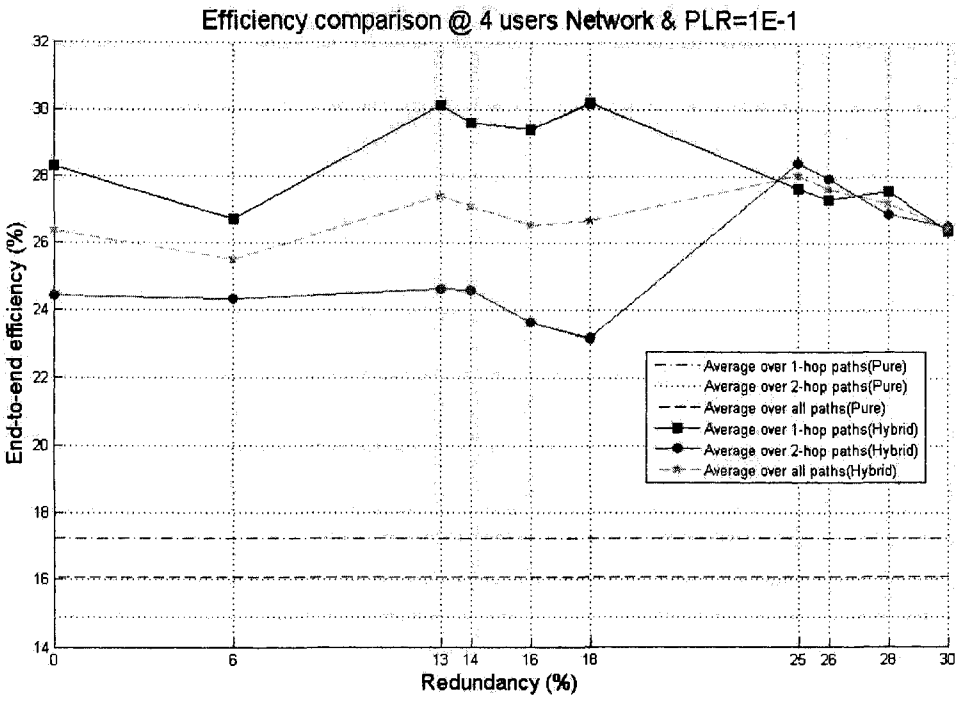

Figure 3.25: Efficiency comparison between hybrid ARQ and pure ARQ

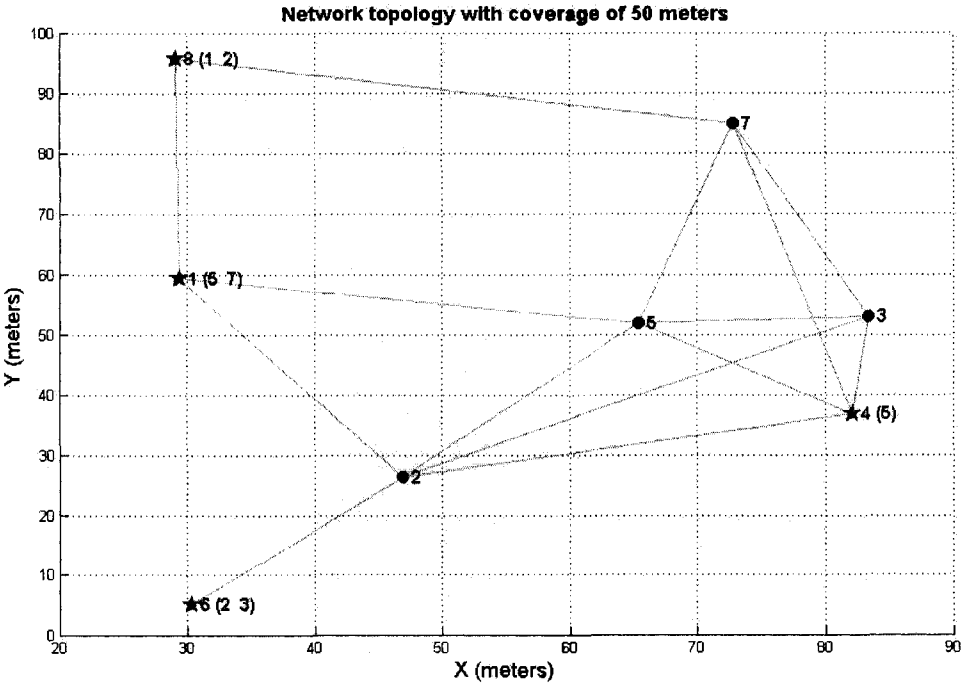

Figure 3.26: An 8-user network

Figure 3.21 shows the network topology of an 8-user network, with the same node coverage radius of 50 meters. It has four source-to-destination paths: 8 – 1 – 2, 1 – 5 – 7, 6 – 2 – 3 and 4 – 5. Figure 3.27 – 3.30 show the performance comparisons between this hybrid ARQ and pure ARQ under different channel conditions, it can be seen that the improvement is still very big, although not as huge as the 4-user network. Also, it should be noticed that the performance decreases significantly compared with 4-user network, this makes sense because in larger network, the channel contention is stronger, so it means fewer chances of channel access and thus more waiting time.

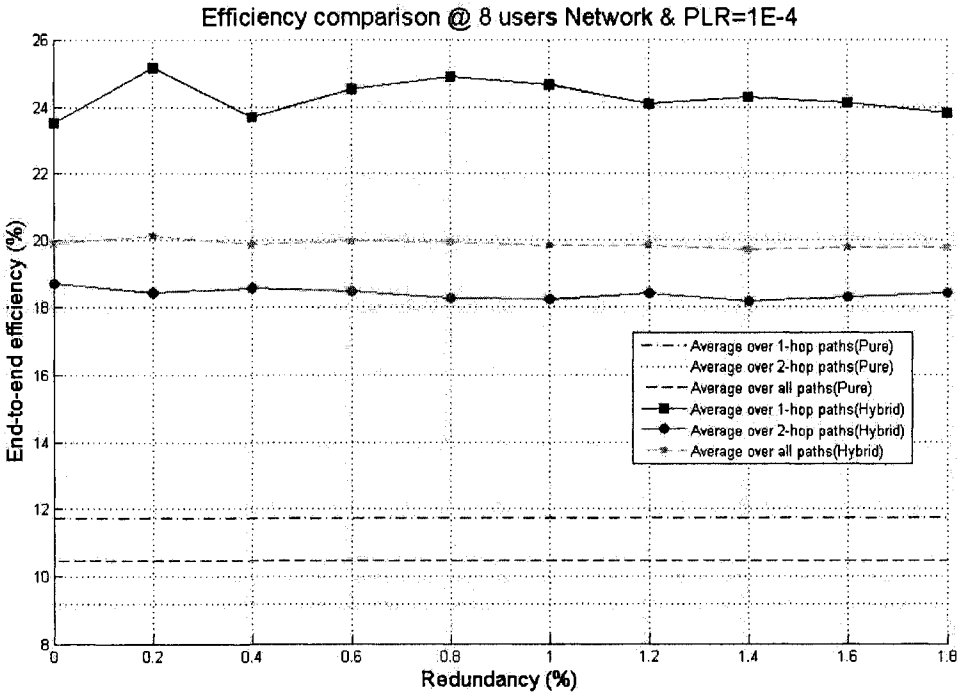

Figure 3.27: Efficiency comparison between hybrid ARQ and pure ARQ

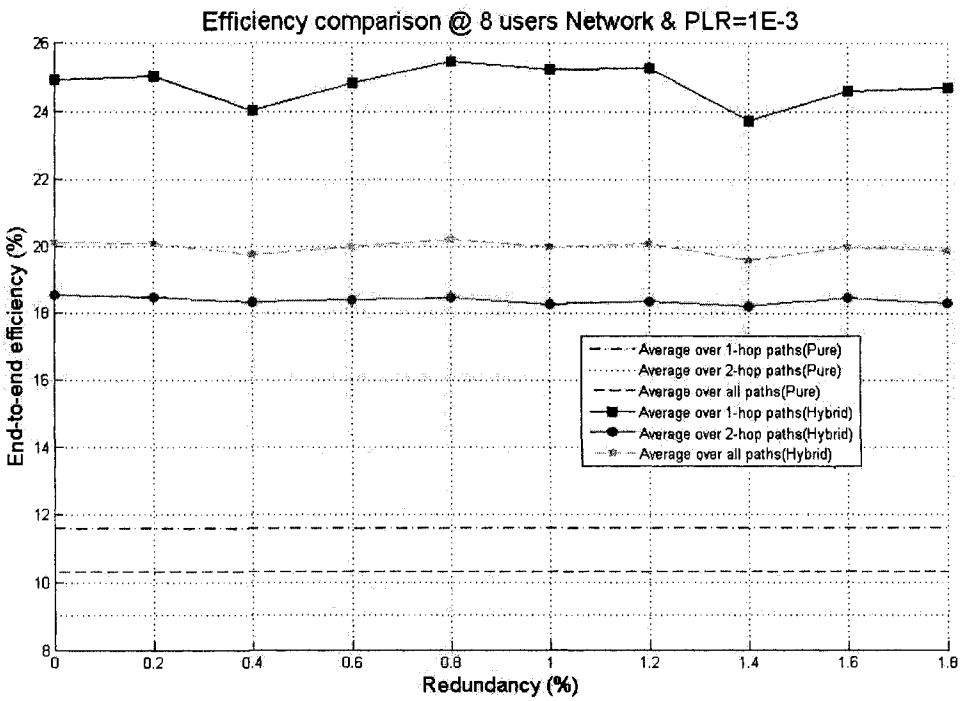

Figure 3.28: Efficiency comparison between hybrid ARQ and pure ARQ

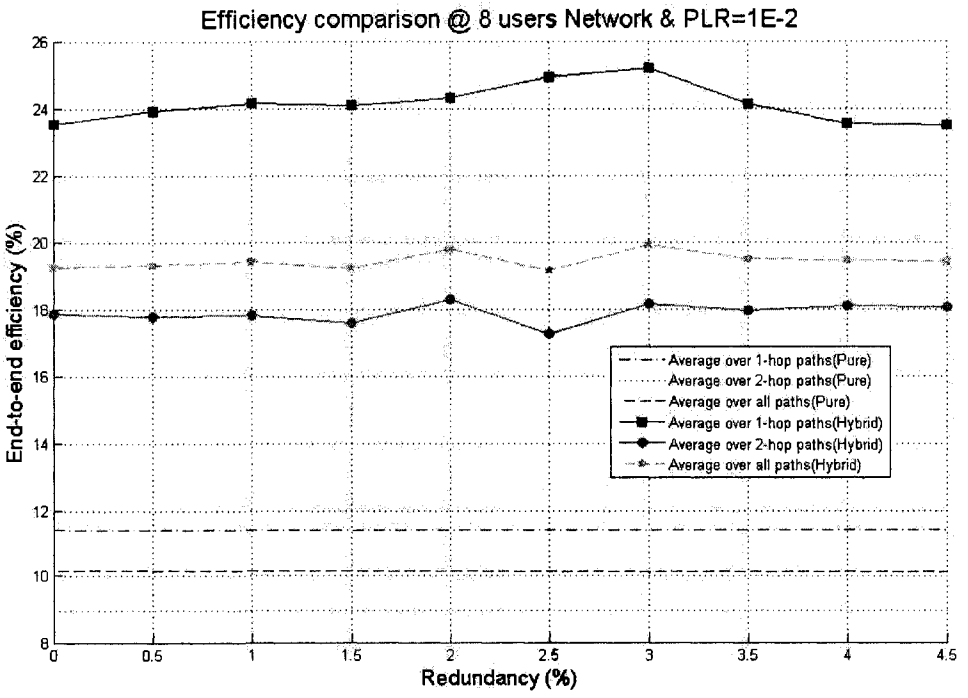

Figure 3.29: Efficiency comparison between hybrid ARQ and pure ARQ

Another key point the figures are showing is that, the minimum redundancy required for Raptor decoding to work varies significantly from path to path, that is, it greatly depends on the length of paths (number of hops throughout the path), and the difference can be considerably huge, as in Table 3.1. In order to show this difference, in many figures, not only the average network performance is plotted, but also the average of each length of paths. This can be explicitly shown in figures with high PLR, such as Figure 3.25, 3.30 and 3.35.

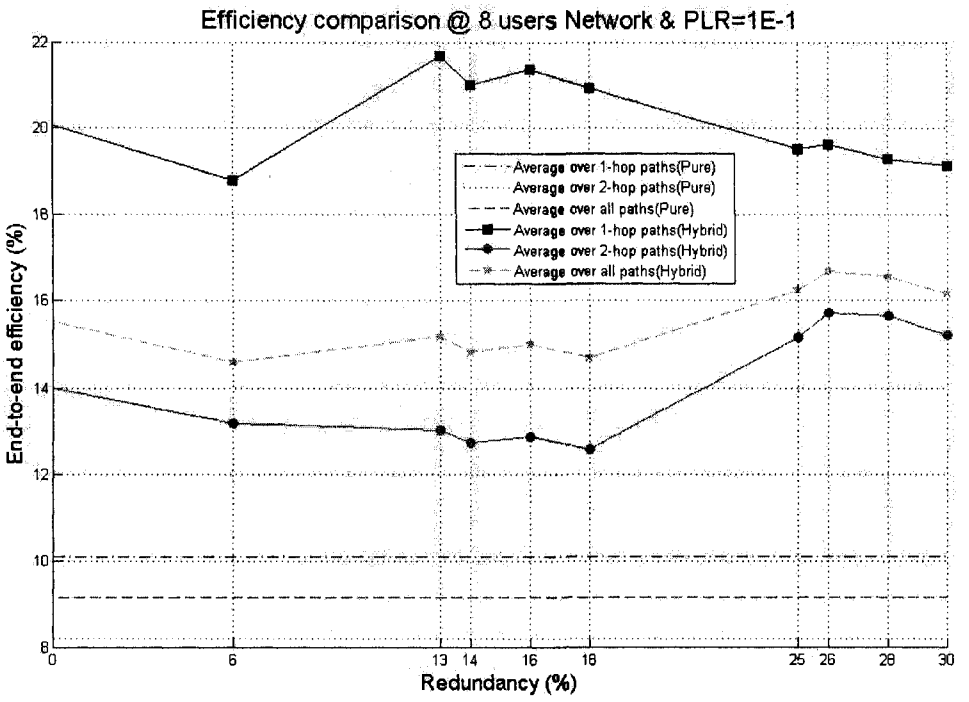

Figure 3.30: Efficiency comparison between hybrid ARQ and pure ARQ

Let us take 3.30 as an example, in Figure 3.30, the curve with square notation on the top represents the average over all single paths. First, at 6% redundancy, it goes down, because it has not reached its minimum required redundancy R_m, which should be around 11.1% according to Table 3.1. Then at 13% redundancy, it goes up, gets to the maximum

and gradually goes down afterwards. The performance of paths with 2 hops is represented by the curve with circle notation on the bottom. Similarly, it first goes down and keeps going down until it reaches its R_m, which again according to Table 3.1 is about 23.5%. It experiences a great raise at the point 25% and gets to a maximum at 26% redundancy and gradually goes down afterwards. The dashed curve with star notation in the middle is the average over all paths, so it reflects the effect of combination: it first goes down, and goes up a little bit after 13% redundancy, that makes sense because all the paths with single hop start to benefit from Raptor decoding; then there is a big raise after 25% redundancy and a maximum at 26%, then gradually goes down afterwards.

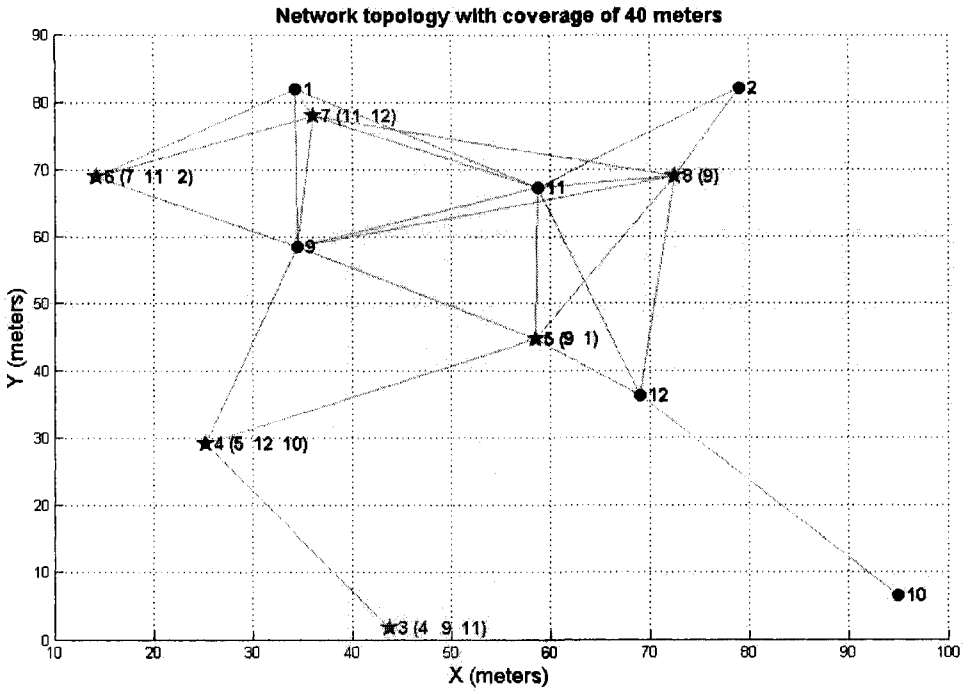

Figure 3.31: A 12-user network

Figure 3.31 shows the network topology of a 12-user network, with the node coverage radius of 40 meters. It has six source-to-destination paths: 6 – 7 – 11 – 2, 7 – 11

– 12, 4 – 5 – 12 – 10, 3 – 4 – 9 – 11, 5 – 9 – 1 and 8 – 9. Figure 3.32 – 3.35 show the performance comparisons between this hybrid ARQ and pure ARQ under different channel conditions, still, that the improvement is great. It can be noticed that the performance continues to decreases further compared with 8-user network, caused by the same reason explained before.

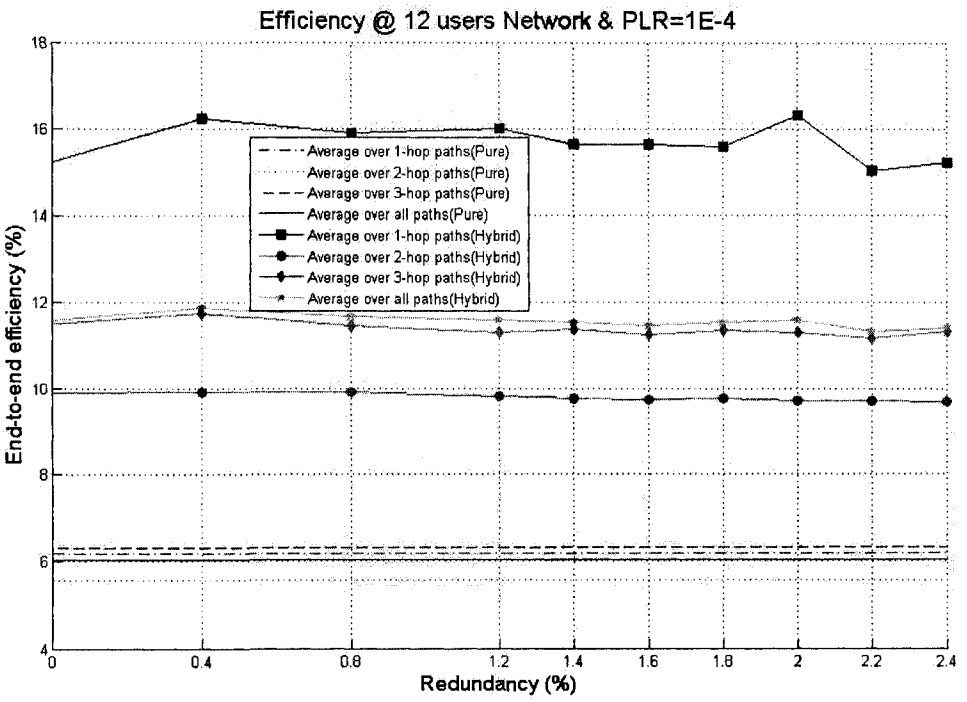

Figure 3.32: Efficiency comparison between hybrid ARQ and pure ARQ

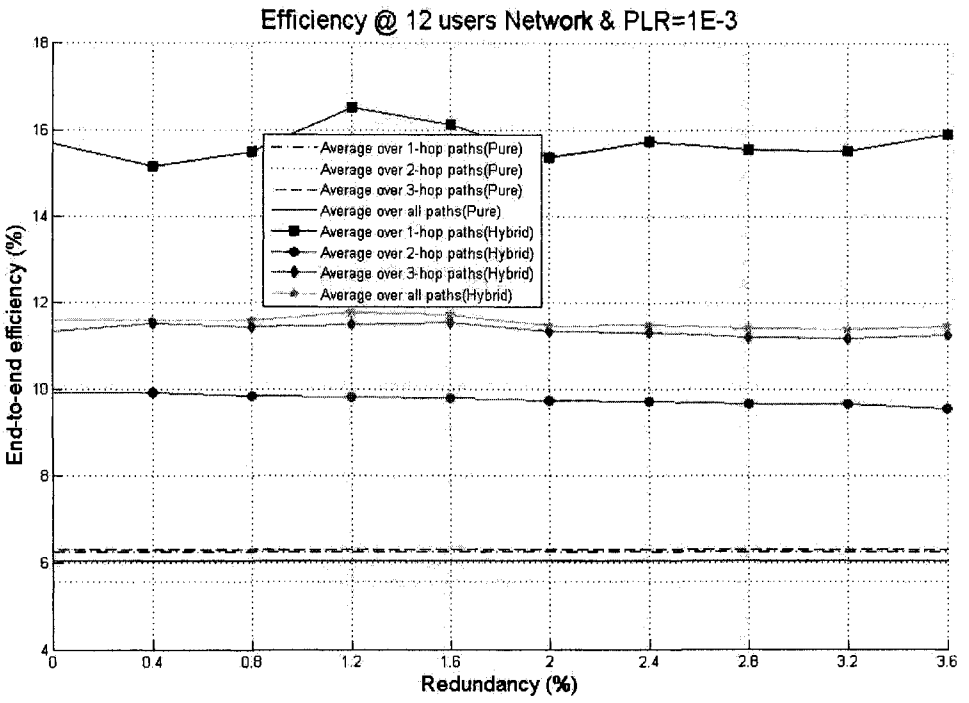

Figure 3.33: Efficiency comparison between hybrid ARQ and pure ARQ

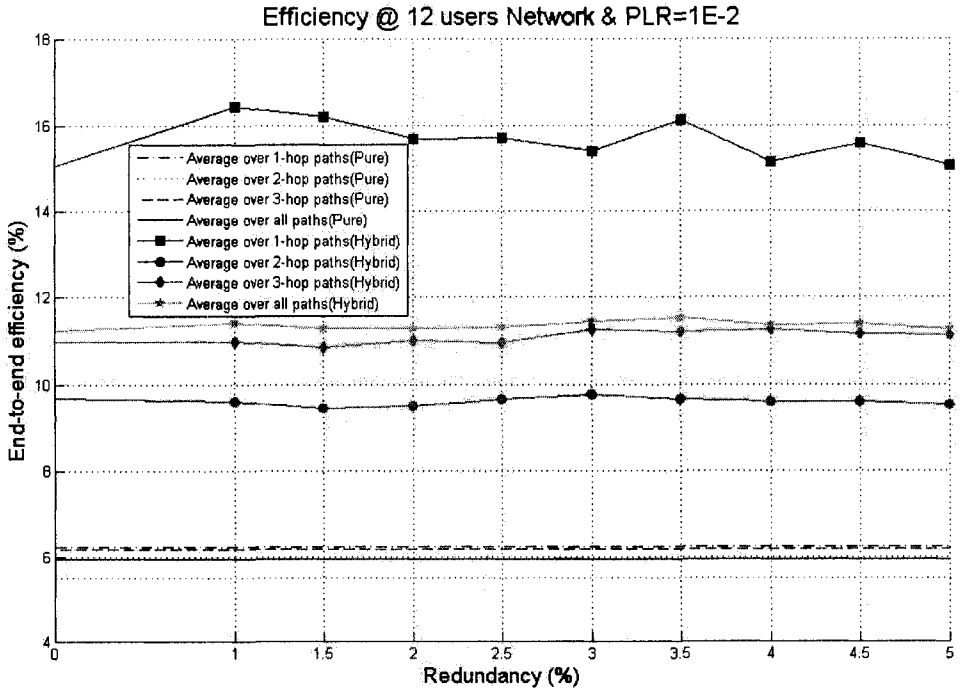

Figure 3.34: Efficiency comparison between hybrid ARQ and pure ARQ

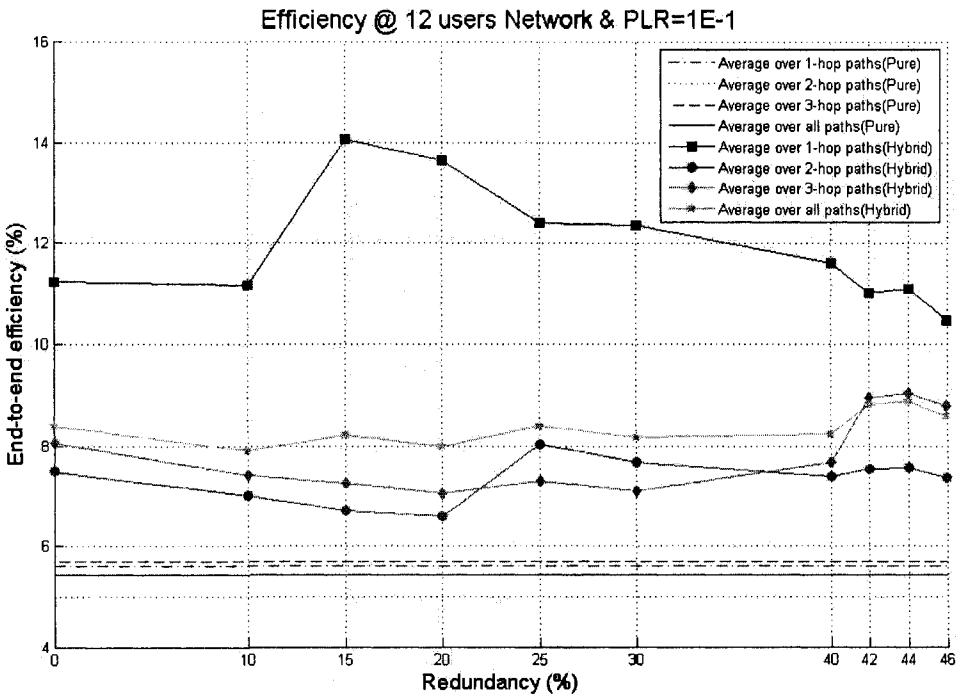

Figure 3.35: Efficiency comparison between hybrid ARQ and pure ARQ

Figure 3.36 – 3.38 are plotted in order to show the comparison from another angle. It can be shown that the performance goes down when the PLR becomes high, which makes sense, as the channel quality is getting worse. It should be noted that, for the curves of hybrid ARQ, each value represents the maximum value of a certain case and the dashed curve with star notation is not the average of the other two curves, it is the maximum of average over all paths. Simply speaking, it is the maximum of averages, not the average of maximums.

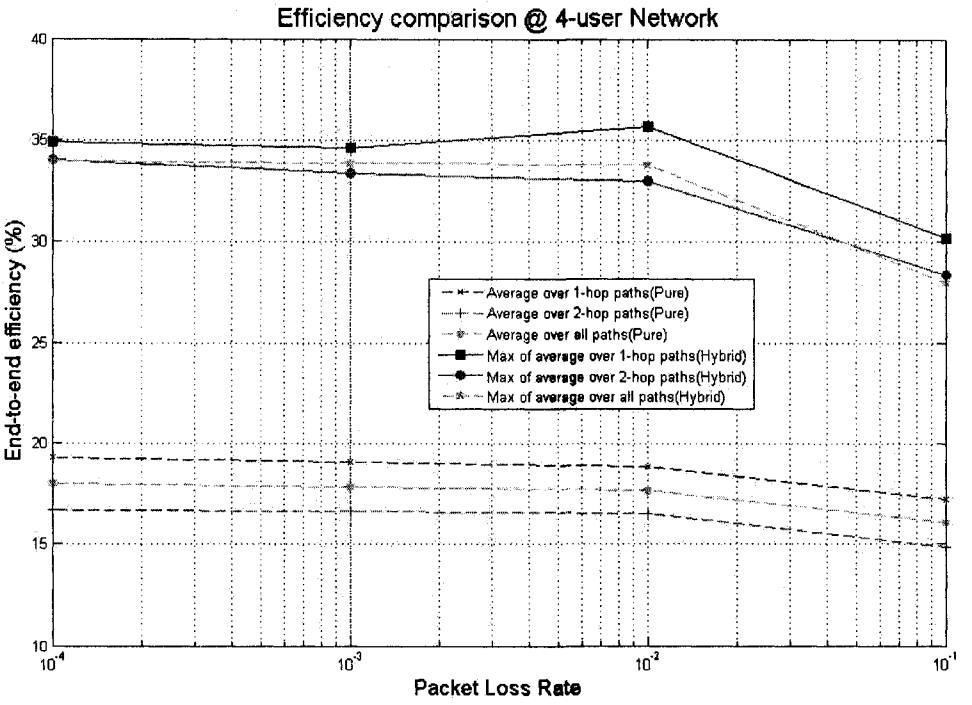

Figure 3.36: Maximum efficiency comparison under different PLRs

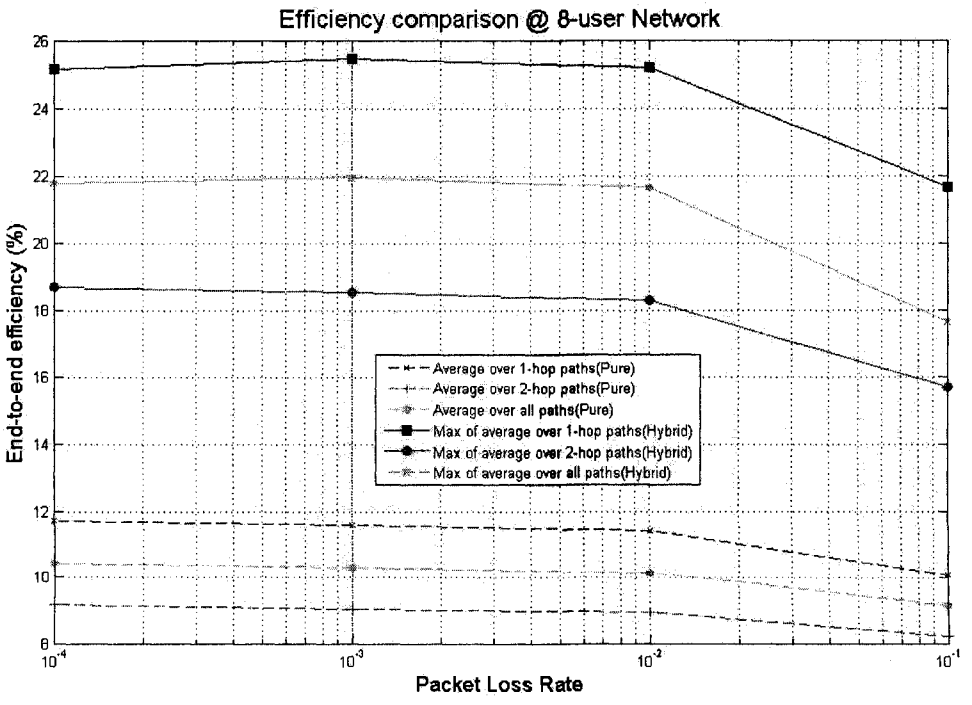

Figure 3.37: Maximum efficiency comparison under different PLRs

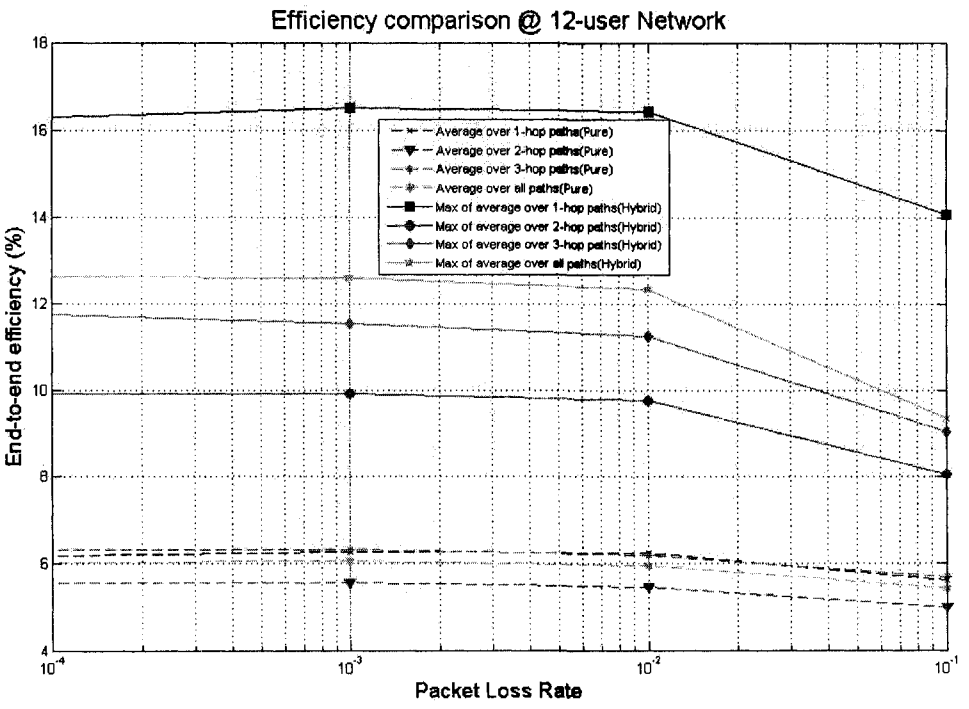

Figure 3.38: Maximum efficiency comparison under different PLRs

Contribution 2: Adaptive redundancy design helps to achieve the best average network performance and to improve the redundancy efficiency.

The adaptive redundancy allows us to dynamically assign each path a certain amount of redundancy that is best suited for this path. As shown in the above Table 3.1, the redundancy required to achieve the best performance varies significantly with the length of the path, i.e. the number of hops throughout the source-to-destination path. Consequently, a uniform redundancy is not suitable for all paths. Especially in the case of high PLR, when it is good for one path, it then either dose not reach the minimum required redundancy for other paths, or it is too high for other paths, becoming useless extra redundancy, which also degrades the performance. With adaptive redundancy, it would be possible for every path to reach its own maximum performance, because every

one of them is using the best suited redundancy. Therefore, it provides the best average network performance. This effect is more noticeable in high PLR, and fades away when PLR becomes low, see Figure 3.42.

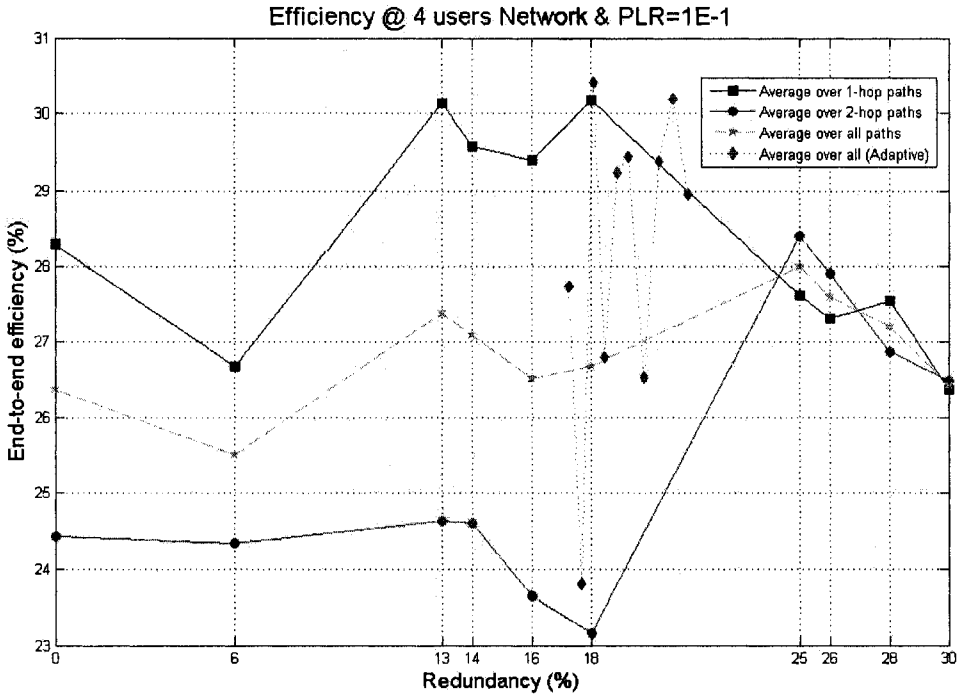

Figure 3.39: Efficiency comparison between adaptive and uniform redundancy

Figure 3.39 – 3.42 show the performance comparisons between adaptive and uniform redundancy, one can see noticeable improvements in these figures. For example, in Figure 3.40, the dotted curve with diamond notation is the performance with adaptive redundancy design; it is obvious that all the sampled points from this curve are above the dashed curve with star notation, representing the average performance with uniform redundancy, namely they have better performance. In addition, the performance with adaptive redundancy not only has a higher maximum, but also reaches its maximum with less average redundancy.

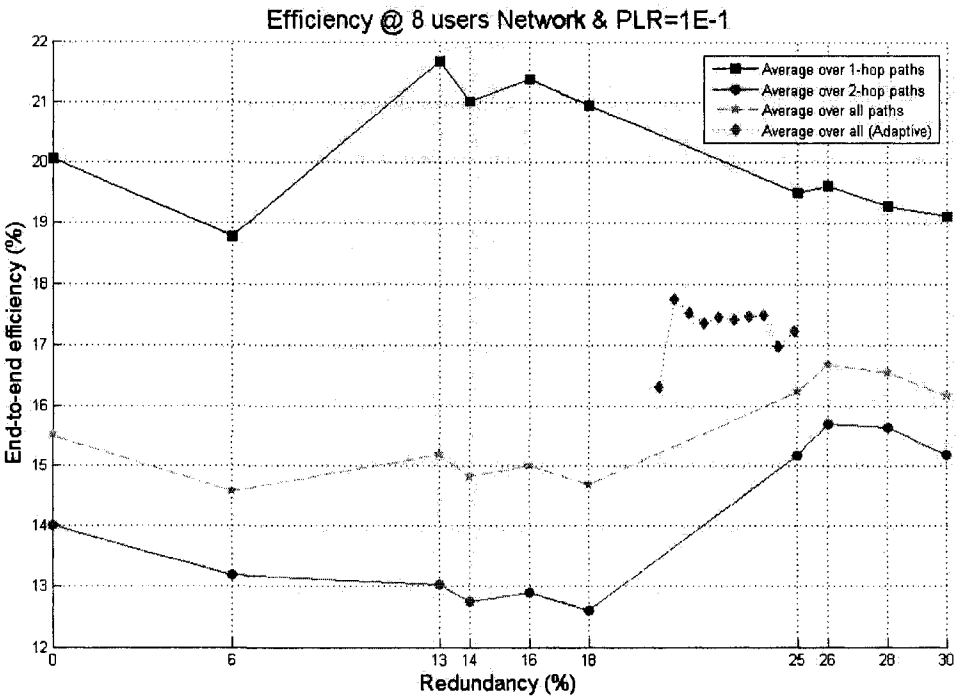

Figure 3.40: Efficiency comparison between adaptive and uniform redundancy

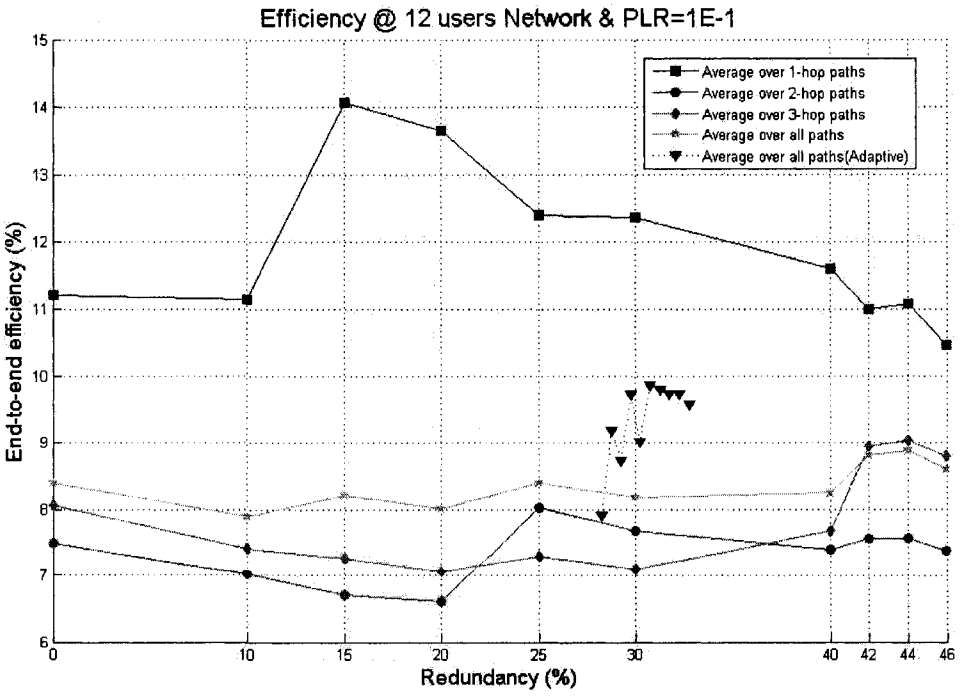

Figure 3.41: Efficiency comparison between adaptive and uniform redundancy

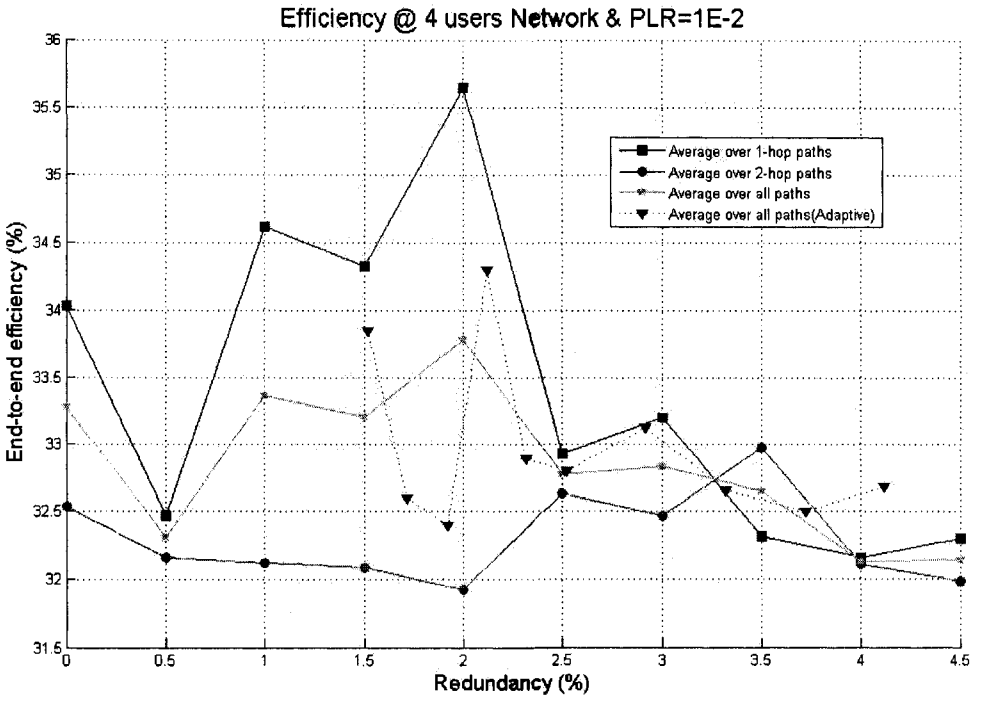

Figure 3.42: Efficiency comparison between adaptive and uniform redundancy

Figure 3.42 shows the comparison at a lower PLR, as mentioned above, the effect of improvement decreases. It can be seen that only a few points are above the average performance without adaptive redundancy. The reason is that at low PLR, the difference among the values of minimum required redundancy for different lengths of path is not big enough. In other words, the gaps are small so that there is no much waste of redundancy.

Finally, a little trick is that the selective decoding at low redundancy and low packet loss rate saves Raptor decoding time and improves the efficiency. Thanks to systematic Raptor coding technique, it is possible to receive original source packets as output of encoder. Hence, in the case of low PLR, for a certain block size, it is very likely to successfully receive all the original source packets in a block, while still losing some repair packets. In this case, if we still perform the regular decoding process, then there

would be a certain failure probability, so this technique skips over the decoding process and continue on the next block, without regard to other repair packets. This effect can only be found obvious in low redundancy and low PLR environment, and it fades away with the redundancy and PLR increases. In particular, to make sure all the original packets will succeed, a simple sense is that as long as $B \times P < 1$, namely, P is less than $\frac{1}{B}$, this technique will make a little improvement on performance, otherwise it makes no difference. Therefore, in our case, for B=1024 and P=10^{-4}, we have $B \times P \approx 0.1 < 1$, so the technique is suitable for this case. It should be found effective in practice, in terms of saving the decoding time, which is not considered in this simulation, so it is not proved in this thesis.

Chapter 4

Comparison and Conclusion

4.1 Comparison

It is difficult to directly compare our results with others, because many factors are considered in our simulation. There are mainly three reasons for this:

First, research from most papers does not include the impact of channel contention, which makes a strong impact on the performance. Since in a wireless network, wireless users transmit data in a manner of broadcast due to the nature of radio signal, so they share a common media resource, and there is also channel interference. As a result, they cannot access the channel whenever they want; instead they have to access the channel according to a certain channel access control scheme. Therefore channel contention significantly degrades the performance of all the users, and becomes even worse when the number of wireless users increases in the network.

Second, research results from most papers usually only study the performance of single hop. In our simulation, we study the end-to-end performance in a network, that is to say, there are many paths with multiple hops. This simply increases the total time needed to deliver a certain amount of data, thus decreases the performance of an end-to-end path. Additionally, as explained before, every node within the same path also

suffers from the channel contention. This further degrades the performance of each path.

Third, most of previous research generally evaluates network performance in terms of throughput, which is usually defined as the gross bit rate transferred physically, that means every physically successfully delivered bit is taken into account, including protocol overhead, retransmitted data packets etc. Throughput is typically measured at a reference point below the network layer and above the physical layer. However, in our case, performance is measured in terms of goodput, which is application layer level throughput, i.e. from end-user's point of view, the number of useful bits per unit of time forwarded by the network from a source to a destination. In other words, goodput is the net achieved average bit rate delivered to application layer, excluding all lower protocols overheads, data retransmissions. Therefore, goodput is usually lower than throughput.

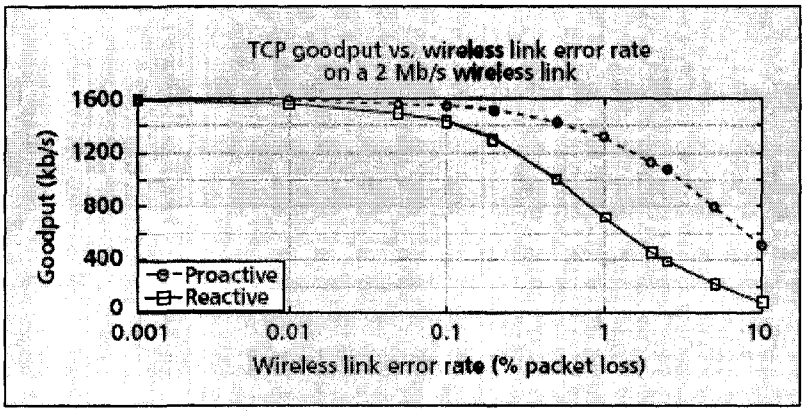

Figure 4.1: Performance of a typical proactive scheme (TCP-Jersey) and a typical reactive scheme (TCP-Reno) in the wireless environment [50]

However, it is still worth to compare our simulation results with others' in a certain way. Although it is hard to find a result based on the exact same situation as ours, we can still find some results with similar conditions to compare with and see the improvement

in a certain way. The efficiency η in our results is like the normalized goodput, so multiply it with a certain channel capacity would give us a certain goodput which we can compare with others, below are a few examples.

Figure 4.1 is a result from [50], it shows the TCP goodput under different channel qualities on a 2Mbps wireless channel. In Figure 4.1, for PLR=1E-1, a typical proactive scheme TCP – Jersey gets a goodput about 500 Kbps, and a typical reactive scheme TCP – Reno gets a goodput less than 100 Kbps. In our case, from Figure 3.25 we can see that under the same channel quality (PLR=1E-1), we can achieve a goodput more than 2000*30% = 600 Kbps with single hop, and a goodput more than 2000*28% = 560 Kbps with two hops. Both paths achieve a better performance than either of the two TCP enhancements, even with the channel contention and multi-hop effects taken into account.

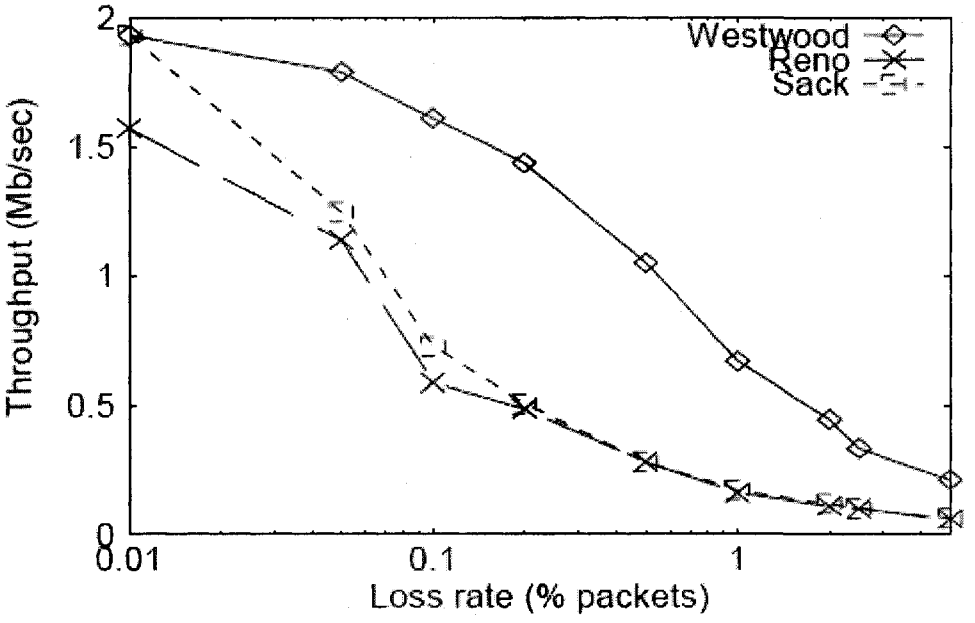

Figure 4.2: Throughput vs. error rate of the wireless link [20]

Another example is in Figure 4.2 from [20], it is under a wireless channel of 2 Mbps,

and it is measured in terms of throughput. In Figure 4.2, for PLR=1E-2, TCP Reno and TCP SACK have the similar throughput about 0.25 Mbps, and TCP Westwood has a throughput of 0.7 Mbps. In our case, from Figure 3.24 we can see that under the same channel quality (PLR=1E-2), we can achieve a goodput at least 2*35.5% = 0.71 Mbps with single hop, and a goodput about 2*33% = 0.66 Mbps with two hops. Both paths achieve a better performance than all of the TCP enhancements except the two-hop performance is slightly lower than TCP Westwood, even with the channel contention and multi-hop effects taken into account, and ignoring the difference between goodput and throughput.

4.2 Conclusion

This thesis first presented a general introduction of wireless LANs and wireless ad hoc networks, and discussed some developing wireless LAN techniques. We discussed TCP fundamentals and some basic problems when TCP is used in wireless environments, and then some proposed improvements focusing on these problems were suggested. We included a review of FEC, followed by a brief description of digital fountain and raptor codes, state-of-the-art concepts and the most advanced FEC techniques. Then we introduce a new HARQ technique for reliable and efficient packets transfer in wireless environment. In contrast to most HARQ techniques proposed so far, which usually employ a byte level FEC combined with ARQ, in our system, we mostly use packet level FEC for the data transfer, in conjunction with ARQ to compensate for the little

inefficiency. It is similar to type II HARQ, except the FEC is applied in a higher layer, i.e. the application layer. We also introduce the notion of adaptive redundancy networks which helps to achieve better average network performance and to further improve the redundancy efficiency. We also use selective decoding at low packet loss rate improves efficiency, which is useful in practice. Finally, although it is hard to find a result based on the exact same situation as ours, we still find some results with similar conditions to compare with and see the improvement in a certain way.

4.3 Future works

In the near future, a more complicated simulation system is expected so that we can simulate all kinds of conditions more accurately, so more concerns should be taken care of. In particular, for example, instead of using a simple equation based model, we can develop a more complex way to better simulate the efficiency of Raptor coding, including the effects of encoding and decoding processing times. In addition, we may look forward to develop a better way to balance the trade-off between FEC and ARQ techniques, so that we can make the most use of both and achieve better performance.

References

[1] Http://en.wikipedia.org/wiki/Wi-Fi

[2] ETSI Normalization Committee, BRAN, "HIPERLAN Type 2; Physical (PHY) Layer," doc. RTS0023003-R2, Feb. 2001.

[3] Http://en.wikipedia.org/wiki/802.11

[4] P. Roshan and J. Leary, "802.11 Wireless LAN Fundamentals" Cisco Press, Dec. 23, 2003, 1-58705-077-3

[5] Http://www.dailywireless.org/2007/05/16/new-logo-for-80211n/

[6] R. Ramanathan and J. Redi, "A brief overview of ad hoc networks: challenges and direction" IEEE Communications Magazine, 50th Anniversary Commemorative Issue/May 2002.

[7] K. Weniger and M. Zitterbart, "Mobile ad hoc networks - current approaches and future directions" IEEE Network, Volume 18, Issue 4, July-Aug. 2004 Page(s):6–11

[8] I. F. Akyildiz and X. Wang, "A Survey on Wireless Mesh Networks," IEEE Commun. Mag., vol. 43, no. 9, Sept. 2005, pp. S23–S30.

[9] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," Comp. Networks, 2002.

[10] Http://en.wikipedia.org/wiki/Image:WSN.svg

[11] J. Postel. RFC 793: Transmission control protocol, Sept. 1981.

[12] A. L. Garcia and I. Widjaja, 'Communication Networks: Fundamental Concepts and

Key Architectures', 2004, pp. 571.

[13] James F. Kurose and Keith W. Ross, 'Computer Networking: A Top-down Approach Featuring the Internet', 2nd Edition, 2000, Fig. 3.7-7.

[14] J. Liu and S. Singh, "ATCP: TCP for Mobile Ad Hoc Networks," IEEE JSAC, vol. 19, no. 7, pp. 1300–1315, 2001.

[15] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks," ACM Wireless Networks, vol. 1, no. 4, Nov. 1995, pp. 469–481.

[16] F. Sun and V. L. Soung C. Liew, "Design of SNACK Mechanism for Wireless TCP with New Snoop," IEEE WCNC, vol. 5, no. 1, Mar. 2004 pp. 1046–51.

[17] N. Vaidya and M. Mehta, "Delayed Duplicate Acknowledgments: A TCP-Unaware Approach to Improve Performance of TCP over Wireless," Texas A&M University, Tech. Report 99 – 003, Feb. 1999.

[18] L. Brakmo and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," IEEE JSAC, vol. 13, no. 8, Oct. 1995, pp. 1465-80.

[19] C. P. Fu and S. C. Liew, "TCP Veno: TCP Enhancement for Transmission over Wireless Access Networks," IEEE JSAC, vol. 21, no. 2, Feb. 2003, pp. 216–28.

[20] C. Casetti et al., "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," ACM Mobicom, July 2001, pp. 287–97.

[21] METZNER, J. J., and CHANG, D.: 'Efficient selective repeat ARQ strategies for very noisy and fluctuating channels', IEEE Trans., 1985, COM-33. pp. 409 - 415

[22] A. Chockalingam, M. Zorzi and V. Tralli, "Wireless TCP performance with link layer FEC/ARQ," in Proc. IEEE ICC'99, Vancouver, BC, June 1999, pp. 1212–1216.

[23] C. F. Chiasserini and M. Meo, "A reconfigurable protocol setting to improve TCP over wireless" IEEE Trans. Veh. Technol., vol. 51, no. 6, pp. 1608–1620, Nov. 2002.

[24] R. G. Mukhtar, S. Hanly, M. Zukerman, and F. Cameron, "A model for the performance evaluation of packet transmissions using Type-II hybrid ARQ over a correlated error channel," Wireless Networks, vol. 10, no. 1, pp. 7–16, Jan. 2004.

[25] A. Ghosh et al.,"Broadband wireless access with WiMax/802.16: current performance benchmarks and future potential," IEEE Commun. Mag., vol. 43, no. 2, pp. 129–136, Feb. 2005.

[26] Forney, G.D. and Costello, D.J., "Channel Coding: The Road to Channel Capacity" Proceedings of the IEEE Volume 95, Issue 6, June 2007 Page(s):1150 - 1177

[27] W. W. Peterson, Error-Correcting Codes. Cambridge, MA: MIT Press, 1961.

[28] E. R. Berlekamp, Algebraic Coding Theory. New York: McGraw-Hill, 1968.

[29] S. Lin, An Introduction to Error-Correcting Codes. Englewood Cliffs, NJ: Prentice-Hall, 1970.

[30] W. W. Peterson and E. J. Weldon, Jr., Error-Correcting Codes. Cambridge, MA: MIT Press, 1972.

[31] F. J. MacWilliams and N. J. A. Sloane, The Theory of Error-Correcting Codes. New York: Elsevier, 1977.

[32] R. E. Blahut, Theory and Practice of Error Correcting Codes. Reading, MA:

Addison-Wesley, 1983.

[33] P. Elias, "Coding for noisy channels", IRE Conv. Rec., pt. 4, pp. 37–46, Mar. 1955.

[34] J. M. Wozencraft and B. Reiffen, Sequential Decoding. Cambridge, MA: MIT Press, 1961.

[35] R. M. Fano, "A heuristic discussion of probabilistic decoding", IEEE Trans. Inform. Theory, vol. IT-9, pp. 64–74, Jan. 1963.

[36] J. A. Heller and I. M. Jacobs, "Viterbi decoding for satellite and space communication", IEEE Trans. Commun. Tech., vol. COM-19, pp. 835–848, Oct. 1971.

[37] Revival of sequential decoding workshop, Munich Univ. of Technology, J. Hagenauer, D. Costello, general chairs, Munich, Germany, Jun. 2006.

[38] R. G. Gallager. Low-density parity check codes. PhD thesis, MIT, 1963. Manuscript published by MIT Press.

[39] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields", J. SIAM, vol. 8, pp. 300–304, Jun. 1960.

[40] M. Watson, "Application Layer Forward Error Correction, Summary of Simulation Results" DVB TM-CBMS1397, August 2005.

[41] J. W. Byers, M. Luby, and W. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," IEEE J. Select. Areas Commun., vol. 20, pp. 1528–1540, 2002.

[42] R. Palanki and J. Yedidia, "Rateless codes on noisy channels," in Proc. Int. Symp.

Inform. Theory, 2004, p. 37.

[43] J. Castura and Y. Mao, "Rateless coding over fading channels," IEEE Commun. Lett., vol. 10, pp. 46–48, 2006.

[44] O. Etasami, M. Molkaraie, and A. Shokrollahi, "Raptor codes on symmetric channels," in Proc. Int. Symp. Inform. Theory, 2004, p. 38.

[45] H. Jenkac and T. Stockhammer, "Asynchronous media streaming over wireless broadcast channels," in Proc. IEEE Int. Conf. Multimedia Expo., 2005, pp. 1318–1321.

[46] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman. Efficient erasure correcting codes. IEEE Transactions on Information Theory, 47(2):569-584, February 2001.

[47] M. Luby. "LT codes" In Proc. of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 271-282, 2002.

[48] A. Shokrollahi. "Raptor codes" IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 52, NO. 6, JUNE 2006 2551

[49] M. Luby, T. Gasiba, T. Stockhammer and M. Watson 'Reliable Multimedia Download Delivery in Cellular Broadcast Networks' IEEE TRANSACTIONS ON BROADCASTING, VOL. 53, NO. 1, MARCH 2007

[50] Y. Tian, K. Xu and N. Ansari, 'TCP in wireless environments: problems and solutions', Communications Magazine, IEEE Volume 43, Issue 3, March 2005.

Appendix

A.1 A basic description of Dijkstra's algorithm

1. Create a distance list, a previous vertex list, a visited list, and a current vertex.
2. All the values in the distance list are set to infinity except the starting vertex which is set to zero.
3. All values in visited list are set to false.
4. All values in the previous list are set to a special value signifying that they are undefined, such as null.
5. Current vertex is set as the starting vertex.
6. Mark the current vertex as visited.
7. Update distance and previous lists based on those vertices which can be immediately reached from the current vertex.
8. Update the current vertex to the unvisited vertex that can be reached by the shortest path from the starting vertex.
9. Repeat (from step 6) until all nodes are visited.

A.2 Source code selections

A.2.1 Network Initialization and Routing

%% All kinds of parameter set-up.

```
disp('          ******************Begin*********************');

U=8; % Number of source-to-destination pairs <<<-------U=[2 4 6 8 10]

nNodes=2*U; % Number of nodes

area=100; % The area

radius=40; % Coverage radius of a node[10-100]

node.cover=zeros(nNodes,nNodes); % Nodes in coverage

% node.v=rand(nNodes,1)*nNodes; % User speed ranges from 0 to 20 m/s(72 km/h)

% node.theta=rand(nNodes,1)*pi*2; % random movement directions

%% Initial environmental setups.

newplot

hold on

grid on

xlabel('X (meters)','fontsize',14)

ylabel('Y (meters)','fontsize',14)

title('\bf{Network topology with coverage of 40 meters}','fontsize',14)

% Generate users randomly distributed in a area*area square

% Randomly match nNodes users up into pairs

node.pos=rand(nNodes,2)*area;

aNode=1:nNodes;

x=nNodes;

tx=zeros(1,U);
```

```
rx=zeros(1,U);
for i=1:U
    n=ceil(rand*x);
    tx(i)=aNode(n);
    aNode(n)=[];
    x=x-1;
    n=ceil(rand*x);
    rx(i)=aNode(n);
    aNode(n)=[];
    x=x-1;
end
pairs=[tx;rx];
%% A check-up, and connect the network.
% Check the network connectivity.
iso_counter=zeros(nNodes,1);
DD=zeros(nNodes,nNodes);
for i=1:nNodes
    for j=1:nNodes
        if i~=j
            DD(i,j)=nDist(node.pos(i,1),node.pos(i,2),...
                node.pos(j,1),node.pos(j,2));
```

```
            if DD(i,j) < radius
                iso_counter(i)=iso_counter(i)+1;
                m=[node.pos(i,:);node.pos(j,:)];
                plot(m(:,1),m(:,2),'g','linewidth',1)
                node.cover(i,j)=j;
                node.cover(j,i)=i;
            end
        end
    end
    if ~iso_counter(i)
        disp(sprintf('Node %d is isolated',i))
        disp(node.pos(i,:))
    end
end
if any(iso_counter==0)
    disp(sprintf('Number of isolated nodes is %d', sum(iso_counter==0)))
else
    disp('No isolated nodes, ^_^')
end
%% Transform the parameters to the inputs of function so as to make it work
nodes=[(1:nNodes)' node.pos];
```

```
routes=[];
for i=1:nNodes-1
    for j=i+1:nNodes
        if node.cover(i,j)
            routes=[routes;[i j]];
        end
    end
end
Q=size(routes);
nSeg=Q(1); %Number of undirectional segments
% Another way to get nSeg:
% nSeg=(sum(sum(node.cover~=zeros(nNodes,nNodes)))-nNodes)/2;
segments=[(1:nSeg)' routes];
%% Apply dijkstra to get the shortest paths
distances=[pairs;zeros(1,U)]';
paths={[]};
for i=1:U
    [d,p]=dijkstra(nodes,segments,pairs(1,i),pairs(2,i));
    distances(i,3)=d;
    paths{i,1}=p;
    disp(['Path ' num2str(i) ':     ' num2str(paths{i})])
```

```
end
for i=1:nNodes
    if any(i==tx)
        y=find(i==tx);
        text(node.pos(i,1),node.pos(i,2),['    ' num2str(i) ' (' ...
            num2str(paths{y}(2:end)) ')'],'color','r',...
            'fontsize',12,'fontweight','b')
        plot(node.pos(i,1),node.pos(i,2),'p',...
        'markersize',12,'markerfacecolor','r','markeredgecolor','r')
    else
        text(node.pos(i,1), node.pos(i,2), ['    ' num2str(i)],...
            'color','b','fontsize',12,'fontweight','b')
        plot(node.pos(i,1),node.pos(i,2),'o',...
        'markersize',8,'markerfacecolor','b','markeredgecolor','b')
    end
end
hold off
```

A.2.2 The Hybrid ARQ System

```
%% All kinds of parameters set-up.
tic
```

```
file=.5*8*2^20; % File size is 512 Kbytes

nBits=512*8; % 512 bytes per Packet

nPackets=ceil(file/nBits); % Number of packets

blk_size=1024; % Number of packets per blocks

PLR=.0001; % The Packet Loss Rate

% PLR=logspace(-4,-1,4);

lplr=length(PLR);

% rp=0:.002:.018; % For 1e-4

rp=0:.001:.009; % For 1e-4

% rp=[0 .004 .008 .012 .014 .016 .018 .02 .022 .024]; % For 1e-4

% rp=[0 .004 .008 .012 .016 .02 .024 .028 .032 .036]; % For 1e-3

% rp=[0 .01 .015 .02 .025 .03 .035 .04 .045 .05]; % For 1e-2

% rp=[0 .1 .15 .2 .25 .3 .4 .42 .44 .46]; % For 1e-1

% rp=[0 .06 .13 .14 .16 .18 .25 .26 .28 .3]; % For 1e-1

REDUN=ceil(blk_size*rp); % Adjust Redundancy!!! <<<---

lrd=length(REDUN);

nBlocks=nPackets/blk_size;

bITER_PATH=zeros(lplr,U,lrd);

bBLK_FAIL=zeros(lplr,U,lrd);

bAPD=zeros(lplr,U,lrd);

bVAPD=zeros(lplr,U,lrd);
```

```
bEFF=zeros(lplr,U,lrd);
%% Main3...
for l=1:lrd
    redun=REDUN(l);
    nCoding=ones(1,U)*(blk_size+redun);
    ITER_PATH=zeros(lplr,U);
    BLK_FAIL=zeros(lplr,U);
    APD=zeros(lplr,U);
    VAPD=zeros(lplr,U);
    EFF=zeros(lplr,U);
    for k=1:lplr
        plr=PLR(k); % Packet Loss Rate [1e-3 to .3] <<<---
        %% Main
        iter=0;
        iter_path=zeros(1,U); % Counter of each path
        b_counter=zeros(1,U); % Block counter
        node.buffer=zeros(nNodes,U);
        pdec=zeros(U,nBlocks); % Probability of decodings
        pkt_delay={};
        pkt_start={}; % For packet delay record
        pkt_start2={}; % For ARQ packet start time
```

```
nh=zeros(1,U); % Number of hops on the path
for i=1:U
    node.buffer(tx(i),i)=nCoding(i); % Fountain code source
    pkt_delay{i}=[]; % For packet delay record
    pkt_start{i}=[];
    nh(i)=length(paths{i})-1;
end
pkt_fail=zeros(1,U); % Packet fail counter
blk_fail=zeros(1,U); % Block fail counter
rs=blk_size*ones(1,U); % Remaining Source packets
lsi=ones(1,U); % Last source packets indicator
fff=ones(1,U); % Point to delay
sss=ones(1,U); % To point the next gone packet in ARQ
gone=ones(1,U); % Gone indicateor
arq=zeros(1,U); % ARQ indicator
send_ack=zeros(nNodes,U);
wait_ack=zeros(nNodes,U);
while any(b_counter < nBlocks) % Received all the blocks?
    iter=iter+1; % Iteration counter
    % disp(['          ********* iter ' num2str(iter) ' *********'])
    transfer=zeros(1,nNodes); % Free all nodes to transfer
```

```
listen=zeros(1,nNodes); % Free all nodes to listen
receive=zeros(1,nNodes); % Free all nodes to receive
rand_order=get_order(nNodes); % Get random order
% disp(['Random Oder: ' num2str(rand_order)])
for i=1:nNodes % Virtual RTS/CTS begin...
    x=rand_order(i); % The node number to be polled
    z=[]; % To store node number which has data
    if receive(x) % Make sure it's not receiving
        % disp(['Node ' num2str(x) ' is receiving data, cannot transfer'])
    else % Not receiving...
        for j=1:U % Check out buffers to get 'z'
            if node.buffer(x,j) && rx(j)~=x % Having data
                nxt_hop=paths{j}(find(x==paths{j})+1); % Next hop
                if ~listen(nxt_hop) && ~transfer(nxt_hop) && ...
                        ~any(receive&node.cover(x,:))
                    % Next hop available & no interf.
                    if ~arq(j) % FEC?
                        z=[z;[j,nxt_hop]]; % All possible senders
                    elseif ~wait_ack(x,j) && ~send_ack(x,j)
                     % Not waiting for ACK and no need to ACK
                        z=[z;[j,nxt_hop]]; % All the possible senders
```

```
                else
%               disp(['Buffer of path ' num2str(j) ' on node ' ...
%               num2str(x) ' is waiting for ACK'])
                    end
                end
            end
        end
        y=size(z);
        for j=1:U
            if send_ack(x,j)
                nxt_hop=paths{j}(find(x==paths{j})-1); % Next hop
                if ~listen(nxt_hop) && ~transfer(nxt_hop) && ...
                        ~any(receive&node.cover(x,:))
                    z=[z;[j,nxt_hop]];
                end
            end
        end
        if isempty(z) % No data?
% disp([' Node ' num2str(x) ' has no packets or may cause interf.'])
        else % Having data and finally can transmit @@@
            transfer(x)=1; % Make itself 'transfer'
```

```
listen=listen+node.cover(x,:);
% Make all nodes in coverage 'listen'
q=ceil(rand*length(z(:,1)));
% Randomly choose one buffer !!!
receive(z(q,2))=1; % Make the next hop in receive status
pid=z(q,1); % Get the Path ID
if arq(pid) % ARQ mode
    if q > y(1) % ACK Tx
        send_ack(x,pid)=0; % ACK finished
        lst_hop=paths{pid}(find(x==paths{pid})-1);
        % Last hop
        wait_ack(lst_hop,pid)=0; % ACK finished
        % disp('    ACKed')
    else % Normal Tx
        if x == tx(pid) && gone(pid)
        % A source and first time
            pkt_start2{pid}(sss(pid))=iter-1;
            % Get the start time
            gone(pid)=0;
      % disp(['The ' num2str(sss(pid)) 'st packet sssss @
    '...      % num2str(iter-1) ' @ path ' num2str(pid)])
```

```
                end
                if rand > plr % Transmit success!
                    node.buffer(x,pid)=node.buffer(x,pid)-1;
                node.buffer(z(q,2),pid)=node.buffer(z(q,2),pid)+1;
                    if x == tx(pid) % A source
                        gone(pid)=1; % Indicate that it's gone
                        sss(pid)=sss(pid)+1;
                    end
                    if z(q,2) == rx(pid) % The last hop!
pkt_delay{pid}=[pkt_delay{pid} iter-pkt_start2{pid}(fff(pid))];
                        fff(pid)=fff(pid)+1;
                % disp(['The ' num2str(fff(pid)) 'st packet fffff @ '...
                % num2str(iter) ' @ path ' num2str(pid)])
                    end
% disp(['ARQ: Packet success from ' num2str(x) ' to ' ...
% num2str(z(q,2)) ' on the ' num2str(pid) 'th path @@@@@@'])
                else % Transmit fail!
                % disp(['ARQ: Packet fail from ' num2str(x) ' to ' ...
                % num2str(z(q,2)) ' on the ' num2str(pid) 'th path ###'])
                end
                send_ack(z(q,2),pid)=1; % Have to ACK!
```

```
            wait_ack(x,pid)=1; % Have to wait!
            % disp('       ACKing')
        end
    else % Raptor mode
        xsn=find(x==paths{pid});
        % X's sequence number in this path
        if x == tx(pid) % This node is a source
            pkt_start{pid}=[pkt_start{pid} iter-1];
            % Get the start time
        end
        node.buffer(x,pid)=node.buffer(x,pid)-1;
        if rand > plr % Transmit success!
          node.buffer(z(q,2),pid)=node.buffer(z(q,2),pid)+1;
            if z(q,2)==rx(pid)
                % Next hop is a destination node
     pkt_delay{pid}=[pkt_delay{pid} iter-pkt_start{pid}(1)];
                pkt_start{pid}(1)=[];
            end
     % disp(['Raptor: Packet success from ' num2str(x) ' to ' ...
     % num2str(z(q,2)) ' on the ' num2str(pid) 'th path @@@'])
        else % Transmit fail!
```

```
                    % disp(['Raptor: Packet fail from ' num2str(x) ' to ' ...
                    % num2str(z(q,2)) ' on the ' num2str(pid) 'th path ###'])
                        pkt_fail(pid)=pkt_fail(pid)+1;
                        pkt_start{pid}(1)=[];
                    end
                    if xsn == lsi(pid) % Last source?
                        rs(pid)=rs(pid)-1;
                        while ~rs(pid) && lsi(pid)<length(paths{pid})
                            nxt_hop=paths{pid}(lsi(pid)+1); % Next hop
                            rs(pid)=node.buffer(nxt_hop,pid);
                            lsi(pid)=lsi(pid)+1; % Next
                        end
                    end
                end
            end
        end
    end % Virtual RTS/CTS finished!
    for i=1:U % Check if ready for ARQ or Raptor decoding
        if b_counter(i) < nBlocks % Not finished yet?
            if arq(i) % ARQ part!!!!!
                if node.buffer(rx(i),i) == blk_size-rs(i) && ...
```

```
            ~sum(wait_ack(:,i))
        arq(i)=0;
        b_counter(i)=b_counter(i)+1;
        node.buffer(rx(i),i)=0;
        node.buffer(tx(i),i)=nCoding(i);
        pkt_fail(i)=0;
        beep; % Alert!
        rs(i)=blk_size; % Remaining Source packets
        lsi(i)=1; % Last source packets indicator
        fff(i)=1; % Point to delay
        sss(i)=1;
        gone(i)=1;
        disp(['ARQ: The ' num2str(b_counter(i)) ...
            'th block succeeded @ Rx ' num2str(rx(i))])
disp(['*** iter ' num2str(iter) ' @ PLR ' ...
num2str(PLR(k)) ' & REDUN ' num2str(REDUN(l)) ' ***'])
 end
 elseif sum(node.buffer(:,i))==node.buffer(rx(i),i)
  % Raptor decoding
 disp(['Raptor: Got ' num2str(node.buffer(rx(i),i)) ...
      ' packets, decoding @ Rx ' num2str(rx(i)) '...'])
```

```
    Pd=.85*.567^(node.buffer(rx(i),i)-blk_size);
     % Failure probability
    pdec(i,b_counter(i)+1)=(1-Pd)*100;
    if rs(i) == blk_size % No original packets lost
% if 0
        b_counter(i)=b_counter(i)+1;
        node.buffer(tx(i),i)=nCoding(i);
        beep % Alert!
        rs(i)=blk_size; % Remaining Source packets
        lsi(i)=1; % Last source packets indicator
        fff(i)=1; % Point to delay
        sss(i)=1;
        gone(i)=1;
        disp(['Directly: The ' num2str(b_counter(i)) ...
            'th block succeeded @ Rx ' num2str(rx(i))])
    elseif rand > Pd % Decoding success
        b_counter(i)=b_counter(i)+1;
        node.buffer(tx(i),i)=nCoding(i);
        beep % Alert!
        rs(i)=blk_size; % Remaining Source packets
        lsi(i)=1; % Last source packets indicator
```

```
                fff(i)=1; % Point to delay
                sss(i)=1;
                gone(i)=1;
                disp(['Raptor: The ' num2str(b_counter(i)) ...
                    'th block succeeded @ Rx ' num2str(rx(i))])
            else % Decoding fail
                arq(i)=1; % Trigger ARQ for this path
                lpkt=blk_size-rs(i); % Number of lost original packets
                node.buffer(tx(i),i)=lpkt;
                pkt_start2{i}=zeros(1,lpkt);
                blk_fail(i)=blk_fail(i)+1;
                disp(['Raptor: The ' num2str(b_counter(i)+1) ...
                    'th block decoding failed @ Rx ' num2str(rx(i))])
                disp('$$$$$ ARQ triggered $$$$$')
        disp(['$$$ ' num2str(lpkt) ' lost original packets will be ARQed'])
            end
            node.buffer(rx(i),i)=0; % This block finished
        disp(['*** iter ' num2str(iter) ' @ PLR ' ...
        num2str(PLR(k)) ' & REDUN ' num2str(REDUN(l)) ' ***'])
            pkt_fail(i)=0;
    end
```

```
                if b_counter(i) == nBlocks % All done on this path
                    iter_path(i)=iter;
                    node.buffer(tx(i),i)=0; % All Done! Stop the Tx
                end
                % disp([num2str(b_counter(i)) ' blocks completed @ Rx ' ...
                % num2str(rx(i)) ' on the ' num2str(i) 'th path' ])
            end
        end
        pause(.01)
    end
    disp(['       ^^^^^^^Round     ' num2str(k) '     Outcomes ^^^^^^^^'])
    beep;pause(.3);beep;pause(.3);beep
    disp(['Iters: ' num2str(iter_path)])
    ITER_PATH(k,:)=iter_path;
    disp(['ARQed Blocks: ' num2str(blk_fail)])
    BLK_FAIL(k,:)=blk_fail;
    eff=nPackets./iter_path; % Network Efficiency
    EFF(k,:)=eff;
    disp(['Path Efficiency: ' num2str(eff)])
    apd=zeros(1,U); % Average path packets delay
    for i=1:U
```

```
        apd(i)=sum(pkt_delay{i})/nPackets;
    end
    APD(k,:)=apd;
    disp(['Path Packet Delay: ' num2str(apd)])
    vapd=zeros(1,U);
    for i=1:U
        x=0;
        y=length(pkt_delay{i});
        for j=1:y
            x=x+(pkt_delay{i}(j)-apd(i))^2;
        end
        vapd(i)=x/(y*(apd(i))^2);
    end
    VAPD(k,:)=vapd;
    disp(['Packet Dealy Variance: ' num2str(vapd)])
    disp(['      ^^^^^^^Round     ' num2str(k) '     Finish ^^^^^^^^'])
end
%% The End...
disp('     ******** The finals **********')
bITER_PATH(:,:,l)=ITER_PATH;
bBLK_FAIL(:,:,l)=BLK_FAIL;
```

```
        bEFF(:,:,l)=EFF;
        bAPD(:,:,l)=APD;
        bVAPD(:,:,l)=VAPD;
end
%% The End...
disp(['The elapsed time is ' num2str(toc/60) ' minutes.'])
rrp=rp*100;
drweff=zeros(U,lrd);
for i=1:U
        for j=1:lrd
                drweff(i,j)=100*bEFF(1,i,j);
        end
end
a=zeros(3,U);
drweff2=[];
for i=1:3
        a(i,:)=nh==i;
        c=sum(a(i,:));
        if c
                b=zeros(U,lrd);
                for j=1:U
```

```
            if a(i,j)
                b(j,:)=a(i,j)*drweff(j,:);
            end
        end
        drweff2=[drweff2;sum(b)/sum(a(i,:))];
    end
end
```