On Algorithms, System Design, and Implementation for Wireless Mesh Networks

YUAN, Yan

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

Abstract

The advances in wireless networking technologies create the possibility of deploying VoIP applications over wireless mesh networks (WMNs). However, the poor performance scalability issue prohibits the proliferation. The reasons include packet overhead in IEEE 802.11 networks, the contention on the medium access by the packets from the different flows, and the packet loss due to the transmission over multiple hops.

We propose a packet aggregation system to improve the performance of VoIP applications in wireless mesh networks. The basic idea of packet aggregation system is to aggregate small VoIP packets together to reduce the overhead at the MAC/PHY layer. In order to increase the chance of performing packet aggregation, we propose a distributed adaptive routing protocol in the wireless mesh network. It uses a scheme similar to AODV (Ad hoc On Demand Distance Vector Routing)to discover the potential routes to the destination. Moreover, when returning the available paths, the system tries to aggregate flows together to increase the chance for path aggregation. The adaptive routing protocol and the packet aggregation system are integrated into a complete system. The performance of the implemented protocol and system are evaluated and the results are analyzed. Experimental results show that the packet aggregation system combined with the adaptive routing protocol can improve the performance of VoIP applications significantly.

Finally, we present a network coding system, an implementation prototype of the network coding theory. The basic idea is to perform XOR operations on the packets from different sources, then broadcast the encoded packets to all the potential receivers. Network coding exploits the shared medium nature of the wireless network, and it reduces the actual number of transmissions by encoding the packets. We also implement the network coding in an prototype and carry out extensive experiments. Experimental results show that the implemented system brings a significant performance improvement to the wireless mesh network.

本論文討論如何提高VoIP 應用程式在無線網路中的性能以及相關的系統設計和實現問題.

近年來, VoIP 程式在無線網絡中的應用變得流行.但是, VoIP 程式在無線網路中的性能不佳, 成爲其發展的一個阻礙. 性能不佳的主要原因在於用於傳輸低層的數據包頭的時間過多。爲了解決這個問題，我們設計並且實現了數據包匯集系統，此系統收集並且匯集來自不同或者相同數據流的數據包，並且將多個原VoIP數據包整合成較大的數據包以此來提高傳輸效率。

爲了進一步提高可以進行數據包匯集的幾率, 我們設計並且實現了一個分佈式的路由協議。此路由協議通過一種類似AODV 的方式來發現可能的通路，並且同時嘗試返回一條可以幫助原發送節點進行數據包匯集的路徑。我們設計並且進行了各種實驗來評估數據包匯集系統和相關路由協議的性能。結果顯示進行路徑和數據包匯集之後，VoIP 程式在無線網路中的性能有很大提高。

最後，我們設計並且實現了一個網路編碼的系統原型。該系統利用無線網路的共享媒介特性，通過對合適的數據包流進行異或操作(編碼)，並且發送編碼後的數據包，從而減少需要發送數據包的總次數，所以可以提高無線網路的性能。實驗結果顯示該系統確實可以大幅提高無線網路的性能。

Acknowledgements

First of all, I would like to express my sincere gratitude towards my supervisor, Prof. John C.S.Lui. My M.Phil. studies could never have been completed without his exceptional guidance and consistent support. His breadth of knowledge and his enthusiasm for research amaze and inspire me. Special thanks go to my thesis committee members, Prof. Ng Kam Wing, and Prof. Michael R. Lyu, for their many comments and feedback on my thesis work and research directions.

Time spent at CUHK would not have been interesting and enjoyable without the friendship provided by my officemates and friends: Tao Chengjun, Cai Yi, Dai Hongling, Xu Xuemiao, Zhang Pingyue, Xu Leilei, Xiong Wei and Cheng Junzhou. I extend my gratitude to Xu Yuedong, Jiang Wenjie, Fan Bing, Wang Yue, for their help and discussion in many aspects of my research work. It has been a great pleasure working with all of you. I also express my appreciation to the faculty and staff in the Department of Computer Science and Engineering.

Finally, this work is dedicated to my family. My parents taught me the value of knowledge, the joy of love, and the importance of family. They have stood by me in everything I have done, providing constant support, encouragement, and love.

Contents

List of Figures

Chapter 1

Introduction

The advances in wireless networking technologies enabled wireless mesh network (WMNs) to become an emerging key technology in recent years. In WMNs, nodes are comprised of mesh routers and mesh clients. Each node is responsible for forwarding packets on behalf of other nodes that may not be within direct wireless transmission range of their destinations. WMNs are dynamically self-organized and self-configured. These features bring many advantages to WMNs such as low up-front cost, easy network maintenance, robustness, and reliable service coverage. WMNs are in fact a special kind of ad hoc network, with the main difference being that mesh routers have minimal mobility and form the backbone of WMNs. They provide network access for both mesh and conventional clients. Some mesh routers with the gateway function can provide the integration with other networks such as the Internet, cellular, IEEE 802.11, IEEE 802.15, IEEE 802.16, sensor networks, etc. WMNs can be employed in a range of application areas, including personal, local, campus, and metropolitan areas. Figure 1.1 shows an example of wireless mesh networks.

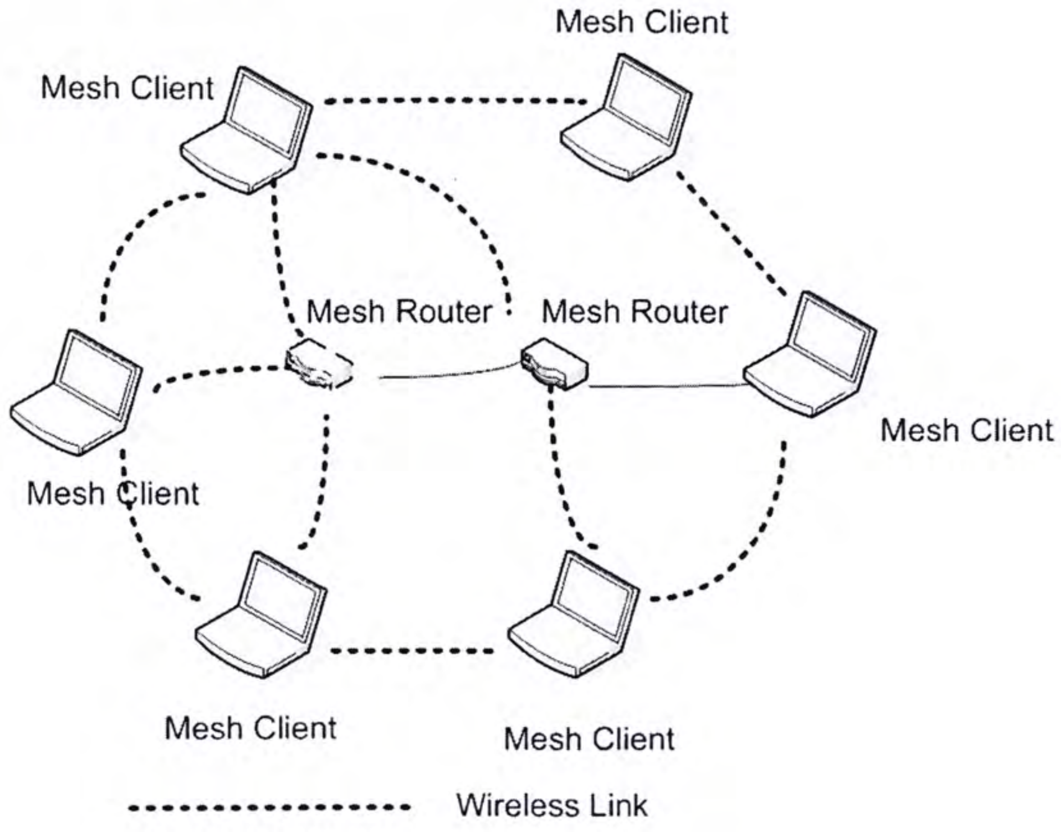

Figure 1.1: Wireless Mesh Network

Voice over Internet Protocol (VoIP) is another important technology and it is becoming one of the fastest growing applications because of its easy deployment and low maintenance cost. VoIP applications which employ the advanced voice-compression techniques can dramatically improve bandwidth efficiency. Moveover, VoIP applications make it possible to enable voice communication support with other media and data applications such as video, white boarding, and file sharing. Last but not least, VoIP calls requires less cost compared to traditional telephone network.

There have been trends of combining VoIP and wireless mesh network together and provide users with mobile phone services. The IEEE 802.11-based multi-hop wireless mesh networks can provide the wireless coverage to areas of enterprise-scale or community-scale, and if VoIP applications can be employed on the wireless network, one can enjoy the benefits like easy-deployment, failure tolerant, less maintenance cost, etc. However, the powerful combination of these two techniques still need to overcome many challenges such as keeping the end to end delay to a small value, because we need to ensure the quality of the phone calls. Thus it becomes an important topic to study the performance of VoIP applications on wireless mesh networks.

Network coding has been proposed in recently years and its goal is to improve the capacity of communication network. It is desirable to implement an experimental system to facilitate the future research.

1.1 Wireless Mesh Network

1.1.1 Architecture Overview

There are mainly three kinds of infrastructure for wireless mesh network [13]. They are :

- Infrastructure/Backbone WMNs. The infrastructure meshing is the most common type. In this architecture the mesh routers provide the gateway/bridge functions to the conventional mesh clients. Mesh clients do not involve in the routing under this scenario. Client meshing looks more like the conventional ad hoc network, in that mesh clients forms the network and are responsible for routing and configuring functionalities besides providing end-user applications to users.

- Client WMNs. Client meshing looks more like the conventional ad hoc network, in that mesh clients forms the network and are responsible for routing and configuring functionalities besides providing end-user applications to users. This type of WMNs has higher requirements on mobile clients.
- Hybrid WMNs. To mix the above two architecture together we get the third type of WMNs: hybrid WMNs. In this structure mesh clients can get access to the network via mesh routers as well as other mesh clients. This type of WMNs have the best mobility and inherits the advantages of the infrastructure meshing and client meshing.

Wireless mesh network mainly consists of two kinds of nodes: mesh routers and mesh clients. The differences between mesh routers and mesh clients include:

- Mobility Mesh routers are usually static compared to mesh clients. Mesh routers are usually built on embedded systems or desktop PCs, while mesh clients can be a variety of mobile devices.
- Number of Wireless interface Mesh routers can have multiple wireless interfaces installed, while each mesh client usually has only one single wireless interface.

- Function Mesh routers have the gateway and bridge functions; mesh clients do not. In some scenario mesh clients are also responsible of routing.
- Power supply Mesh routers are usually static so their power supply can be considered to be "infinite" while the power supply is an important issue for mesh clients.

Mesh routers in WMNs form the backbone of the network. Besides the routing capability for gateway/repeater functions as in a conventional wireless router, a wireless mesh router contains additional routing functions. Mesh routers are more "static" compared with mesh clients, so they are free of power supply problems. Mesh clients can also work as a router, but this will consume much more power. Usually gateway or bridge functions do not exist in mesh clients. In addition, a mesh client usually has only one wireless interface. They can be a laptop/desktop PC, pocket PC, PDA, IP phone, RFID reader, BACnet (building automation and control networks) controller, etc.

1.1.2 Routing Protocols

A lot of routing protocols are available in wireless mesh network. Two dynamic on-demand routing protocols in WMNs are DSR (Dynamic Source Routing Protocol) and AODV (Ad Hoc On-Demand Distance Vector protocol). The common characteristics of these two routing protocols include: they both try to find a route when necessary. Their differences include: DSR uses source routing, while AODV uses a table-driven routing framework and destination sequence numbers; AODV employs the time-out mechanism, but

DSR does not [5].

DSR Routing Protocol

The most prominent feature of DSR routing is the use of source routing, which means the sender knows the complete hop-by-hop route to the destination [12]. DSR is suitable when network diameter is small and there are moderate host mobility. DSR opens the promiscuous receiving mode to enable for form of optimizations, like gratuitous RREP. When the sender needs to send a packet, it will initialize a route discovery process by broadcasting route request(RREQ) packets. Those nodes who received RREQ packets will help forward them if they are not the destination themselves. Eventually the RREQ packets will arrive the destination or some node who knows the route to the destination, and a RREP(route reply) packet will be sent back to the source. In the route probing process, the intermediate nodes will cache the route found for future use. If any link is broken, the source node will receive a RERR(routing error) packet. Then a new route probing process will be initialized.

AODV Routing Protocol

AODV routing has a very similar route discovery process as DSR in that it also uses RREQ, RREP, RERR packets to establish the route and do error recovery. [6] However, in AODV routing, nodes only keep one entry in the routing table for a given destination, and AODV uses sequence numbers in packets to determine the freshness of the routing information and eliminating routing loops. Moreover, AODV maintains a timer in the node, when a route is inactive for a given period of time, it will be deleted from routing table. In contrast of DSR, RERR packets will be received by all the nodes along the routing path to notify them about the broken link.

1.2 Contribution of this Thesis

- Design and implement an adaptive routing protocol supporting path aggregation in wireless mesh networks.

- Build a packet aggregation system and combine it with the adaptive path aggregation routing protocol.

- Carry out extensive experiments to measure the performance improvement of packet and path aggregation system.

- Design and implement a network coding module to improve network capacity.

- Measure the performance gain of network coding on the built testbed.

1.3 Organization of this Thesis

The rest of the thesis is organized as following: In chapter two, we provide related works on VoIP, which include its performance, path / packet aggregation, and network coding issues in wireless mesh networks. Chapter 3 describes the design and system implementation of an adaptive path and packet aggregation system and also present the experiment results. The design and performance issues of network coding scheme are discussed in chapter 4. Finally, Chapter 5 summarizes the thesis and discusses future directions.

Chapter 2

Background and Literature Review

In this chapter, we conduct a detailed review on motivation, performance, and system design issues of VoIP wireless mesh networks.

2.1 VoIP on Wireless Mesh Networks

2.1.1 Performance of VoIP on Wireless Mesh Networks

Voice over IP(VoIP) has become one of the fastest growing application areas because of its tremendous benefits. However, the performance of VoIP over WMNs is a great hindrance on its way to proliferate [11]. The main reasons for the poor performance of VoIP applications in the wireless mesh network include:

- Packet overhead in IEEE 802.11 Networks. The overhead is caused by several reasons. First of all, in the wireless network transmissions between nodes suffer from reliability issues due to the interference and high loss ratio. Thus the 802.11 standard specifies a checksum for the physical layer encapsulation (preamble and PLCP header) to solve this problem in addition to the checksum for the MAC header and the packet payload. Moreover, to ensure the data has been transmitted successfully, the receiving node of a data frame needs to reply with an acknowledgment. Another important difference is, a wireless node cannot send and receive packets using the medium simultaneously. Thus the sender of a message cannot determine a collision on the transmission until the transmission of the frame is finished. What is more, the "hidden node" problem further exacerbate this effect. Another factor that causes the packet overhead in 802.11 wireless network is the low transmission rate of the common header at the MAC/PHY layer. The 802.11b standard offers several different transmission rates, and these rates use different modulation techniques. Each access point or node can use any of the modulation techniques and automatically changes to different modulations depending on bit error rate and signal strength. Because different stations served by the same access point may use different modulation techniques, each message uses a common preamble that encodes the configuration options. The common modulation technique uses the lowest transmission speed (1mb/s) for the common header. Thus messages are transmitted in two different rates, with the common header transmitted in a relative much lower rate, and brings in the overhead.

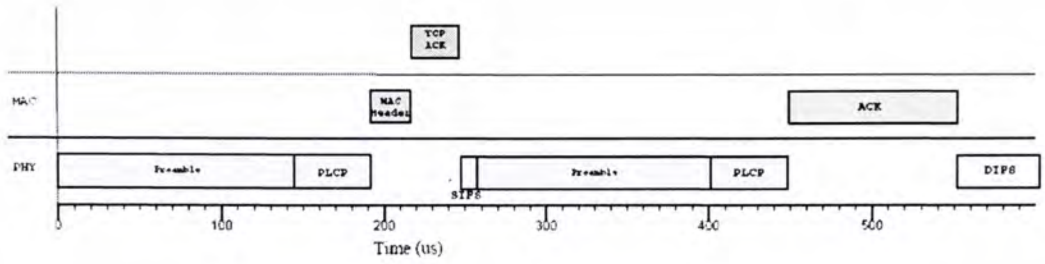

Figure 2.1: Overhead in wireless network

Figure 2.1 from [3] illustrates the transmission overhead of an packet over the 802.11b wireless network. In this graph, we are assuming the standard long preamble and the transmission rate is 11Mbps. We can see from the graph that the required MAC and physical layer transmissions occupy the medium for an order of magnitude longer than the actual data packet. A large part of this overhead is caused by the requirement to transmit the preamble, PLCP header and control frames in the MAC layer at the Basic Service Set Basic Rate, typically 1 Mbps. Thus if we can reduce the overhead of transmitting the headers, the network throughput should be much higher.

- The high protocol(IP/UDP) overhead compared to the relative small packet payload. The VoIP applications usually have a system architecture like the following: There are encoders and decoders at each of the client nodes, and they exchange packets via an IP network. The packet size and frame rate used by VoIP codecs usually have following characteristics: the voice data are encoded and compressed into constant bit rate packets. Generally speaking, the packet payload size ranges from 10 bytes to 100 bytes, and the packet interval ranges from 10ms to 30ms. Therefore the overhead of the VoIP packets are large if we take into account the 28 bytes-long IP/UDP protocol header, which further degrades the utilization of VoIP packets.

- The contention on the medium access by the packets from the different/same flows. Another factor that influences the VoIP utilization is that VoIP applications usually use UDP to transmit the data packets; however, it is shown that the UDP performance degrades very quickly with the increased number of hops. This phenomenon is mainly caused by the self interference of the UDP packets which belongs to the same flow as they are competing for the medium access. Experiments have shown that even in a linear topology, where there are n nodes which are all interferencing with others, the UDP throughput can degrade to $\frac{1}{n}$.

- the packet loss due to the transmission over multiple hops;

To cope with the problem of low performance of VoIP in WMNs, we need to apply optimizations for it to improve the performance.

2.1.2 Optimizations for VoIP over Wireless Mesh Networks

To improve the performance of VoIP traffic over wireless mesh networks, several optimization techniques have been proposed. Authors in [25] propose to

improve VoIP quality through path switching. Authors in [22] propose to use DCF to support VoIP. However, no solution is provided to improve the VoIP performance over WLAN. Authors in [22], [28], and [1]investigate various schemes for improving the VoIP capacity, but all the proposed schemes require modifications of the MAC protocol used by the VoIP applications. Authors in [1] need to modify the non-VoIP data stations of the MAC protocol. In [9], authors propose an M-M scheme. The main idea of the packet MCM scheme is to combine the data from several downlink streams into a single larger packet. In this way, the overhead of multiple VoIP packets can be reduced to the overhead of one packet. Then the multiplexed packet is multicasted so all nodes can receive it by a single transmission.
Authors in [8] experimentally investigate several methods to improve the quality of VoIP over WMNs. They are using multiple interfaces, label based forwarding architecture, and packet aggregation. They implement a distributed packet aggregation strategy that exploits the MAC waiting time and perform packet aggregation, without introducing unbounded packet delays. These performance optimizations are implemented in a 15 node wireless mesh network, and the experimental results show a performance increase of 13 times for a six-hop network when all optimizations are used. Authors in [3] analyze the overhead of 802.11b networks at high modulation rates. They also propose a list of advantages on packet aggregation. Their simulation results show significant improvements in actual network capacity, ranging from a 60-270% improvement for traditional adhoc scenarios and to a 100-500% improvement for wireless networks using unidirectional antennas.

2.1.3 Path and Packet Aggregation Scheme

Packet aggregation is adopted by VoIP applications to increase the network utility. To enhance the chance of packet aggregation, path aggregation is proposed to further improve the network performance.
In general, the basic operation of packet aggregation is to reduce the overhead by combining multiple small VoIP packets from either the same flow or different flows which have the same next hop together as to form a large packet. There are several concerns about the pack aggregation: the first one is that larger packets are more prone to error, and the increased packet size will result in a lower overall performance. Actually, experiments have shown that the advantages of packet aggregation still provide higher throughput than the scenario without packet aggregation. Another concern is the extra delay introduced at the aggregation nodes may accumulates at the intermediate nodes and affect the end-to-end delay. However, experiments also show that even with the small delay imposed at the aggregation nodes, we can still obtain significant reduction in packet latency compared to non-aggregated case under congested scenarios.

Packet aggregation only aggregates packets on the same route, and actually we can also try to aggregate the packets from different flows by properly changing their route. However, current routing protocols in the wireless ad hoc network such as AODV and DSR provide no such support to increase the chance of packet aggregation. Thus, we need to propose a new routing protocol which can simultaneously return the path and to increase the chance of performing packet aggregation on the intermediate nodes. We name the process as path aggregation.

2.2 Network Coding on Wireless Mesh Networks

2.2.1 The Concept of Network Coding

Network coding is proposed to improve the performance of wireless networks. In network coding scheme, in addition to forwarding packets, routers encode packets from different sources to increase the information content of each transmission.

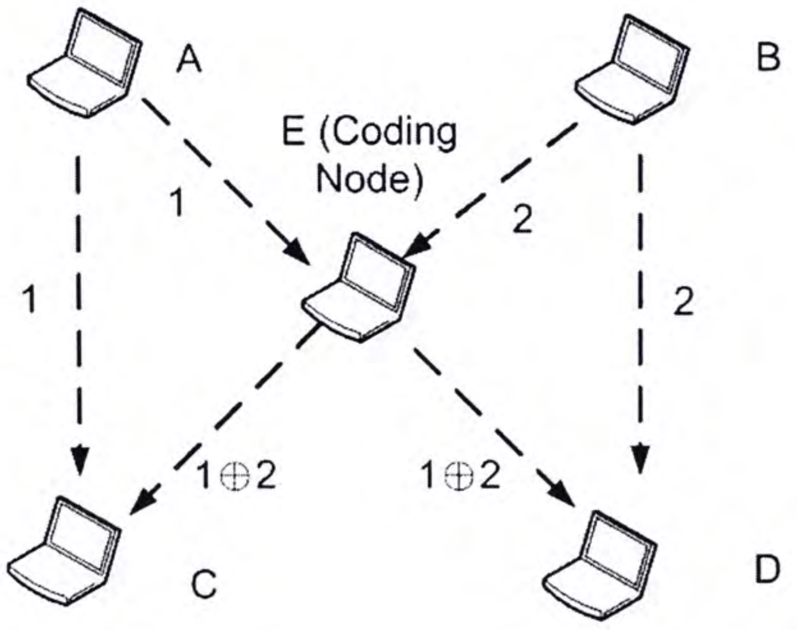

Figure 2.2: Example of network coding

The basic idea network coding can be illustrated using the following example. Consider the network in Fig 2.2 , where source node A wants to send a message 1 to node D, and source node B wants to send a messages 2 to node C. Here we assume that A cannot directly communicate with D, and B cannot directly communicate with C. But the node C can overhear the transmission signal of node A, and similarly, node D can overhear the transmission signal of node B. Assume all links have a capacity of one message per unit of time. If the intermediate node E directly forward the messages it receives, the middle link will be a bottleneck, which for every time unit, can either deliver 1 to D or 2 to C. However, if we employ network coding in this scenario, the router can perform an XOR operation to encode the two messages 1 and 2, as shown in the figure, both receivers (C and D)can obtain two messages in every time unit. The receivers can decode the XORed packet by XOR it again using the overheared packets. The shared medium nature of wireless network is exploited and it is shown that mixing packets can reduce the number of transmissions. Thus, network coding, i.e., allowing the routers to XOR the bits in forwarded messages, can increase network throughput.

2.2.2 Related Work

There is a rich literature on the issues on network coding. The first paper on network coding is written by Ahlswede et al. [20], who shows that having the routers mix information in different messages allows the communication to achieve optimal natural capacity. Following this work, Li et al. shows that, for multicast traffic (e.g., the butterfly scenario), linear codes are sufficient to achieve the maximum capacity bounds in [24]. Koetter and Medard [16] present polynomial time algorithms for encoding and decoding, and Ho et al. extend the results via random codes [29]. Recently, [2, 23]studies network coding in wireless environments. In [7] Lun et al. studies network coding with omni-directional antennae and shows that the problem of minimizing the communication cost can be formulated as a linear program and solved in a distributed manner. Above work are mainly theoretical. A few papers study specific unicast topologies showing that, for the studied scenario, network coding results in better throughput than pure forwarding [10, 17, 30]. Much system research has also been done on the problem of improving the throughput of wireless networks. The proposed solutions include designing better routing metrics [6, 14, 21] to changing the TCP protocol [19], and improved routing and MAC protocols [4, 15, 18]. In [26] the authors propose COPE, a new forwarding architecture for wireless mesh networks. It inserts a coding layer between the IP and MAC layers, which detects coding opportunities and exploits them to forward multiple packets in a single transmission. COPE incorporates three main techniques: (a) Opportunistic Listening:(b) Opportunistic Coding:(c) Learning Neighbor State. They define the coding gain as the ratio of the number of transmissions required by the current non-coding approach, to the minimum number of transmissions used by COPE to deliver the same set of packets. By definition, this number is greater than or equal to one. They analyze certain basic topologies that reveal some of the factors affecting COPEs coding gain. Their analysis assumes identical nodes, omnidirectional radios, perfect hearing within some radius, and the signal is not heard at all outside this radius, and if a pair of nodes can hear each other the routing will select the direct link. They claim that in the absence of opportunistic listening, COPEs maximum coding gain is two, which is achievable. In this work, authors also propose several design principles. The first one is never delaying packets. Second, to give preference to XOR-ing packets of similar lengths. Third, searching for appropriate packets to code due to the maintenance of virtual queues. Last, try to ensure each neighbor to whom a packet is headed has a high probability of decoding its native packet by calculating an estimate of the coding probability.

Pseudo-broadcast and Hop-by-hop ACKs and retransmissions are also deployed in COPE to improve the performance. Extensive experiments are carried out in gadget topologies, Ad Hoc Network, and Mesh Access Network. Both TCP an UDP flows are tested. The results show that COPE increases network throughput. The gains vary from a few percent to several folds depending on the traffic pattern, congestion level, and transport protocol.

Chapter 3

Adaptive Path and Packet Aggregation System

3.1 Overview

The emerging mobile wireless environment poses exciting challenges to VoIP applications. We propose a distributed adaptive routing protocol in the wireless mesh network. It uses a scheme similar to AODV to discover potential routes to the destination, and moreover, when returning the available paths, tries to aggregate flows together to increase the chance for path aggregation. Besides the adaptive routing protocol, in this section we also describe the system design and implementation details of the packet aggregation system. We also explain how the adaptive routing protocol and the packet aggregation system are integrated into a complete system. The performance of the implemented protocol and system is evaluated and the results are presented.

3.2 The Adaptive Path Aggregation Routing Algorithm

3.2.1 Protocol Overview

The adaptive path aggregation(APA) routing protocol returns the source node an available route to the destination while trying to aggregate network flows together to create more opportunities for them to share the same path. Like other ad-hoc routing protocols such as AODV, APA uses route request(RREQ), and route reply(RREP) messages to probe and return possible routes. Moreover, APA also uses weight-setting messages to adjust the "weight"s of the network links to help the routing of the whole active network converge to a subgraph. A node disseminates a RREQ when it determines that it needs a route to a destination and does not have one available (including the case that an existed route has already expired). When an intermediate node receives a RREQ, it will calculate and update the weight field in the RREQ. If an intermediate node finds itself already in the path probed, it will discard the RREQ. Otherwise, it will append its own IP address to the RREQ. To prevent unnecessary network-wide dissemination of RREQs, the forwarding node should keep a temporary table for every source IP address for a short time period (about the same as the normal route discovery time period), and apply certain filtering rules to the RREQs to reduce the number of unnecessary re-broadcastings of RREQs. A node only generates a RREP if it is the destination node. The destination is responsible for making the decision on which path to return, and it will unicast the RREP containing the selected route to the source node. One or more weight changing messages are also created according to the different situations and sent to adjust the related link weights.

3.2.2 Data Structure

In this section we present the data structures used by APA protocol. In the APA routing algorithm, there are mainly three kinds of control messages : Route Requests (RREQs), Route Replies (RREPs), and Weight Changing Messages(WCM). RREQs are used to probe the possible routes to a certain destination, and RREPs are used to return the selected path to the RREQ originating node by the destination nodes. WCMs are unicasted by the destination after it sends out the RREP and change the link weights along a certain path.

Routing Request Message(RREQ)

The format of the Route Request message is illustrated above in Figure 3.1. It contains the following fields:

- type: the enumerated type of the message, set to be APA_RREQ;
- id: the unique id of the message at the originator, range from 0-127;
- TTL: indicates how many hops the message can be forwarded before it is discarded;

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |      id       |     TTL       |   Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             flag              |            weight             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP Address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Originator IP Address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Intermediate Node 1 IP Address             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Intermediate Node 2 IP Address             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Intermediate Node ...IP Address            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Intermediate Node N  IP Address            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3.1: RREQ message structure

- hcnt: the number of hops this message has traversed so far;
- flag: the value can be either 0 or 1, which indicates the RREQ message is on the first stage and second stage respectively;(The meaning of stages will be explained later)
- weight: the weight of the path the packet has traversed so far;
- orig_addr: the IP address of the originator;
- dest_addr: the IP address of the destination;
- intermediate node N: the IP addresses of the intermediate nodes on the probed path.

Routing Reply Message(RREP)

The structure of routing reply messages is similar to the routing request messages. Figure 3.2 illustrates the format of RREP. A RREP message consists of:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |      id       | Current Count|   Hop Count    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            weight             |          reserved             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Destination IP address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Originator IP address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Intermediate Node IP address                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             ......                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Intermediate Node IP address                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3.2: RREP message structure

- type: the enumerated type of the message, set to be APA_RREP;
- id: the unique id of the message at the originator, range from 0-128;
- ccnt: indicates which hop on the path the message is currently at, used to determine the next hop to forward this message;
- hcnt: the number of hops of the path contained in this message;
- orig_addr: the ip address of the originator;

- dest_addr: the ip address of the destination;
- intermediate node N: the ip addresses of the intermediate nodes on the returned path;

Weight Setting Message(WSM)

Another type of message in APA is weight setting messages. Figure 3.3 illustrates the WSM format. Following discusses the meaning of each field:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   Hop  Count  | Current Count |    Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Originator  IP Address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP Address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Intermediate Node 1 IP Address             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Intermediate Node 2 IP Address             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Intermediate Node ...IP Address            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Intermediate Node N  IP Address            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Weight 1  |       Weight 2  |  Weight ...  |     Weight n  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3.3: WSM message structure

- type: the enumerated type of the message, set to be APA_WSM;
- ccnt: indicates which hop on the path the message is currently at, used to determine the next hop to forward this message;

- hcnt: the number of hops of the path contained in this message;
- reserved: padding space; (Not used now, to keep the bits aligned and for possible future use)
- orig_addr: the ip address of the originator;
- dest_addr: the ip address of the destination;
- intermediate node N: the ip addresses of the intermediate nodes on the path which needs its weight adjusted
- weight N: the weight to be set on the N th link
- weight: the new weight to be set on each link along the path

Route Table Entry

APA uses the following fields to construct a route table entry:

- dest_ip: destination ip Address;
- id: id of the entry, used to determine the "freshness" of the entry;
- firstHop: the pointer which points to a list of intermediate nodes
- hcnt: the number of hops on the path;
- time_expire: the time left before the expiration time of the route;
- state: state of the entry; (possible values are: active/inactive/seeking. The meaning of each possible state will be discussed in section 3.2.4)

3.2.3 The Concept of Link Weight and Path Weight

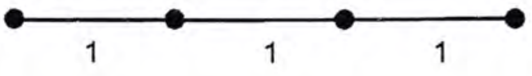

Figure 3.4: Example: path weight = 3

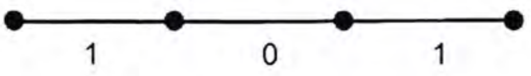

Figure 3.5: Example: path weight = -2

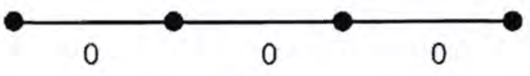

Figure 3.6: Example: path weight = 0

A parameter called "weight" is associated with each directional link at the ending node, which is initialized to be zero when the APA system is launched. A link can have a weight of either 1 or 0. A path also has its weight, depending on the link weights along this path. The possible path weights could be :

- a positive integer equal to its length, when all the links of it has the weight of one. Figure 3.4 is an example of a positive path. In this example the weight is three, whose physical meaning is the path consists of three hops and all the links are of weight one;
- a negative integer whose absolute value is equal to the number of its links that have the weight of one. Figure 3.5 is an example of a negative

path. In this example the weight is minus two, which means the path consists of three hops and two the links are of weight one;

- zero, when all the links are of weight 0. Figure 3.6 is an example of a zero path. In this example the weight is zero, which means all the links in the path are of weight zero;

Path weights are used to determine which route to select by the destination, and help the network topology converge. We will discuss how the link weights are changed in later section.

3.2.4 APA Operations

APA Algorithm

The basic algorithm of adaptive path aggregation routing protocol is illustrated below.

Proof on the Convergence of APA Algorithm

We define the term "Convergence" here to be a state that the route between any two nodes in the network keeps stable(Not changed frequently).
Assumption: All links in the network are bi-directional.

In a network, randomly pick up two nodes A and B; according to APA Algorithm, all links' weight are zero initially. Once A selects one path to B, (mark it as path 1) it is impossible for A to select another different path to B. Because the positive paths are selected with highest priority. If another path (mark it as path 2) is selected by some other node to pass through A to B, then path 2 must be shorter than path 1. But this is impossible, because

Algorithm 1 APA Algorithm: The source node

```
$Input$ : source IP $s$, destination IP $d$, outgoing Packet $p$, delay constraint $dc$;
$isAvailableRoute = checkRouteTable(d)$
**if** $isAvailableRoute = TRUE$ **then**
    put packet $p$ into aggregation queue $Q_d$
**else**
    $rreq = newRREQ(d, dc)$
    broadcast out $rreq$
    **if** RREP received in given time period **then**
        put packet $p$ into aggregation queue $Q_d$
    **else**
        drop packet $p$
    **end if**
**end if**
```

Algorithm 2 APA Algorithm: The intermediate node

```
$Input$ : RREQ packet $rreq$;
$isLoopFree = checkLoop(rreq)$
**if** $isLoopFree = FALSE$ **then**
    drop rreq;
**end if**
**if** $remainingTTL(rreq) = 0$ **then**
    drop rreq;
**end if**
forward the rreq to neighbors
```

Algorithm 3 APA Algorithm: The destination node

```
*Input* : received RREQ set $rreqSet$, $posWeightPathNum$, $negWeightPathNum$, $zeroWeightPathNum$)
if $posWeightPathNum > 1$ then
  randomly pick up one path $p$ from the $rreqSet$;
  $q \leftarrow$positive paths in $rreqSet$ except $p$
  $s = sharedLinks(p, q)$
  generate weight setting message $wsm$ to set all the link weights on all the other positive paths in rreqSet to be zero, except the links in $s$
else if $posWeightPathNum = 1$ then
  pick up the only path $p$ of positive weight;
else if $negWeightPathNum > 0$ then
  pick up the path $p$ with the shortest length
  if there are multiple paths with the same length then
    break the tie by choosing the path $p$ with least weight value;
  end if
  if there are multiple paths with the same weight value then
    break the tie by randomly pick up $p$;
  end if
  generate weight setting message $wsm$ to set all the links weights on the chosen path $p$ to be one
else
  pick up the shortest path $p$ from rreqSet in which all paths are of weight zero
  if there are multiple paths with the same length then
    break the tie by randomly choosing one path $p$
  end if
  generate weight setting message $wsm$ to set all the links weights on the chosen path $p$ to be one
end if
sent out the weight setting message
generate rrep message using the chosen path $p$ and return it to the source
```

initially A will select the shortest path to B. (According to the rule when all links' weight are zero) Thus A will have at most one path to any node B. The network is stable.

Routing Table

Because the APA routing protocol is an on-demand routing protocol, the source node will only initiate the route discovery process when it needs to send packets to an unknown destination. The source should first consult the route table to check if an available route entry exists. If so, the path in the route table will be taken. If not, there will be three cases to consider:

- If there is no record for the destination ip address, the route table should create a temporary entry for the destination ip, and mark its status to be RT_SEEKING. The packet will be put into a temporary queue and sent out when the route is available.
- If an route table entry of the destination ip address is found, and the status is RT_SEEKING, the incoming packet will also be buffered in the temporary queue and wait to be sent out.
- If an route table entry of the destination ip address exists and its status is RT_INACTIVE, the route entry will be set to RT_SEEKING, and the packet should be buffered in the temporary queue.

The routing table keeps an unique complete path to every destination. The path is obtained from the RREP message. Every entry in the routing table has a timer which indicates how much time left before the entry expires. The entry for a certain destination will be set to inactive if no flow is destined to the destination for a certain time period (expire time in the route table). In this case, the sending node will need to initiate a new route discovery process if later the source nodes needs to send packets to the destination again. The id field in the route entry indicates the "freshness" of the path contained in the entry, and is exactly the same with the id value in the RREP message used to construct this route entry. If a nodes receives a RREP with a smaller id compared to the id of the current route entry, the RREP will be discarded. Note that because the data width used to represent the RREP id is limited (8 bits), there may be cases in which the RREP id overflows. In order to cope with this problem, the node will not discard the RREP if the difference between the current id and received id is greater than 10.

Generating Route Requests

A node disseminates a RREQ when it determines that it needs a route to a destination and does not have one available. This could happen if the destination is previously unknown to the node, or if a previously valid route to the destination expires. Every source node should keep an unique global integer to represent the greatest currently used id number. The source node will first increase the local global integer by one, then use it in the new RREQ message as the id. Then all necessary fields including TTL, destination ip address, ..., etc, are filled according to the tuneable parameters in a configuration file. The route discovery process consists of two phases. The first phase is to find a possible route in the subgraph of the network topology which includes only the weight one links. This is done by setting the "flag" attribute in the

RREQ to be zero. The intermediate nodes will not forward the RREQ if the traversed path contains any zero-weighted links. If an available path is found during the first phase, the algorithm will not proceed to the second phase. Otherwise, the source node initiates the second phase of the algorithm, in which every node in the network needs to forward the RREQ message. The purpose of the two-phase route probing is to reduce the number of broadcasted packets, and this works especially well when the source node is just performing an regular "update" on the current path.

Controlling Dissemination of Route Request Messages

To prevent unnecessary network-wide dissemination of RREQs, the intermediate nodes should filter out those unnecessary RREQs when forwarding them. Also, proper TTL value in RREQs should be set according to the network diameter. In order to achieve the first, we keep a temporary table for every source ip address for a relative short time period (about the same as the normal route discovery time period), and apply certain filtering rules to the RREQs matched to the pair. Those packets which match one or more of the rules are simply dropped at the intermediate nodes. The temporary table contains following columns:

- The largest RREQ id forwarded for the source ip address;
- The maximum number of weight-one links on those negative paths forwarded for that source ip;
- The length of the above negative path;

- The minimum length(number of hops) of the zero path forwarded for that source ip;

Following filtering rules are applied at intermediate nodes when forwarding the RREQs:

- If the RREQ contains an id which is smaller than the id kept in the table, discard it; (Because a smaller id means the RREQ is out-of-dated)
- If the RREQ contains a path of positive weight, forward it; because when the destination picks up the path to be returned, all positive paths are eligible to be chosen. We can not discard any RREQ containing positive paths.
- If the RREQ contains a path of zero weight, first check if we have forwarded any negative path; if so, only forward the RREQ if the length of the forwarded negative path is longer than the zero weighted path from the RREQ; if local node has never forwarded any negative path, forward the RREQ only if the length of the path in the RREQ is shorter or equal to the minimum length of forwarded zero weighted paths recorded in the table, and update the minimum length of zero weighted path in the table.
- If the RREQ contains a path of negative weight, forward it if it is shorter than the negative path recorded in the table; if they have the same length, only forward it if the path in the RREQ has more weight one links than the negative path in the table, and update the maximum

number of weight one links along negative paths in the table. The reason for doing this is to filter out the paths which will not be chosen at the destination node.

Processing and Forwarding Route Requests

When a node receives a RREQ, it first increments the hop count value in the RREQ by one, to account for the new hop through the intermediate node. Then it will decrement the TTL value in the RREQ head by one. A RREQ is discarded if its TTL value becomes zero. Then the intermediate node will calculate and update the weight field in the RREQ with the weight value of the link, whose two ends are the node itself and the incoming RREQ node according to following rules:
When the weight of the link between the local node and the RREQ receiving node is equal to one:

- If the path up to the local node consists of links that are all of weight one, increase the path weight by one;
- If the path up to the local node consists of links that are all of weight zero, set the path weight to be minus one;
- If the path up to the local node contains both weight one links and weight zero links, decrease the path weight by one.

When the weight of the link between the local node and the RREQ receiving node is equal to zero:

- If the path up to the local node consists of links that are all of weight one, multiply the path weight with minus one;

- If the path up to the local node consists of links that are all of weight zero, keep the current path weight unchanged;
- If the path up to the local node contains both weight one links and weight zero links, keep the current path weight unchanged.

The above rules can be easily derived from the definition of the path and link weights. If the intermediate node finds itself already in the probed path, it will discard the RREQ to ensure the path is loop-free. Otherwise, it will append its own IP address to the end of the RREQ. If the node is not the destination node, and if the remaining TTL is larger than 0, the node updates and broadcasts the RREQ to all its neighbors.

Figure 3.7 shows an example of RREQ forwarding process. Here node A is

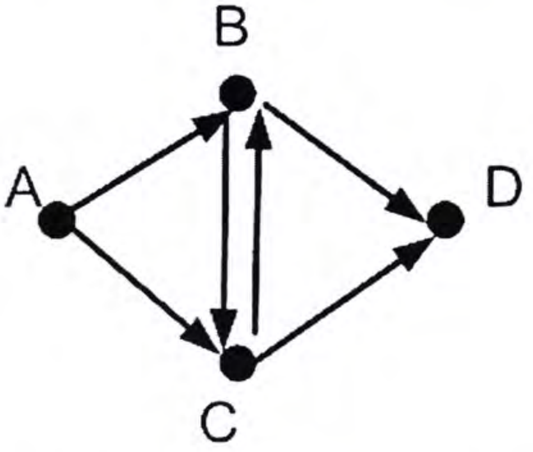

Figure 3.7: Example: RREQ Forwarding

trying to find a route to node D. First, A generates and broadcasts RREQs to its immediate neighbor B and C, then B and C forward the RREQ to their immediate neighbors, respectively. The RREQs which come across a loop are discarded. For example, the RREQ message from node B (C) to node C (B) will be discarded by node C (B). Eventually the RREQ arrives at the destination, D.

Generating Route Replies

Only the destination node can generate the route replies. An unique RREP containing the selected route to the source node will be unicasted back to the source node according to the following principles. Upon the first arrival of a (source, RREQ id) pair, the destination will create a temporary list to contain all the probed paths for that (source ip, RREQ id) pair. The list will have a timeout value initialized to be around the normal RREQ propagation time period when created. When the list times out, the destination will pick up one path from the list according to following rules, and unicast the selected path in a RREP to the source IP.
Rules for selecting paths:

- If there are multiple positive paths, randomly select one and set all the link weights along other positive paths except those links which are shared by the selected path to be zero;
- If there is only one positive path, select it and return it to the source;
- If there is no positive path, and there is at least one negative path, select the shortest negative path; break the tie by selecting the negative path with most weight one links; if there are still multiple paths fulfilling the condition, randomly pick up one;

- If there are only zero weighted paths, select the shortest one; break the tie by randomly selecting the qualified path. Then set all the link weights along the selected path to be one.

To generate the RREP, the destination node first fills in the RREP with the weight and the hop count of the selected path, which can be obtained directly from the corresponding RREQ. Similar to the RREQ, the node appends the complete list of intermediate nodes' IP addresses to the end of the RREP. However, the order of the intermediate nodes are reversed compared to RREQ.

Figure 3.8 shows an example of RREP forwarding. Assume the desti-

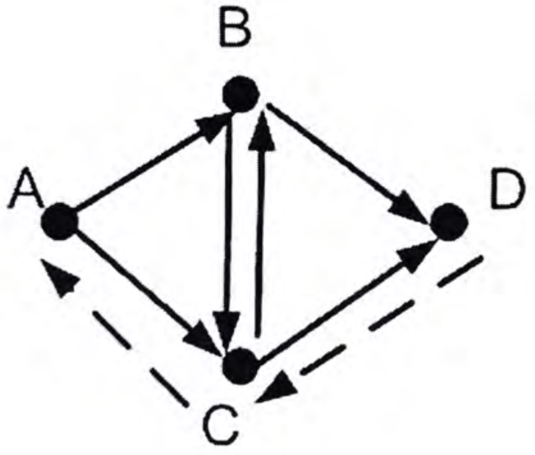

Figure 3.8: Example: RREP Forwarding

nation D picked up the route A-C-D, it will generate the RREP containing the above path and unicast it back along the reverse way of the RREQ as indicated by the dash arrow(D-C-A).

Receiving and Forwarding Route Replies

When a node receives a RREP message, it will first identify the next hop with the ccnt field in the RREP, and forward it to the right neighbor if itself it not the source node. When the source nodes receives the RREP, it will update the route entry in the route table for the destination IP. All the necessary information, including the hop count, the complete path, and the id of the path, ..., etc, are copied into the route entry. Then the node will scan the queues in the buffer to send out the packets waiting for this RREP.

Changing the link weights along the selected path

After generating route reply message, the destination will also produce one or more weight changing messages according to the different cases specified in the "Generating route replies" section. Then the weight changing messages are sent out to adjust the link weights in the network graph. The nodes on the path contained in the weight changing message will first identify its own position along the link, then set the new value of the link weight, which is also specified in the weight changing message. Finally the node will forward the weight changing message to the next hop according to the path contained in the weight changing message.

Figure 3.9 and 3.10 shows the change of the link weights after the WSM (Weight Setting Message)is generated and sent as the dash arrow. In this case, destination D decides to return the path D-C-A, and it will set the link weight between A-C and C-D to be one.

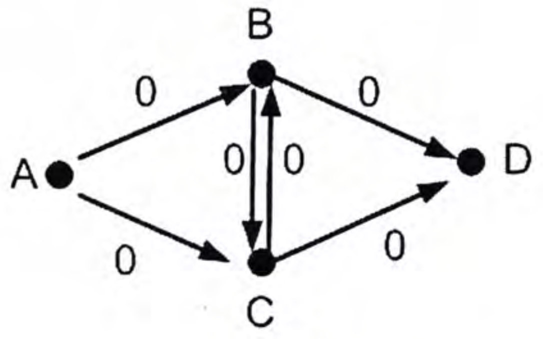

Figure 3.9: Example: Link weights before WSM is sent

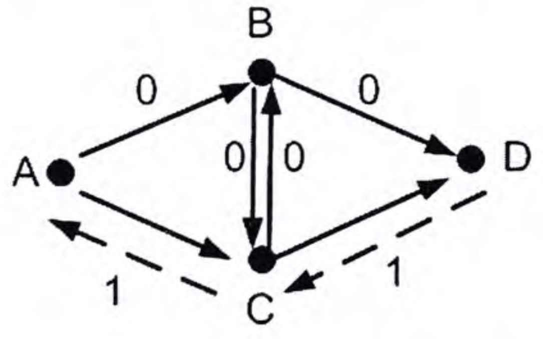

Figure 3.10: Example: Link weights after WSM is sent

3.3 The Packet Aggregation System

3.3.1 Overview

In this section, we present the design of our packet aggregation scheme and explain how it can be integrated to current wireless mesh network architecture and work with existing VoIP applications. The idea of packet aggregation is motivated by the observation that, in the wireless mesh network, VoIP packets are transmitted with a huge number of small packets, which brings high percentage of overhead, and also incurs network contention. The basic idea of packet aggregation is to combine several small packets together and send them using a single header. It's a reasonable to argue that the extra packet delay may affect the overall performance; but in the experiment result section we will show that if we can design the routing algorithm correctly, the extra waiting time spent on intermediate nodes will not affect the overall performance significantly.

In our implementation, the packet aggregator/deaggregator will be running as a daemon process at each wireless node. We name the daemon as PAS, which means "Packet Aggregation System". PAS will be responsible for collecting packets from locally generated traffic as well as aggregating packets received from other nodes. Here, we classify the network traffic in two classes: aggregated traffic and non-aggregated traffic. All the aggregated traffic will be transmitted and received at a particular uniform udp port; while the rest of the "normal "traffic are not affected. The daemon process will work at the application layer just like a normal user application, and does not have to modify of the kernel code. This helps our implementation to be deployed easily on any wireless platform regardless of the underlying routing protocol, network architecture, etc.

3.3.2 Packet structure

The packet types used by the packet aggregation system includes: APA_LC_PACKET and APA_PACKET. APA_LC_PACKET means a packet that has already been locally compressed (We will explain this in next section), and APA_PACKET represents a normal aggregated packet. Every PAS pacekt will be constructed from several small VoIP packets, which has a small PAS header at the beginning. PAS packets are transmitted as normal UDP packets at port 16888, and the receiving and sending process will happen at the same port. The structure of the PAS header is illustrated in Figure 3.11:

The first byte next to the UDP header records the total number of

IP	UDP	PacketNum	PktLen1	PktLen2	...	Packet1	Packet 2	...

Figure 3.11: The structure of APA PACKET

individual packets contained in this aggregated packet. The next N bytes will be the corresponding lengths of the individual packets in binary form, while N is equal to the total number of small packets. The remaining part is consisted of several small original VoIP packets, each of them with their complete IP/UDP headers. At the end of the packet, all the ip addresses of the intermediate nodes are appended here.

3.3.3 Local Compression

One problem with the source routing algorithm is that it takes too much space in the packet header to carry the IP addresses of the complete path, and this overhead is obvious affecting the overall performance. In order to cope with is problem, we need to first combine several original VoIP packets together before putting them into the aggregation queue to reduce the relative percentage of space used to accommodate the complete path. This is done by deploying a process which listens to the port of the VoIP applications, and intercept all the outgoing packets. The process will put the original VoIP packets into a compression queue and combine several (which is a tunable parameter) original packets into a large packet, then forward it to the packet aggregation daemon.

3.3.4 Packet Aggregation/Disaggregation

Algorithm 4 PAS Algorithm: Aggregation Procedure

```
$Input$ : VoIP packet $p$;
$n = getNextHop(p)$
put p into aggregation queue $Q_n$
**if** $size(Q_n) \geq MTU$ or $pktNum(Q_n) \geq MPN$ **then**
    encapsulate packets in $Q_n$ into aggregated packet $p_a$
    forward $p_a$to  next hop $n$
**end if**
return
```

Next we describe how the packets (after being locally aggregated) are aggregated / disaggregated in details. The algorithm for aggregating packets are shown in Algorithm 4. After PAS receives an locally compressed packet from local compressor, it will directly put the packet into the aggregation queue. If PAS receives aggregated packets from some neighboring nodes, it will need to first disaggregate it according to the information contained in the header. The total number and lengths of the VoIP packets are decoded from the PAS header, and PAS will extract out the original VoIP packets one by one. Then PAS will look into the destination ip address contained in the ip header of VoIP packets, and starts to find a table matching to find the corresponding next hop ip address to the destination. In order to create and update the routing table, we need the underlying routing algorithm. In the current implementation we adopt APA as the routing algorithm, which is a kind of source routing protocol. PAS will find the next hop for a given packet from its APA header. Then the packet is put into an aggregation queue according to the ip address of next hop. Several queues are maintained in the form of linked lists to serve as the aggregation buffer in PAS; each queue corresponds to an unique next hop ip address. The PAS will periodically check (i.e. for every 10ms) the status of the queues in the buffer, and if any one of the following conditions is fulfilled, the queue will be encoded into one large packet based on the PAS protocol and and sent to next hop immediately:

- The queue has been waiting for a certain period and times out;
- The length of the queue has exceeded certain specified constraint;
- The number of small packets in the queue has exceeded a pre-defined limit.

We introduce a budget of waiting time for each queue when the queue is created, and decrease the time on every checking round. When the waiting delay decrease to zero, the queue time outs and if there is any waiting packet, it will be sent out. Although the waiting time may accumulate at every intermediate node, it won't affect the overall performance because the amount of waiting time is very small compared to the total transmission time. We also need to make sure that the aggregated packet cannot exceed the maximum size (usually the maximum PAS packet size is set to be equal or less than

MTU). This checking action will be taken when trying to add a new packet to the queue.

The whole aggregation/deaggregation scheme is very similar to the bus transportation: the individual packets are just like people who are going to travel to certain destinations; and they take a bus according to some fixed routes (analogy to the routing path). At each station, (wireless nodes in the network), people (packets) get on the bus, (being put in the aggregation queue) and the bus waits for a same time before it leaves. After the bus reaches the next station, some people get off the bus and probably do a bus change, (packets being disaggregated and put into another queue). The only difference between our implementation and the real bus traveling scheme is that, we enforce every "passenger" to get off the "bus" and join a new waiting queue for the same "bus". The reason for this is to make the deaggregation/aggregaton process on the intermediate nodes simpler and easier to implement.

3.4 Performance Analysis

In this section we carry out analysis on the performance of our implementation. As stated, the main problem for VoIP packets in a wireless mesh network is that, the VoIP application usually generates a large number of small packets with a relatively large protocol overheads, which creates high overhead in transmitting the 802.11 headers and acknowledgements, waiting for separation DIFS and SIFS, and thus incurs contention for the medium.

We assume the average VoIP packet payload size is X bytes. For a typical VoIP packets, we have the following overhead:

- RTP/UDP/IP header: $12 + 8 + 20 = 40$bytes
- MAC header + ACK = 38 bytes
- MAC/PHY procedure overhead = Yμs)

Thus we can get the throughput function in Mbps:

$$T(x) = \frac{X}{Y + \frac{(78+X)}{B}}$$

For example, a typical VoIP packet at the IP layer consists of 40Bytes IP/UDP/RTP headers and a payload ranging from 10 to 100 Bytes, depending on the codec used. Therefore, the efficiency at the IP layer for VoIP is already less than 50%. At the 802.11 MAC/physical (PHY) layers, the performance degradation is much worse.

If instead, we combine N packets together into one packet, and assume the processing time at PAS is negligible, then the throughput function should be:

$$T(x) = \frac{NX}{Y + \frac{78+N(40+X)}{B}}$$

From above equation we can see that the main performance improvement comes from the reduced MAC layer overhead and less network contention. However, our implementation did not contain header compression and still have a considerable percentage of protocol header overhead. One possible factor that may affect the performance is that our implementation introduces extra delay for VoIP packets. But, the additional delay brought by packet aggregation is negligible; we will show and verify this claim in the experimental result section.

3.4.1 Integration of the path aggregation routing protocol and the packet aggregation system

In this section, we describe how the packet aggregation system integrates with the adaptive routing protocol mentioned in previous section. An overall picture of the whole system is shown in Figure 3.12. The packet aggregation system will use the adaptive routing protocol to probe the route to the destination, and collect and aggregate the VoIP packets. All the locally generated VoIP packets are intercepted by the packet aggregation system, and then the packet aggregation system will investigate the destination ip address contained in the ip header of these packets. RREQs are sent out and route discovery process are initiated if necessary. Those packets are buffered in a temporary queue before the destination returns the RREP. As soon as an available route is found, the packets are put into the aggregation queue. Those packets are sent out when the aggregation queue fulfills the two conditions mentioned in section 3.3.4. To summarize, the adaptive routing protocol is responsible for finding and returning the route while performing path aggregation, and the packet aggregation collects and aggregates the VoIP packets.

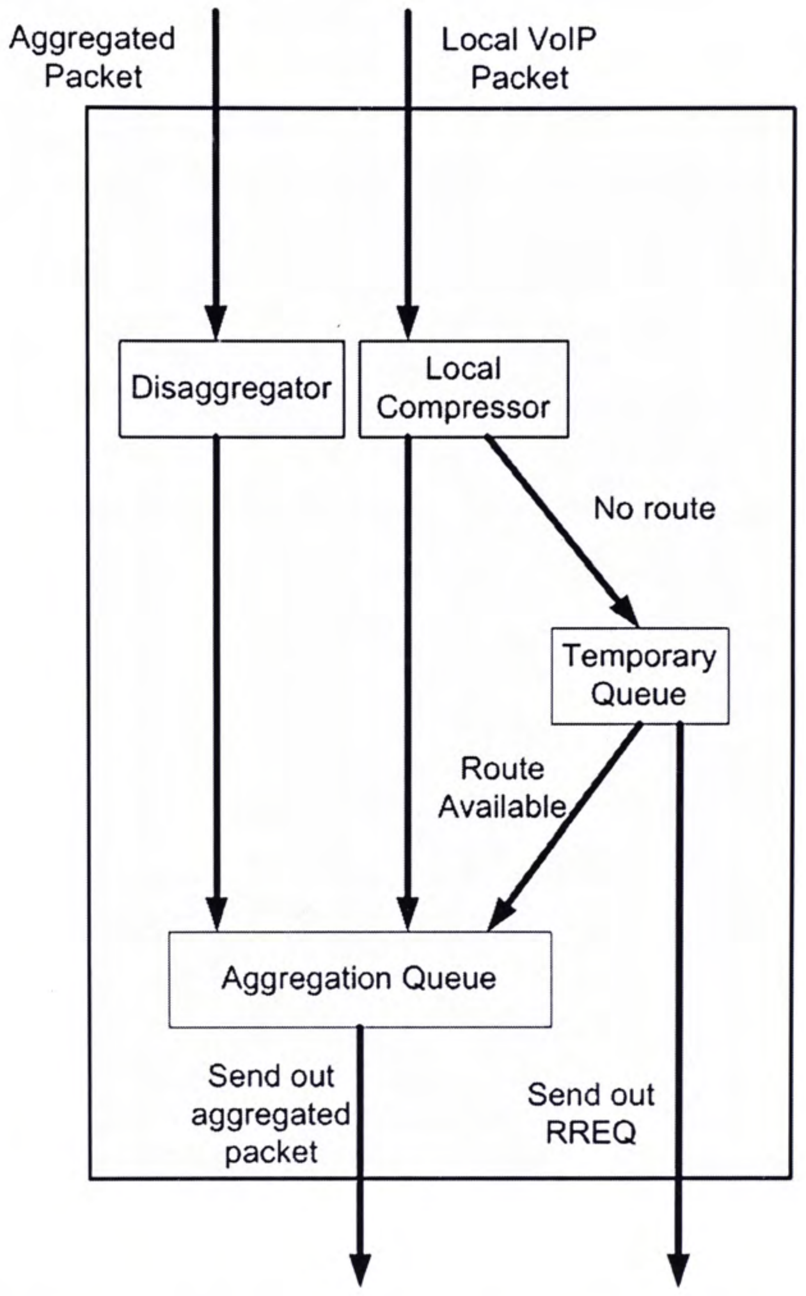

Figure 3.12: Path and Packet Aggregation System Structure

3.5 Performance Evaluation

3.5.1 Testbed Setup

To verify the performance gain, we carry out several experiments on a real testbed at which we implement the APA, PAS, ..., etc, and all the modules. We have deployed about ten table pcs which have Buffalo wireless cards installed as the wireless interface. Those nodes are installed with redhat linux (Kernel version 2.4.20-8) and hardcoded the routing paths. They are placed in a single room and works under the ad-hoc mode to form a small mesh network. Because of the actual limited available space and security reasons besides the fact that we need to frequently change the topology, we do not separate the nodes physically outside their transmission range if we want to disconnect them; instead, we use iptables on the nodes to filter out all the udp traffic on the AODV port. This method has the advantage that an arbitrary network topology can be formed and adjusted quickly without moving the machines. However, it is easy to see that this kind of "soft" separation cannot be considered as an alternative to the actual topology because the signal interference between the machines.

3.5.2 Packet aggregation

First we evaluate the performance of the single packet aggregation system. In these experiments, there is no path aggregation and the routing paths are hardcoded at each wireless node.

Parameter Setup

The parameters used in the experiments are described below. We compare the performance of the packet transmission with packet aggregation and without packet aggregation by flooding the network with small packets whose sizes range from 10 bytes to 100 bytes (which maps the actual packet size distribution of normal VoIP packet sizes in most popular VoIP codecs). For most experiments, we used the payload size of 40 Bytes. For most experiments set the maximum waiting time at each node to be 50ms, and the maximum aggregated packet size to be 1400bytes. (because the network MTU is 1500bytes, it will be meaningless to construct and send a packet larger than MTU) The maximum number of packets aggregated into one large packet is set to be 20, which conforms to the quotient of MTU and average packet size(28+40)(IP/UDP header plus payload size).

Performance Metric

We compare the wireless mesh network with packet aggregation and without packet aggregation in terms of packet loss rate and packet delay. The packet loss rate and delay are defined as:

- Packet loss rate: The percentage of packets lost during the whole transmission;
- Packet delay: The end to end delay of the packets;

We also measure the number of flows that can be supported when the different parameters change.

Experiment Results

The experiment results are shown below.

The parameters used in Figure 3.13 are: Packet interval = 10ms, MPN

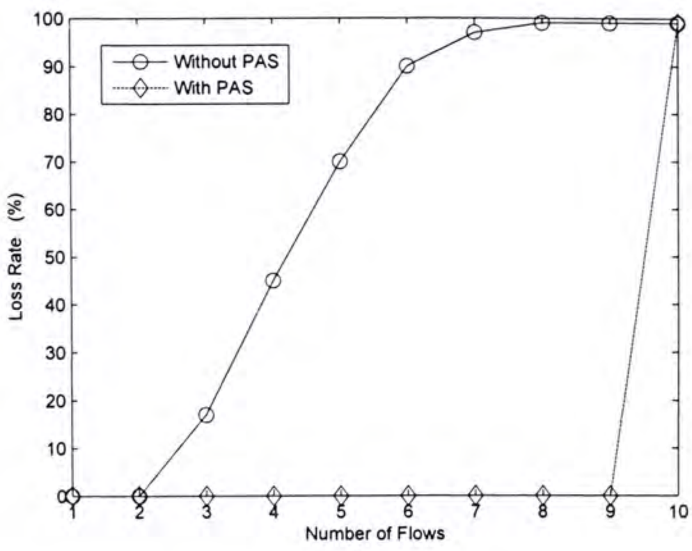

Figure 3.13: Number of flows supported vs. packet loss rate

= 5, Packet size = 40Bytes. We can see that the packet loss rate for PAS is very low compared to the scheme without PAS. Under the same constraint of delay and loss rate, i.e.(delay$\leq$200ms and loss rate$\leq$5%.)With PAS, the number of flows(calls) supported along a link are usually increased up to five times than the case without PAS given that the parameters are set optimally.

The parameters used in Figure 3.14 are: Packet interval = 10ms, MPN

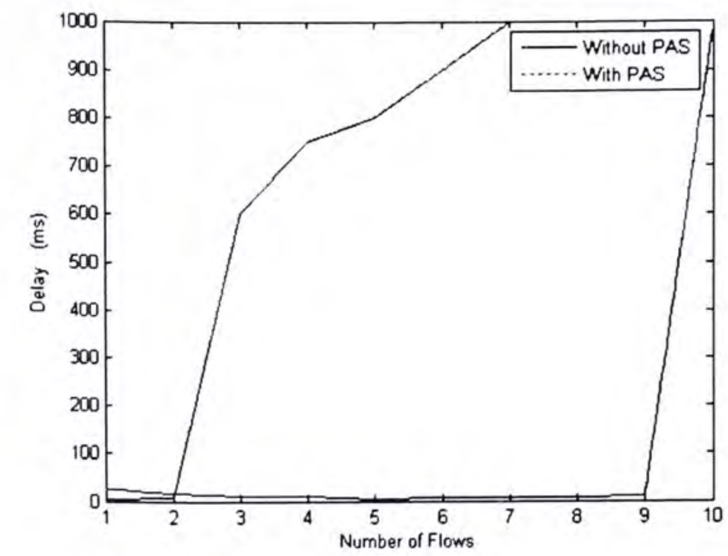

Figure 3.14: Number of flows supported vs. packet delay

= 5, Packet size = 40Bytes. It is shown that with the same number of flows, packet delay for PAS is lower than non-PAS when the flow number is greater than two. The relative higher delay of PAS when the flow number is small is reasonable; we will discuss more on this issue in later section.

The parameters used in Figure 3.15 are: Packet size = 40Bytes, MPN = 5, MWT = 50ms. When the packet interval increases, both PAS and non-PAS can support more flows. However, PAS always support more flows than non-PAS.

The parameters used in Figure 3.16 are: Packet interval = 10ms, MWT = 50ms, Packet size = 40Bytes. When PAS is active, the number of flows increases with the increase of MPN.

The parameters used in Figure 3.18 are: Packet interval = 10ms, MPN = 5, MWT = 50ms; Packet size = 40Bytes. Figure 3.18 depicts the number

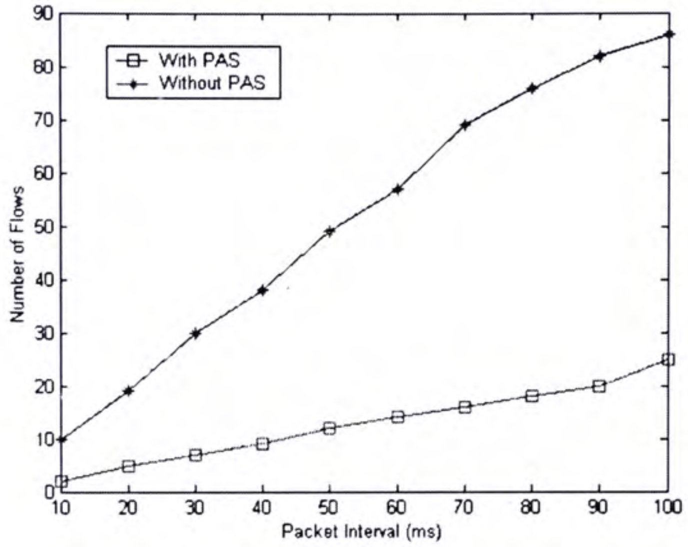

Figure 3.15: Number of flows supported vs. packet interval

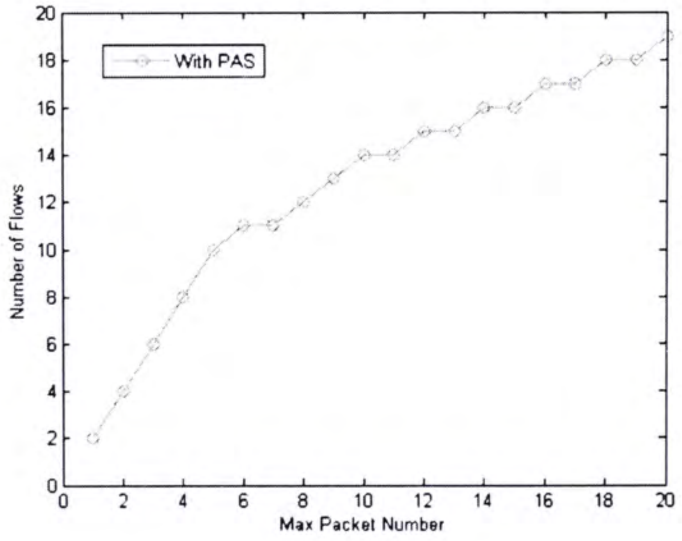

Figure 3.16: Number of flows supported vs. packet MPN

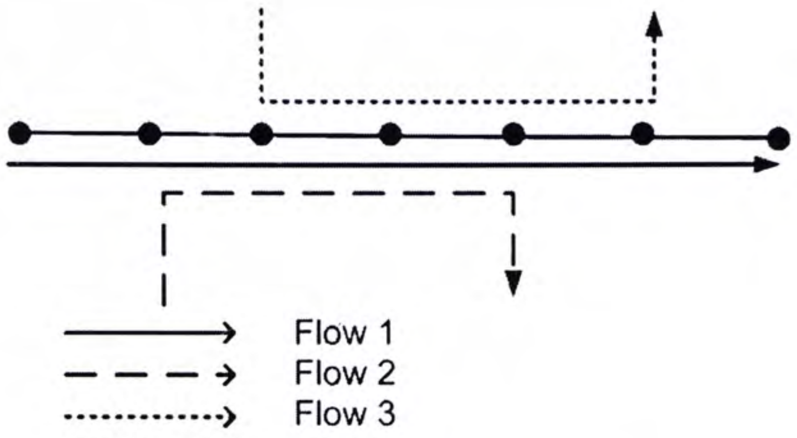

Figure 3.17: Multiple Flows: PAS vs. non-PAS

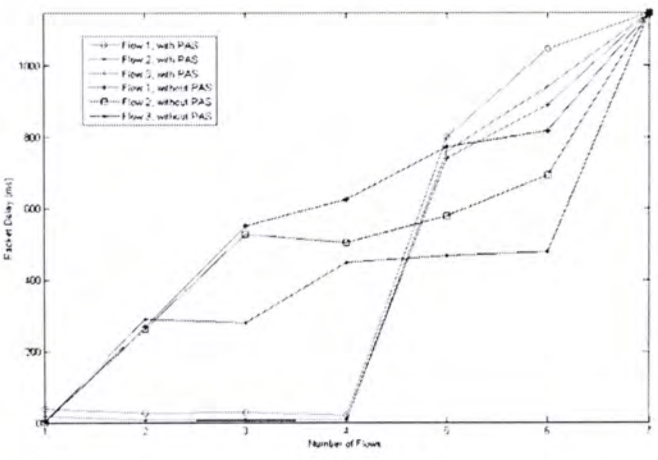

Figure 3.18: Multiple Flows: PAS vs. non-PAS

of flows that can be supported along different paths when there are multiple flows in the network. The topology of the network used in this experiment are shown in Figure 3.17. It can be seen from the figures that when there are same number of flows in the network, PAS has lower packet delay at most of the time.

Discussion:How to tune the PAS parameters to achieve the best performance?
There are three main parameters in PAS:

- The maximum waiting time at each aggregation queue,
- the maximum aggregated packet size,
- the maximum number of small packets aggregated into a single PAS packet. We will give a detailed discussion on the fundamental factors that should be considered when tuning these parameters.

- The maximum waiting time (MWT) for the aggregation queue: when a packet is added to the aggregation queue, it may need to wait for a certain amount of time for the queue to collect "enough" packets before it is sent out. The value of MWT cannot be set too high as transmission delay will accumulate along the nodes on the transmitting path; nor shall be it too small or there will not have enough packets to be aggregated. There is a trade off between less waiting delay and higher aggregation ratio. It is easy to see that the minimum value for the MWT at the aggregation node should be equal to PACKET_INTERVAL. We also need to consider the maximum delay the VoIP users can tolerate.As stated in [9], the maximum delay for VoIP should be less than 200ms to get a satisfactory call quality. So for our experiments, MWT is set to be 50ms.

- The Maximum aggregated packet size (MAP): We can't put into the PAS packet with infinite number of small packets because the presence of MTU (Maximum Transmission Unit) at the link layer. A packet which is larger than MTU will be split into several smaller packets before they get transmitted, and the MTU by default is set to be 1500 Bytes on most systems. It is clear that we should keep the size of PAS packets being less or equal to MTU. Also considering the MAC/RTS/IP/UDP overhead, we set the MAP to be 1400.

- Maximum number of small packet aggregated into one PAS packet(MPN). This is another important parameter whose value has a significant influence on the performance of PAS. MPN has the similar trade off to MWT. Assume every flow has the same packet generation rate: A proper value for the maximum number of small packet in one PAS packet may be $\frac{AverageNumberofFlows*MWT}{PacketInterval}$. The reason for above equation is, when MPN is near the result, packets from all flows can be aggregated and the aggregated packet will be sent at the same rate as the original packets. We use a value ranging from 5 to 20 as the default value for most experiments.

Analysis of the experiment results

We observed the following interesting results when carrying out the experiments:

- It seems that the system can only transmit a fixed number packets per second.

- The accumulated delay on the transmitting path first decreases as the number of flows increases, then start to increase after the number of flows increases to a certain value.

Explanation of the first phenomenon In [27] the authors also observed the similar phenomenon. Because of the minimum time needed to transmit a packet (no matter how large the payload is). Even if the payload size is 0, the network interface needs some time because of the physical/MAC layer overhead. From the experiments we get the result that, for a packet payload size of 10 bytes and packet interval = 10 ms, with max aggregation number = 20 and max packet size = 1400, without PAS, the number of flows supported when $lossrate \leq 10\%$ and $delay \leq 400ms$ is 3; with PAS, we should be able to support much more flows. In fact, PAS can support up to 17 flows under the above condition.

We can see that the network bottleneck is not at the data rate of the wireless links, instead it is the low sending rate of the physical/mac headers. Usually the transmission rate of the physical/MAC headers is at 1Mbps, while the payload sending rate ranges from 2Mbps to 24Mbps. The transmission time spent on the payload is very small compared to the transmission time on physical/MAC overhead. So in order to increase the throughput, what we can do is to increase the payload size as much as possible, which is exactly the purpose of the path aggregation.

Explanation for the second observation: For a given max aggregation number, before the number of flows reaches MPN, packets have to wait up to MWT

(For the worst case, it will be a single flow) and thus the total accumulated delay on the path is relatively high; as the number of flows increases, aggregation is done more quickly at each intermediate node. Aggregation queues don't need to wait until MWT expires to be sent out. So the delay is decreasing. When total number of flows raises up to the max aggregation number, that means the packet do not need to wait at the intermediate nodes' aggregation queue, and thus reaches the minimum delay. No extra delay is introduced by PAS now. The delay will be at the trend of increasing again when there are enough flows to make the network capacity saturated. For schemes without PAS, delay increases as the number of flows roughly with a linear relationship, when the number of flows exceeds certain threshold, the delay jumps to very high value probably because there are too many flows to cause transmission buffer overflow.

Summary

From the above experiment results we can see that under various topologies and experiment parameters, although PAS introduces a little bit protocol overhead to the packets and at each hop, and sometimes a little delay is also introduced; the resulting performance is still much better than the scenario that without packet aggregation. The great advantage of packet aggregation should comes from the reduced physical/MAC layer overhead and probably less network contention. Accordingly, the smaller the VoIP packet size is, the more obvious PAS's advantage becomes. Transmission with packet aggregation can outperforms the one without packet aggregation by at average five times in terms of number of flows supported. Moreover, to get the above result, at this stage we only aggregate the packets at the application layer, if we can apply more optimization techniques like header compression, it is possible that the performance can be better. With PAS, the delay even decreases when there are not sufficient flows to aggregate. The latency and PAS overhead we added at each aggregation queue almost impose no influence to the final performance because they are small enough. The experiments are carried out under good network conditions, for those network which severely suffers from packet loss and contention, the advantage of PAS should be more significant.

3.5.3 Combined scenario: path and packet aggregation

In this section, we describe the experimental setup used to profile the integrated system's performance with both path aggregation algorithm and packet aggregation. We also built a real experimental testbed which consists of about ten wireless nodes and carried out experiments on various network topologies.

Parameter Setup

The experiments are conducted on linux (Redhat 2.4.20-8) machines with buffalo wireless card installed. The machines are distributed across the engineering building of CUHK and form different topologies. However, sometimes we need to set the filtering rules in iptables to disable few links to get the expected topology. We developed a traffic generator to produce synthesized VoIP packets. In most of the experiments, we use a sample packet length of 40 bytes and set the packet interval to be 10ms. The maximum number of packets that can be integrated into a large packet is six, and the maximum queuing time is 30ms. The largest aggregated packet size is set to be 1400 bytes, which is near the MTU.

Performance Metric

For a given source-destination pair, we measure the maximum number of flows that can be supported under a certain delay constraint (typically 200 ms). Then we compare it to the scheme that uses AODV as the routing protocol.

Experiment Results

- Experiment 1: Single source-single destination and multiple flows
  The topology used in experiment 1 is shown in Figure 3.19 and Figure 3.20. For the same source-destination pair, we generate multiple flows (Which forms a flow group), and measure the number of flows that can be supported. We first carry out the experiment using AODV as the routing protocol, then repeat the experiment on the same topology using APA instead. Figure 3.19 shows the route of the flow group from node 9 to node 14 under APA, and Figure 3.20 shows the route under AODV. The routes taken by the flow groups under AODV and APA are shown in the tables. The number of flows that can be supported are also presented in the tables. We can see that even taking the same route, APA can support eight flows while AODV can only support three.

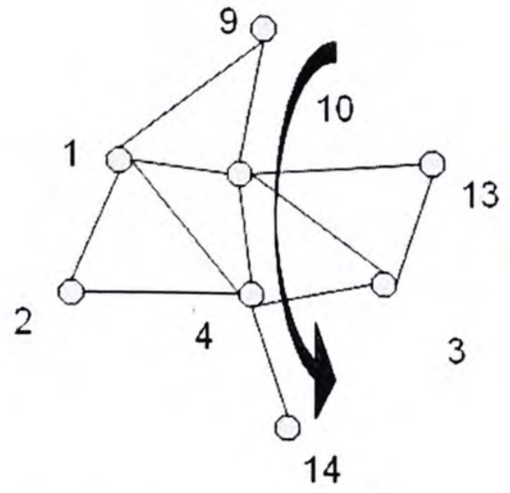

Figure 3.19: Experiment 1: Route taken by APA

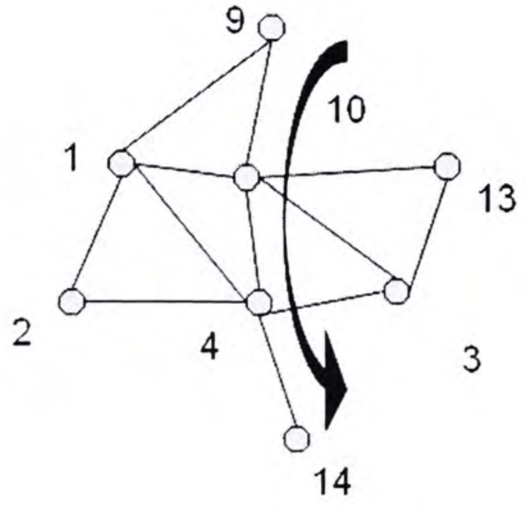

Figure 3.20: Experiment 1:Route taken by AODV

- Experiment 2: Multiple sources-single destination and multiple flows
  In this experiment we compare the performance of AODV and APA on the same topology with multiple sources and single destination flows.

AODV	9 - 10 - 4 - 14
APA	9 - 10 - 4 - 14

Table 3.1: Experiment 1:Route Taken

AODV	APA
3	8

Table 3.2: Experiment 1:Number of flows supported

The topology used in experiment 2 is shown in Figure 3.21 and Figure 3.22. Results are shown in following tables. We can see that for both

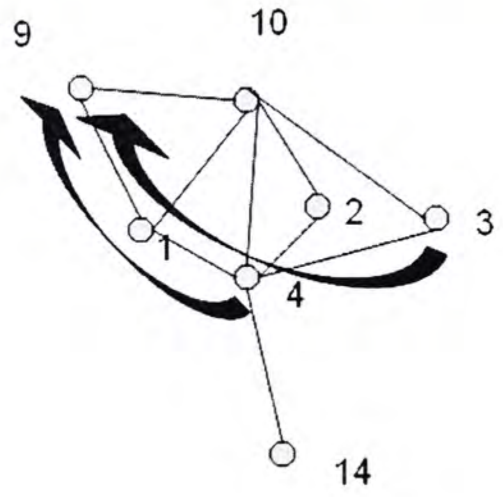

Figure 3.21: Experiment 2:Route taken by APA

flow group one and two, APA can support five and five flows while AODV can only support two and two, respectively.

- Experiment 3: Single source-multiple destinations and multiple flows
  In this experiment we compare the performance of AODV and APA on

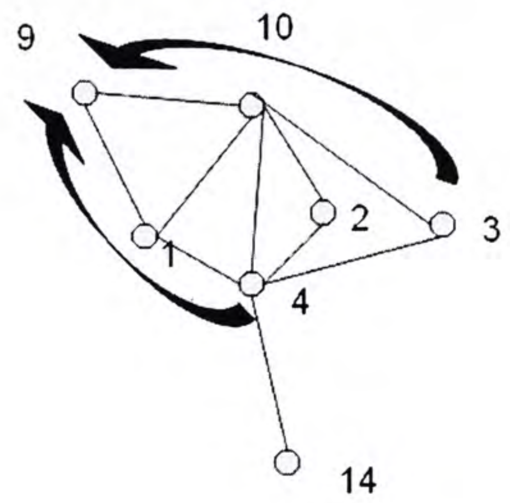

Figure 3.22: Experiment 2:Route taken by AODV

	Flow Group 1	Flow Group 2
AODV	4-1-9	3-10-9
APA	4-1-9	3-4-1-9

Table 3.3: Experiment 2:Route Taken

	AODV	APA
Flow Group 1/Flow Group 2	2/2	5/5

Table 3.4: Experiment 2:Number of flows supported

the same topology with single sources and multiple destination flows. The topology used in experiment 3 is shown in Figure 3.23 and Figure 3.24. Results are shown in following tables. We can see that for both flow group one and two, APA can support four and four flows while AODV can only support one and one, respectively.

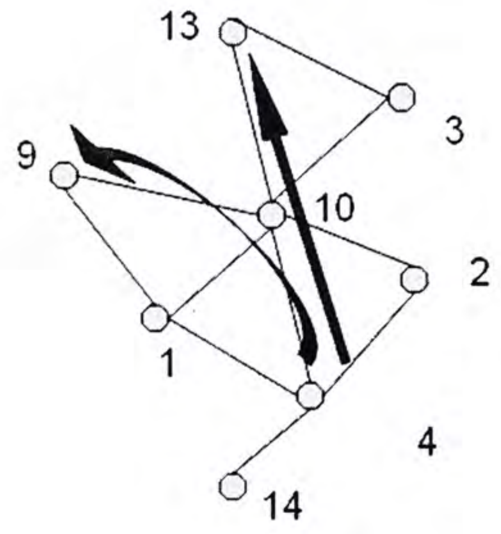

Figure 3.23: Experiment 3:Route taken by APA

	Flow Group 1	Flow Group 2
AODV	4-10-13	4-1-9
APA	4-10-13	4-10-9

Table 3.5: Experiment 3:Route Taken

	AODV	APA
Flow Group 1/Flow Group 2	1/1	4/4

Table 3.6: Experiment 3:Number of flows supported

- Experiment 4: Multiple sources-multiple destinations and multiple flows
  In this experiment we compare the performance of AODV and APA on the same topology with single sources and multiple destination flows. The topology used in experiment 4 is shown in Figure 3.25 and Figure 3.26. Results are shown in following tables. We can see that for flow

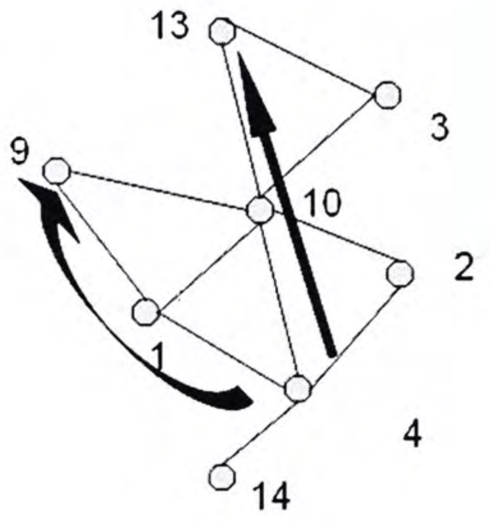

Figure 3.24: Experiment 3:Route taken by AODV

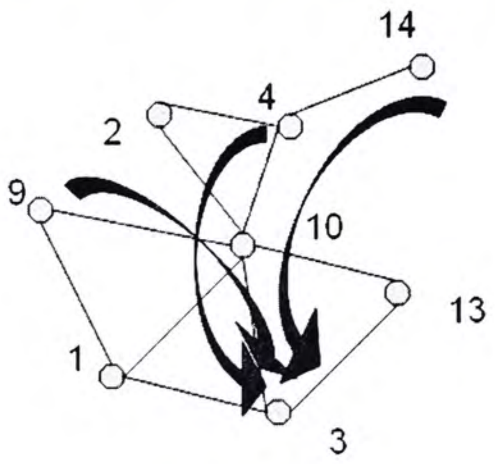

Figure 3.25: Experiment 4:Route taken by APA

group one, two, three, APA can support two, two and and one flows while AODV can only support one, one and one flow, respectively.

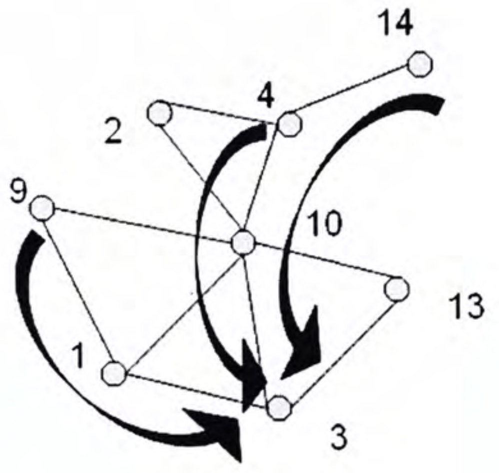

Figure 3.26: Experiment 4:Route taken by AODV

	Flow Group 1	Flow Group 2	Flow Group 3
AODV	14-4-10-13	4-10-3	9-1-3
APA	14-4-10-13	4-10-3	9-10-3

Table 3.7: Experiment 4:Route Taken

	AODV	APA
Flow Group 1/Flow Group 2/Flow Group 3	1/1/1	2/2/1

Table 3.8: Experiment 4:Number of flows supported

3.6 Summary

From above experiment results we can see that in terms of maximum number of supported flows under the given delay constraint(200ms), APA performs no worse than AODV and under most situations, outperforms AODV from 100% to 400%, depending on the network flow and topology. The performance gain mainly comes from reduced number of actual flows and packet overhead.

Chapter 4

Network Coding System in wireless network

4.1 Overview

In this chapter we present our network coding system, a simple implementation of the network coding theory. The basic idea of network coding is to perform an XOR operation on the packets from different sources, then broadcast the encoded packet to all the possible receivers. Network coding exploits the shared medium nature of the wireless network, and try to reduce the actual number of transmissions by encoding the packets.

4.2 System Architecture

4.2.1 Packet Format

The implemented network coding system works on the application level and encapsulates the original packet by adding a header at the front of the IP header. The structure of the encoded packet is illustrated in Figure 4.1:

The packet header contains following fields:

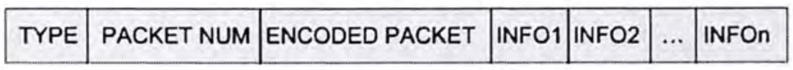

Figure 4.1: Encoded Packet Format

- TYPE: The first byte of the packet header indicates the type of the packet, either ENCODED or ORIGINAL. If an intermediate node receives an encoded packet, it will try to decoded it first; otherwise the packet will be treated as a normal packet and forwarded to the upper layer directly.

- PACKET NUM: The second field of the packet header is of one-byte long, which represents the number of encoded packets in this mixed packet. Note that only the encoded packet will have this field.

- PACKET ID: In order to prevent the packet reordering and ensure that encoded packets can be correctly decoded, we need to specify an id to every original packet. Every time when the source node send out a a packet, it needs to increment the id (Whose value is stored in a local variable) by one, and specify it in the packet header. Note that only the original packet has this field. For the encoded packet, the same field represents the packet number.

- INFO: At the end of the encode packet, several INFO structures are appended to contain the information about the encoded packet inside this mixed packet. The INFO structure consists of three sub-fields: ID, SRC_IP, DST_IP. Each INFO structure corresponds to one encoded packet in the mixed packet. The id subfield indicates the id of the original packet, while the SRC_IP and DST_IP represent the source and destination ip address of the original packet, respectively.

4.2.2 Encoding and decoding

Packet encoding

To encode the packets from different sources, we need to keep a buffer to temporarily accommodate the packets to be encoded. The buffer is implemented as a FIFO packet queue, and the intermediate node will periodically check the queue for coding opportunities. If there are several packets that can be encoded into a single one, these packets are extracted from the queue and encoded into XORed version. The mixed packet header fields are also filled in with proper value. Then the encoded packet will be broadcasted to all the near-by neighbors. The are several principles when performing encoding:

- The packets in the encoding queue will never wait for a given time period which is a tunable parameter. That means when we keep the packets in the coding queue to wait for coding opportunities, we should not make the queuing delay too much. The packets which cannot find other packets to be encoded together will only wait for a small time period (i.e. 10 ms), then it will be directly forwarded to next hop in the format of original packet.

- Similar to COPE in [26], we prefer to encode packet of similar sizes. At the end of the shorter packet, zero paddings are added. The reason for doing this is to ensure encoding packets of the similar length can increase the utilization of the encoded packet.

Packet decoding

When a node receives an encoded packet, it needs to decode it first. First it should go through the INFO structures in the encoded packet to find out whether the node itself is the next hop of the packet. If it is, then the node needs to decode it. The decoding process is as follows: assume the received packet is encoded using n original packets, the decoding node should make use of the INFO structure and retrieve the n-1 original packets from the packet buffer except the packet which is destined to itself. Then the decoding node needs to perform XOR operation again with the other n-1 original packets and the encoded packet to get the original packet.

Coding Algorithm

We present in this section the coding algorithm.

Algorithm 5 Coding Procedure

```
Pick packet $p$ from the coding queue
if $size(p) > 100$ bytes then
    $largeQueue \Leftarrow 1$
end if
for $Neighbori = 1$ to $M$ do
    Pick packet $p_i$ from coding queue $Q(i, largeQueue)$
    $p = p \oplus p_i$
end for
return $p$
```

4.3 Performance Evaluation

To evaluate the performance of the implemented network coding system, we performed some experiments on our own testbed. In this section, we begin with describing the experiment setup and performance metric, then we present the experiment results.

4.3.1 Experiment Setup

We employed a five-node wireless testbed which is built across the first floor of the Engineering Building, CUHK. The following experiments are run on 802.11. All the wireless nodes are installed with buffalo wireless card and run redhat linux(Kernel version 2.4.20-8). The network coding system is implemented in the user space. We also implemented a traffic generator to produce constant bit rate network traffic.

4.3.2 Performance Metric

Our experiments use the network throughput gain as the performance metric, which is defined as the ratio of the measured network throughputs with and without our network coding system. We compute the throughput gain from two consecutive experiments, one with coding turned on, another without coding.

4.3.3 Experiment Results

Alice and Bob Topology

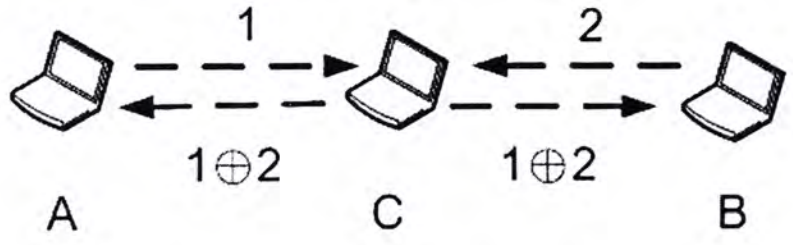

Figure 4.2: Experiment 1: Linear Topology

The classical Alice-and-Bob topology is shown in Figure 4.1. We generate traffic at both ends(Node A and B) and measure the end to end throughput. The node A and B both buffer their sent packets. The intermediate node C XORs the packets received from A and B, then broadcast encoded packets to A and B. Thus A and B can decode the encoded packet using their buffered sent packets.

Figure 4.3 depicts the throughput gain of the network coding system,

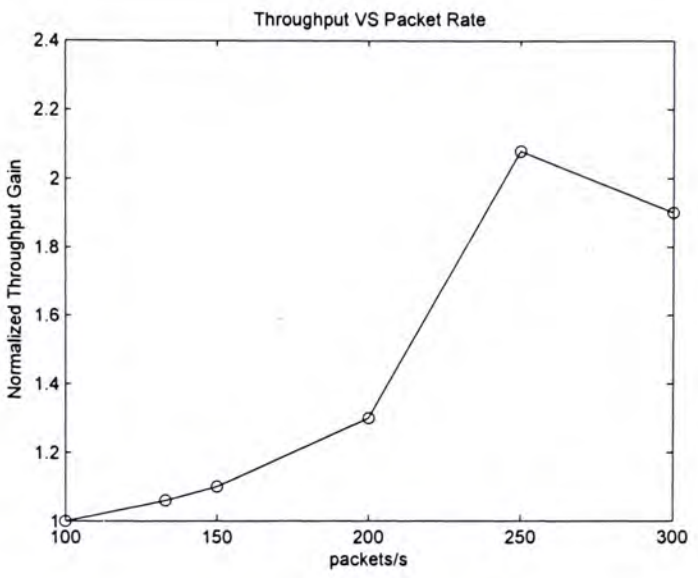

Figure 4.3: Throughput gain in linear topology

with the throughput of the case without network coding normalized to one. We can see from the result that network coding system brings a significant throughput gain, which varies with the packet rate. In the experiments we use a packet size of 40 bytes, and packet rate ranging from 100 packets per second to 300 packets per second. We can see from the graph that when the traffic demand is low, the benefit of network coding is not significant because the coding chance is low and the medium is not busy; as the traffic demand increases (packet rate increases), coding chance increases as well as network contention. The advantage of network coding becomes more and more obvious because the the scenario without coding suffers from network contention and packet loss due to the collision.

Cross Topology

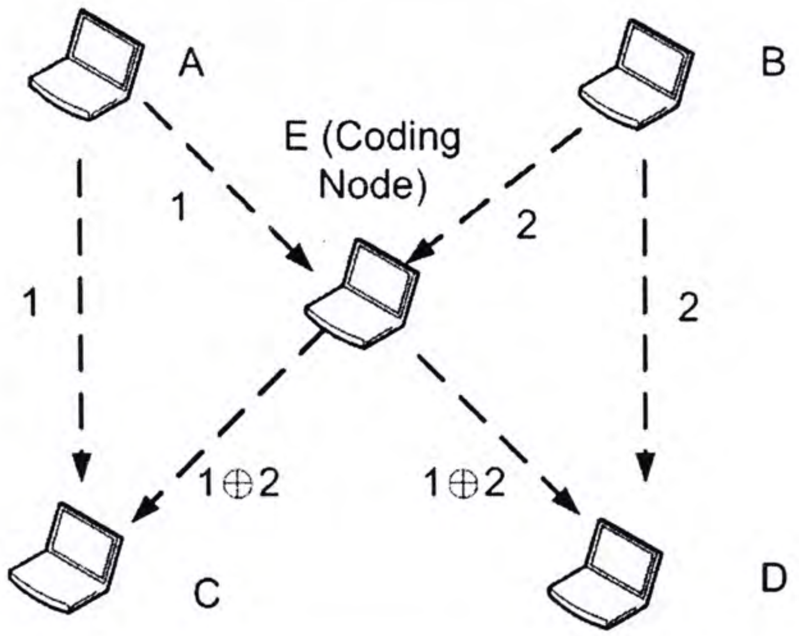

Figure 4.4: Experment 2: Cross Topology

This experiment measures the performance of the network coding system in the cross topology. The cross topology is shown in Figure 4.4 . Node A sends traffic to node D, and node B sends traffic to node C. The intermediate node E performs the network coding. The throughput gain is shown in Figure 4.5.

Similar to the Alice-and-Bob topology, the throughput in network coding system outperforms the scenario without coding. The normalized throughput gain is as much as 27% at its peak. One may wonder why we have less throughput gain as the Alice-and-Bob topology. One possible explanation could be that, the node C needs to overhear every packet from node A to de-

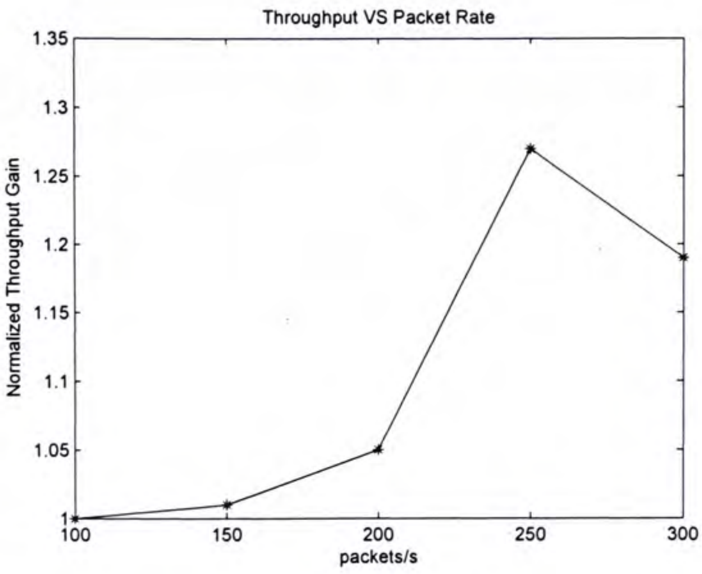

Figure 4.5: Throughput gain in cross topology

code the packet received from E, but in the real network, C is not guaranteed to be able to overhear all the packets from A, thus affects the throughput. But for the Alice-and-Bob topology, the sender always has the packets itself sent out, so they do not have this problem.

Wheel Topology

This experiment measures the performance of the network coding system in the wheel topology. The wheel topology is shown in Figure 4.6 . Node A sends traffic to node C, node B send traffic to node A, node C send traffic to node B. The intermediate node E performs the network coding. The throughput gain is shown in Figure 4.7.

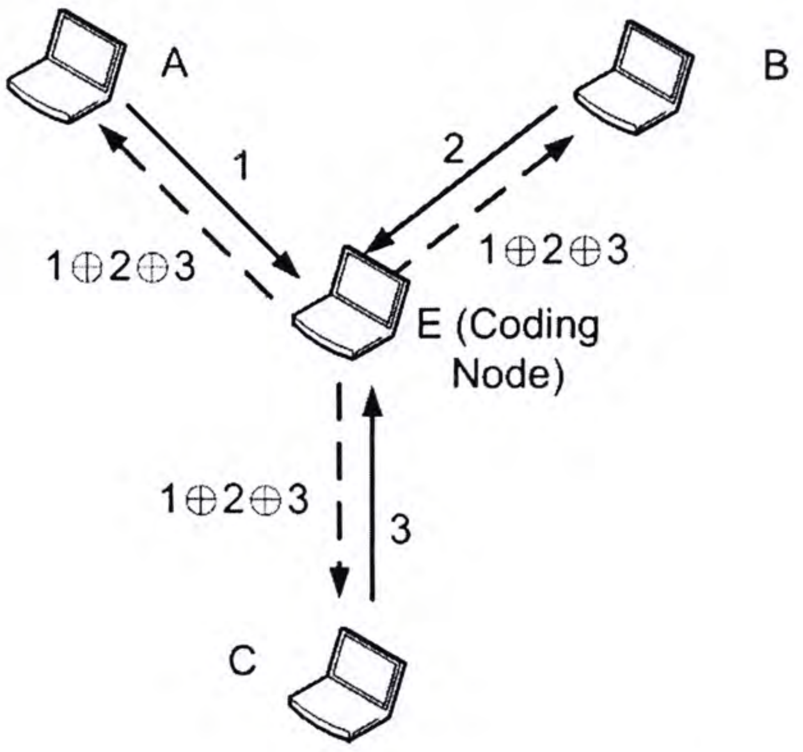

Figure 4.6: Experment 3: Wheel Topology

Similar to other topologies, network coding brings throughput gain. The throughput gain is as much as 45% under the best scenario.

Different packet rate vs. coding fraction

Under the optimal case, we assume all the flows that to be coded share the same packet rate. However, in practice, there are always flows with different packet generating rates. It is interesting to us to study the effect of asymmetric packet rate (traffic load) on the fraction of the packets that will be coded. We carry out the experiment by intentionally unbalancing the traffic load on different flows in the cross topology. This is done by gradually

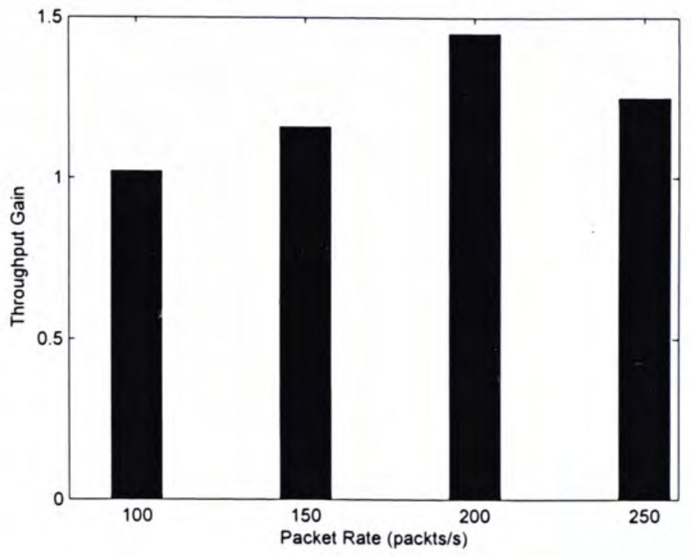

Figure 4.7: Throughput gain in wheel topology

adjusting the packet rate of one flow, and repeat the experiment and the measurement. Fig. 4.8 shows the coded packet fraction as a function of the ratio of the packet rate of the two flows. As the ratio of packet rate increases, one flow gradually occupies the queue in the coding node, reducing coding opportunities, and the aggregate network throughput. The effect of coding fraction (percentage of packets got coded) on the overall throughput is shown in Figure 4.9. It is shown that more packets got coded together, more throughput gain we can get; on the contrary, biased packet rate leads to less overall throughput.

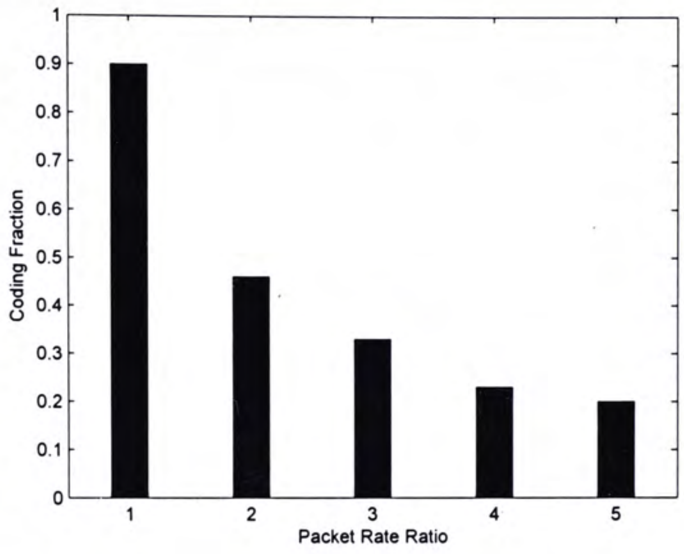

Figure 4.8: Coded packet fraction vs. ratio of packet rate

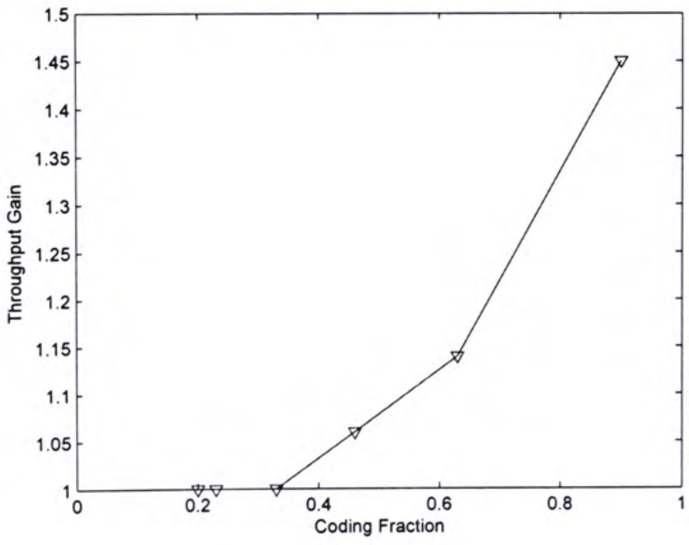

Figure 4.9: Throughput gain vs. coded packet fraction

Distribution of number of coded packets

It is another interesting question to find out the distribution of number of the packets coded together at the coding node. In order to investigate this, we carry out our measurements on the topology shown in Figure 4.10. There are in all six flows to be coded at the centering node, and we measure the PDF of number of packets coded together. Results are shown in Figure 4.11. We can see from the graph that most packets are coded in a number of four or five, which implies the theoretical coding gain will be very good.

4.4 Summary

From above experiments we can see that the implemented network coding system brings a significant performance improvement, which varies with network topology and traffic pattern. However, the current system is just an prototype of the implementation on network coding concept, there is still much space left for further improvement. Optimizations such as the more "intelligent" coding algorithm and better coding scheduling may be applied in future to make it work even better.

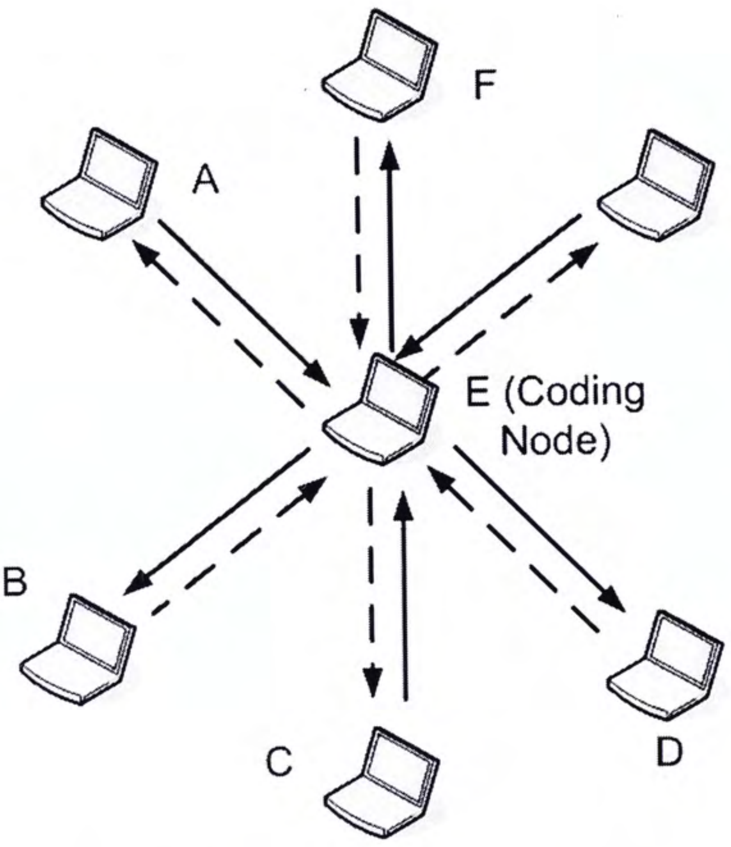

Figure 4.10: Wheel topology used to measure PDF of number of packets coded

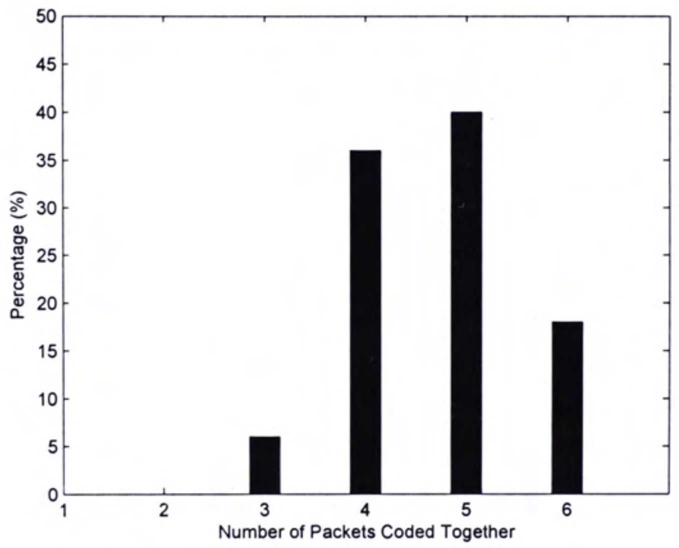

Figure 4.11: PDF of number of packet coded together

Bibliography

[1] M. Radimirsch A. Banchs, X. Perez and H. Stuttgen. Service differentiation extensions for elastic and real-time traffic in 802.11 wireless lan. May.

[2] J. Shi A. Ramamoorthy and R. Wesel. On the capacity of network coding for wireless networks. *In 41st Annual Allerton Conference on Communication Control and Computing.*

[3] Mike Neufeld Dirk Grunwald Ashish Jain, Marco Gruteser. Benefits of packet aggregation in ad-hoc wireless network. *Technical Report.*

[4] S. Biswas and R. Morris. Opportunistic routing in multi-hop wireless networks. *ACM SIGCOMM*, 2005.

[5] Samir R. Das Charles E. Perkins, Elizabeth M. Royer and Mahesh K. Marina. Performance comparison of two on-demand routing protocols for ad hoc networks. *IEEE INFOCOM conference*, 2000.

[6] J. Bicket D. S. J. De Couto, D. Aguayo and R. Morris. A high-throughput path metric for multi-hop wireless routing. *In ACM MobiCom 03, San Diego, California.*

Chapter 5

Conclusions and Future Directions

In this thesis, we investigated the performance and system implementation issues in wireless mesh networks.

First, on the packet and path aggregation concepts, we build a practical system which implements the combination of packet and path aggregation. The system deploys an adaptive routing protocol to improve the chance of path aggregation. Different active flows in the network will gradually converge and tend to share the same path for packet aggregation. The system serves as a daemon and collects and aggregates packets which share the same next hop. The performance of the system is evaluated by various experiments on different network topologies and traffic patterns. The experiment results show that the packet and path aggregation system significantly increases the VoIP performance in the wireless mesh network.

After that, we build a prototype of network coding system. The system perform XOR operations on packets from different flows when there is coding chance. The network coding system exploits the shared medium feature of the wireless network, and by coding packets together, the number of transmissions is reduced because we can send multiple packets in a single transmission. We evaluate the performance of the network coding system in a real testbed by measuring the ratio of throughput improvement. Experiment results show that the network coding system greatly increases the network throughput.

There are a number of issues remain to be studied. There could be further improvement on the adaptive routing protocol of path aggregation, to find better tradeoff between path length and packet aggregation chance. The packet aggregation can employ more advanced header compression techniques to improve the network utility. For the network coding system, finding better coding and queue scheduling algorithms could be an important future direction.

[7] R. Koetter M. Medard E. Ahmed D. S. Lun, N. Ratnakar and H. Lee. Achieving minimum-cost multicast: A decentralized approach based on network coding. *IEEE INFOCOM*, 2005.

[8] Kyungtae Kim Rauf Izmailov Dragos Niculescu, Samrat Ganguly. Performance of voip in a 802.11 wireless mesh network. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings.*

[9] Jack Y. B. Lee H. P. Sze, Soung C. Liew and Danny C. S. Yip. A multiplexing scheme for h.323 voice-over-ip applications. *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, 20(7).

[10] T. Ho and R. Koetter. Online incremental network coding for multiple unicasts. *In IDIMACS Working Group on Network Coding*, 2005.

[11] D. Hole and F. Tobagi. Capacity of an ieee 802.11b wireless lan supporting voip. *In Proceedings of IEEE ICC*, 2004.

[12] Belding-Royer Ian D. Chakeres, Elizabeth M. Aodv routing protocol implement design. *ICDCS Workshops*, 2004.

[13] Weilin Wang Ian F. Akyildiz, Xudong Wang. Wireless mesh networks: a survey. *Computer Networks*, 2005.

[14] S. Biswas J. Bicket, D. Aguayo and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. *In ACM MobiCom*, 2005.

[15] B. Karp. Geographic routing for wireless networks. *PhD thesis, Harvard University*, 2000.

[16] R. Koetter and M. Medard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 2003.

[17] Z. Li and B. Li. Network coding: The case for multiple unicast sessions. *Allerton Conference on Communications*, 2004.

[18] R. Guillier M. Heusse, F. Rousseau and A. Duda. Idle sense: an optimal access method for high throughput and fairness in rate diverse wireless lans. *In ACM SIGCOMM*, 2005.

[19] N. Venkitaraman R. Sivakumar P. Sinha, T. Nandagopal and V. Bharghavan. Wtcp: A reliable transport protocol for wireless wide-area networks. *Wireless Networks, 8(2-3):301C316*, 2002.

[20] S. R. Li R. Ahlswede, N. Cai and R. W. Yeung. Network information flow.

[21] J. Padhye R. Draves and B. Zill. Comparison of routing metrics for multi-hop wireless networks. *In Proceedings of ACM SIGCOMM*, 2004.

[22] S. F. Midkiff R. O. Bladwin, N. J. Davis IV and R. A. Raines. Packetized voice transmission using rt-mac, a wireless real-time medium access control protocol. *Mobile Comput. Commun. Rev.*, 5(3), 2001.

[23] T. Ho D. R. Karger R. Koetter D. S. Lun M. Medard S. Deb, M. Effros and N. Ratnakar. Network coding for wireless applications: A brief tutorial. *IWWAN*, 2005.

[24] R. W. Yeung S. R. Li and N. Cai. Linear network coding. *In IEEE Transactions on Information Theory*, 2003.

[25] A. Estepa T. Fei L. Gao R. Guerin J. Kurose S. Tao, K. Xu and D. Towsley. Improving voip quality through path switching. *In Proc. of IEEE INFOCOM.*

[26] Wenjun Hu Dina Katabi Muriel Medard Sachin Katti, Hariharan Rahul and Jon Crowcroft. Xors in the air: Practical wireless network coding. *In ACM SIGCOMM*, 2006.

[27] Masakatsu Kosuga Satoko Itaya and Peter Davis. Improving voip quality through path switching. *In Proceedings of the 24th International Conference on Distributed Computing Systems Workshops*, 2004.

[28] M. Iizuka T. Hiraguri, T. Ichikawa and M. Morikura. Novel multiple access protocol for voice over ip in wireless lan. *IEEE Int. Symp. Computers and Communications.*

[29] M. Medard D. Karger T. Ho, R. Koetter and M. Effros. The benefits of coding over routing in a randomized setting. *ISIT*, 2003.

[30] P. A. Chou Y.Wu and S. Y. Kung. Information exchange in wireless networks with network coding and physical-layer broadcast. *MSR-TR-2004-78.*