UNIVERSITY of CALIFORNIA
SANTA CRUZ

PROACTIVE, TRAFFIC ADAPTIVE, COLLISION-FREE MEDIUM ACCESS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

by

Vladislav V. Petkov

June 2012

The dissertation of Vladislav V. Petkov is approved:

Katia Obraczka, Chair

J.J. Garcia-Luna-Aceves

Ram Rajagopal

Venkatesh Rajendran

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Contents

List of Figures v

List of Tables vii

Abstract viii

Dedication xi

Acknowledgements xii

1 Introduction 1
- 1.1 Introduction . . . 1
- 1.2 Related Work . . . 5
  - 1.2.1 Contention Based MAC Protocols . . . 5
  - 1.2.2 Schedule Based MAC Protocols . . . 10
- 1.3 Contributions . . . 14

2 The Utility of Traffic Forecasting to Medium Access Control 16
- 2.1 Introduction . . . 16
- 2.2 Related Work . . . 18
  - 2.2.1 Schedule Based MAC Protocols . . . 18
  - 2.2.2 Traffic Forecasting . . . 19
  - 2.2.3 Quality of Service . . . 20
- 2.3 Benefits of Traffic Forecasting . . . 21
- 2.4 Forecasting Traffic . . . 25
- 2.5 Challenges and Future Work . . . 29

3 Characterizing Network Traffic Using Entropy 30
- 3.1 Introduction . . . 30
- 3.2 Datasets . . . 32
  - 3.2.1 Real-time flows . . . 34
  - 3.2.2 Media streaming flows . . . 35
- 3.3 Self-similarity . . . 36
  - 3.3.1 Self-similarity related work . . . 39
  - 3.3.2 Self-similarity results . . . 40
- 3.4 Packet timing entropy estimation . . . 43

3.4.1 Entropy rate . . . 43
3.4.2 Multiscale plug-in packet timing entropy estimator . . . 44
3.4.3 SampEn entropy estimator . . . 47
3.4.4 Entropy estimation related work . . . 47
3.5 Entropy-Based Traffic Complexity Analysis . . . 48
3.5.1 PPTEn estimator results . . . 48
3.5.2 SampEn estimator results . . . 55
3.6 ARMA and Entropy based Predictors . . . 57
3.7 Applications of entropy estimator . . . 59
3.8 Conclusion . . . 61

4 Design of a Traffic Forecasting Medium Access Control Protocol 62
4.1 Introduction . . . 62
4.2 Related Work . . . 64
4.3 TRANSFORMA . . . 66
4.3.1 Protocol Overview . . . 66
4.3.2 TRANSFORMA's Control Plane . . . 68
4.3.3 Traffic Forecaster . . . 70
4.3.4 Flow Selection: Scheduling Medium Access . . . 73
4.4 Performance Evaluation . . . 76
4.4.1 Experimental Setup . . . 76
4.4.2 Hot Spot Topology . . . 79
4.4.3 Multihop Grid Topology . . . 82
4.4.4 What about 802.11e? . . . 86
4.5 Conclusion . . . 86

5 Implementation of TRANSFORMA 88
5.1 The Starix 1201 Development Platform . . . 88
5.2 Implementation Goal . . . 89
5.3 Firmware . . . 91
5.3.1 USB Communication . . . 91
5.3.2 Establishing Superframes . . . 93
5.3.3 Data Transmission . . . 95
5.4 Driver . . . 96
5.4.1 Handling Outgoing Data . . . 97
5.4.2 Interfacing To Forecaster . . . 98
5.4.3 USB Communication . . . 98
5.5 Forecaster Daemon . . . 100
5.6 The journey of an outgoing packet . . . 100
5.7 Experiments . . . 102
5.7.1 Heterogeneous Flows . . . 102
5.7.2 VoIP With Background Traffic . . . 104
5.8 Conclusion . . . 109

6 Conclusion and Future Work 113
6.1 Future Work . . . 115

Bibliography 116

List of Figures

2.1 Simulation scenario: each node randomly generates traffic for its 1-hop neighbors. 22
2.2 Average queueing delay of IEEE 802.11, DYNAMMA, and DYNAMMA-PRED. 23
2.3 Average delivery ratio of IEEE 802.11, DYNAMMA, and DYNAMMA-PRED. 24
2.4 Percentage of time spent sleeping in DYNAMMA, and DYNAMMA-PRED. 24
2.5 Cumulative distribution function plots showing packet inter-arrival time and packet size. There are around 5 discrete inter-arrival times and 3 discrete packet sizes. 26
2.6 Forecaster output: prediction of slot period required to service a flow that produces lowest delay and wasted slots. Given that the slot duration was set at $638.125\mu s$, a slot period of 16 corresponds to $10.210ms$ and this is exactly the inter-arrival time indicated by the lowest dark cluster in the top plot of Figure 2.5. 27
2.7 Average delay incurred by the experts forecasting algorithm. 28
2.8 Slots used and wasted by the forecaster over time. 28

3.1 CDFs of packet inter-arrival times for real-time flows and media streaming flows 37
3.2 Packet arrival patterns for VoIP flows, video conferencing flows, and media streaming flows 38
3.3 Whittle estimator Hurst estimates of (a) CAIDA flow (known to be self-similar), (b) iChat audio flow, (c) iChat video flow and (d) Skype audio flow. 41
3.4 Abry Veitch estimator Hurst estimates of (a) CAIDA flow (known to be self-similar), (b) iChat audio flow, (c) iChat video flow and (d) Skype audio flow. 42
3.5 Entropy estimates (a) and equivalent probability (b) of 2 types of synthetically generated flows. Using larger word lengths reduces the entropy estimate of the dataset because packet arrivals in a flow are not independent of each other. 49
3.6 Entropy estimates (a) and equivalent probability (b) of synthetic flows as a function of time interval, τ. 50
3.7 Entropy estimates of VoIP and video conferencing flows using $m = 15$. 53
3.8 PPTEn estimates of media streaming flows using $m = 15$. 54
3.9 SampEn estimates of real-time flows with SampEn parameter $m = 2$ 56
3.10 SampEn estimates of media streaming flows with SampEn parameter $m = 2$ 57
3.11 Cumulative distribution functions of the relative error using ARMA predictors. 58
3.12 ARMA estimated relative error. 59

4.1 TRANSFORMA's superframe structure 66
4.2 The TRANSFORMA beacon holds neighborhood information of the sending node and forecast summaries that need to be known by all nodes within 2-hops of the sender. . . . 69

4.3 The **share algorithm** used by TRANSFORMA 71
4.4 The data rate forecast compared to the measured instantaneous data rate of a Skype flow. 73
4.5 Average delay (a), packet delivery ratio (b), and total goodput (c) for heterogeneous flows in hot spot topology 77
4.6 Per flow delays using TRANSFORMA (a), DYNAMMA (b), and 802.11 DCF (c) in hot spot topology 78
4.7 Two topologies used in our experiments. 79
4.8 Traffic delay (a) and delivery ratio (b) for Skype foreground traffic. Goodput (c) for Skype (foreground) flows and background flows. 80
4.9 Average delay (a), packet delivery ratio (b), and total good put (c) of all heterogeneous flows using grid topology 83
4.10 Per flow delays using TRANSFORMA (a), DYNAMMA (b), and 802.11 DCF (c) in multihop grid topology 84
4.11 The order in which flows are added to the 4 by 4 grid 85

5.1 The Starix 1201 Development Platform 89
5.2 Implementation block diagram 90
5.3 TRANSFORMA's superframe structure 94
5.4 The journey of the first outgoing packet in a flow. 101
5.5 Heterogeneous flow experiment network setup 103
5.6 Average delay of heterogeneous flows in each direction going over a TRANSFORMA link. 105
5.7 Average delay excluding first 5s of heterogeneous flows in each direction going over a TRANSFORMA link. 105
5.8 Delivery ratio of heterogeneous flows in each direction going over a TRANSFORMA link. 107
5.9 Goodput of heterogeneous flows in each direction going over a TRANSFORMA link. . . 107
5.10 VoIP experiment network setup 108
5.11 Average delay across TRANSFORMA link seen by flows over time 110
5.12 Average goodput across TRANSFORMA link seen by flows over time 111

List of Tables

3.1 A taxonomy of common network applications. Applications whose traffic will be studied in this chapter are marked with "*". 33
3.2 Network traces that form our dataset 34
3.3 Summary of Hurst estimator results. For each flow we run 7 Hurst parameter estimators over a variety of aggregation levels. The results are plotted and in this table each plot is summarized by a (-), (0), or (+), indicating **no** self-similarity, **maybe** self-similarity, or **yes** self-similarity, respectively. A (*) indicates that $< 50\%$ of the points fell outside of the 0 to 1 range, and a (**) indicates that $> 50\%$ of the points fell outside of the 0 to 1 range (this is an indication that the estimator is not robust for this data and calls into question its output) 40
3.4 ARMAX model-based prediction quality 58
3.5 Train and test errors of Entropy based predictor 59

5.1 Delay values of Figure 5.6 106

Abstract

Proactive, Traffic Adaptive, Collision-Free Medium Access

by

Vladislav V. Petkov

Wireless networks are a fixture of present day computing. We are seeing a simultaneous increase in network density and throughput demand as the clients of these networks grow accustomed to more data hungry applications. Contention-based channel access methods take bigger performance hits and waste more energy as network density and load increases. It is therefore clear that the future of wireless networking will need to exploit some form of schedule based channel access in order to simultaneously solve the problems of energy consumption and maximization of channel utilization.

The focus of this work is on leveraging implicit properties of network traffic to benefit the performance of schedule based medium access mechanisms. We focus on one of these properties: the packet arrival behavior of the traffic. We chose to start our work by trying to answer the following question: "If we use predictions of the behaviors of flows in the network, can we decrease the delay in schedule-based medium access control?" The main idea is to use traffic forecasting to anticipate transmission schedules instead of establishing them reactively, i.e., as traffic arrives at the MAC layer. Although not all applications generate forecastable traffic, we contend that many applications do. Examples of predictable network traffic include Voice-over-IP (VoIP) applications such as Skype, iChat, and Google talk. Video streaming applications have lower QoS demands but also contain many predictable patterns. All of these applications are becoming increasingly commonplace in the home networks of today.

An experimental method was used to evaluate the benefit that accurate traffic prediction could have on the performance of a schedule based MAC protocol (DYNAMMA). Comparing the performance of DYNAMMA to our modified version of it (DYNAMMA-PRED) in simulations showed that predic-

tion does improve delay performance of the schedule based protocol significantly, particularly at lower network loads.

The next step was to address the topic of extracting patterns out of packet arrival times of each flow with more mathematical rigor. We did this by measuring the entropy of packet arrivals in a network flow. Given that entropy is defined as the "measure of information" [1], its value in this context signifies the amount of pattern in the packet arrival times of a flow – the less information each arrival holds, the more pattern there is overall.

During our investigation of the entropy of the packet arrival times, our research produced the concept of an "entropy fingerprint" – a plot of the entropy of the packet arrival times of a flow over a range of time scales. Each entropy fingerprint has numerous characteristics that are related to the packet arrival behavior of the flow that generated it. These fingerprints can be used in many ways, such as identifying what application generated the flow or whether the flow's packet arrivals are likely to be regular or irregular at a given time scale. In addition to the entropy fingerprints, the entropy estimator that we developed turned out to be usable as forecaster as well, able to predict the chances of a packet arrival in the next slot.

Analyzing the entropy fingerprints of various types of traffic confirmed that there was useful information in the packet arrival times of network flows that could be leveraged in a schedule based MAC protocol. Furthermore, the work presented us with a traffic forecaster that we could use to extract this information for use in such a MAC protocol.

Following this work, we designed a medium access control protocol to embody the fusion of traffic forecasting with a schedule based access control mechanism. We called the protocol TRANSFORMA, which stands for TRAffic FORecasting Medium Access. TRANSFORMA was designed using the principle that the MAC layer should detect the properties of each flow transparently and adapt its level of service accordingly. TRANSFORMA attempts to do that by observing an application flow, learning its pattern, and forecasting the flows future behavior based on the observed one. In its current imple-

mentation, TRANSFORMAs forecaster examines the packet arrival process of each application flow and determines the corresponding per-flow inter-packet arrival times. It then uses this information to establish the flows medium access schedule. TRANSFORMA operates under the assumption that applications that place more stringent requirements, e.g., higher data rates and delay sensitivity have forecastable network usage patterns. The simulation results show that TRANSFORMA significantly improves on the delay performance of its predecessor, DYNAMMA.

The final contribution of this work is a real-system implementation of TRANSFORMA. This fully functional network link enables experimentation with real application traffic, serves to validate our simulation results and demonstrates that the concepts embodied in TRANSFORMA are practical.

To my parents:

Thank you for always believing in me! Now you've earned some bragging rights!

To my wife:

Thank you for your support and patience! I love you!

Acknowledgements

I would like to thank Katia Obraczka for being such a supportive and great advisor! Thank you, Venkatesh, for helping shape the idea that turned into my thesis. Thank you, Ram, for your valuable contribution! Thank you, JJ, for providing great insights into all things networking. Thank you, Fred and Inanc, for your generous contribution of resources and manpower to enable me to bring TRANSFORMA into reality on the Starix STX1201.

The text of this dissertation includes reprints of the following previously published material:

- V. Petkov and K. Obraczka, "The case for using traffic forecasting in schedule-based channel access," in *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, Jan. 2011, pp. 208 –212.
- V. Petkov, R. Rajagopal, and K. Obraczka, "Characterizing per-application network traffic using entropy," in *19th IEEE MASCOTS Symposium*, 2011.
- V. Petkov, K. Obraczka, "Collision-Free Medium Access Based on Traffic Forecasting", in *13th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, June 2012

The co-authors listed in these publications directed and supervised the research which forms the basis for the dissertation.

Chapter 1

Introduction

1.1 Introduction

Wireless networks are a fixture of present day computing. Ever since the release of the original IEEE 802.11 standard, IEEE 802.11-1997, and its subsequent amendments, the user base of wireless networks has expanded and and the networks have blanketed almost every corner of the spaces we occupy in our homes, offices, and even daily commutes. Presently, we are seeing a simultaneous increase in network density and throughput demand as the clients of these networks grow accustomed to more data hungry applications. In addition, the proliferation of mesh networks will mean that wireless networks are beginning to be used for more than just the *last-hop*. Wireless bandwidth, however, remains a precious resource and can not be increased without bound to meet these demands. The only way to allow emerging data-intensive applications to thrive is to more efficiently utilize all the wireless bandwidth at our disposal.

Contention-based channel access methods, like IEEE 802.11, take bigger and bigger performance hits due to an increasing number of collisions as network densities increase. These collisions are a normal part of contention-based channel access and are solved through retransmission. In contrast,

schedule-based channel access methods remove collisions by design by giving each node a transmission opportunity that all other nearby nodes respect and don't interfere with. Therefore, as network densities increase, schedule based channel access methods continue to utilize the channel fully without collisions. Furthermore, contention based approaches have inferior energy consumption properties to schedule based channel methods for two reasons. The first reason is the energy spent on transmissions that are not properly received due to collisions (the amount of energy wasted for this reason increases as the network becomes more dense). The second reason is that, due to the random nature of the channel access, nodes must spend some amount of time with their radios on and listening even when not actively receiving messages destined to them. It is therefore clear that the future of wireless networking will need to exploit some form of schedule based channel access in order to simultaneously solve the problems of energy consumption and maximization of channel utilization.

The focus of this work is on leveraging implicit properties of network traffic to benefit the performance of schedule based medium access mechanisms. The starting point was an existing state of the art schedule based medium access control protocol named DYNAMMA (DYNAmic Multi-channel Medium Access) [2]. DYNAMMA is a fully schedule-based channel access framework and has very good channel utilization and energy conservation characteristics. The only drawback in DYNAMMA is a high latency. The reason the latency is high is that slots are scheduled in DYNAMMA based on average traffic flow rates which have a tendency to lag behind packet arrivals.

Among the properties of network traffic, one of the most useful to medium access control is the packet arrival behavior of the traffic. We chose to start our work by trying to answer the following question: "If we use predictions of the behaviors of flows in the network, can we decrease the delay in schedule-based medium access control?" The main idea is to use traffic forecasting to anticipate transmission schedules instead of establishing them reactively, i.e., as traffic arrives at the MAC layer. Although not all applications generate forecastable traffic, we contend that many applications do. Examples of predictable network traffic include Voice-over-IP (VoIP) applications such as Skype, iChat, and

Google talk. Video streaming applications have lower QoS demands but also contain many predictable patterns. All of these applications are becoming increasingly commonplace in the home networks of today.

An experimental method was used to evaluate the benefit that accurate traffic prediction could have on the performance of a schedule based MAC protocol (DYNAMMA). Comparing the performance of DYNAMMA to our modified version of it (DYNAMMA-PRED) in simulations showed that prediction does improve delay performance of the schedule based protocol significantly, particularly at lower network loads.

The next step was to address the topic of extracting patterns out of packet arrival times of each flow with more mathematical rigor. We did this by measuring the entropy of packet arrivals in a network flow. Given that entropy is defined as the "measure of information" [1], its value in this context signifies the amount of pattern in the packet arrival times of a flow – the less information each arrival holds, the more pattern there is overall.

During our investigation of the entropy of the packet arrival times, our research produced the concept of an "entropy fingerprint" – a plot of the entropy of the packet arrival times of a flow over a range of time scales. Each entropy fingerprint has numerous characteristics that are related to the packet arrival behavior of the flow that generated it. These fingerprints can be used in many ways, such as identifying what application generated the flow or whether the flow's packet arrivals are likely to be regular or irregular at a given time scale. In addition to the entropy fingerprints, the entropy estimator that we developed turned out to be usable as forecaster as well, able to predict the chances of a packet arrival in the next slot.

Analyzing the entropy fingerprints of various types of traffic confirmed that there was useful information in the packet arrival times of network flows that could be leveraged in a schedule based MAC protocol. Furthermore, the work presented us with a traffic forecaster that we could use to extract this information for use in such a MAC protocol.

Following this work, we designed a medium access control protocol to embody the fusion of traffic forecasting with a schedule based access control mechanism. We called the protocol TRANSFORMA, which stands for TRAffic FORecasting Medium Access. TRANSFORMA was designed using the principle that the MAC layer should detect the properties of each flow transparently and adapt its level of service accordingly. TRANSFORMA attempts to do that by observing an application flow, learning its pattern, and forecasting the flows future behavior based on the observed one. In its current implementation, TRANSFORMAs forecaster examines the packet arrival process of each application flow and determines the corresponding per-flow inter-packet arrival times. It then uses this information to establish the flows medium access schedule. TRANSFORMA operates under the assumption that applications that place more stringent requirements, e.g., higher data rates and delay sensitivity have forecastable network usage patterns. The simulation results show that TRANSFORMA significantly improves on the delay performance of its predecessor, DYNAMMA.

From the beginning of the protocol design effort, the goal was to follow a simulation-based validation of the protocol with a real-system implementation. Some of the design decisions of the protocol were steered by this goal. A notable example is the super frame structure of TRANSFORMA – this was strongly motivated by the WiMedia MAC protocol [3], which became a strong candidate to be the basis for a TRANSFORMA implementation early in its design. The final contribution, the real-system implementation, is presented in Chapter 5.

In the remainder of this chapter is a summary of related work in contention-based channel access and schedule-based channel access. Following that, we outline the contribution of our work (Section 1.3).

1.2 Related Work

In order to put the challenges in creating and contributions of TRANSFORMA – a schedule-based MAC protocol – in context, in this section I will summarize some important work in the field of MAC (Medium Access Control) protocols. MAC protocols can be neatly categorized into two categories: contention based protocols and schedule based protocols. Some well-known examples of contention based protocols are ALOHA [4], CSMA [5], MACA [6], MACAW [7], IEEE 802.11 [8], and FAMA []. CATA [9], NAMA [10], LAMA [10], PAMA [10], HAMA [], TRAMA [11], FLAMA [12], and DYNAMMA [2] make up the list of equally well known schedule based protocols. The following two sections will overview these protocols in chronological order.

1.2.1 Contention Based MAC Protocols

The first of the contention based protocols, *ALOHA* [4] was the first broadcast radio wireless communication protocol [5]. ALOHA comes in two flavors: pure ALOHA, and a later-developed slotted ALOHA. ALOHA shares the broadcast medium in a very straight forward and simple manner. In pure ALOHA, when a node has a packet to send it is sent immediately. If another node begins to transmit its packet at any time during the transmission of the first node, all the packets are lost and both nodes try again. The only method of feedback regarding collisions that nodes have in ALOHA is in the form of positive acknowledgments from the receiving node: if an acknowledgment is not received, the packet is assumed to have been lost and must be retransmitted.

In slotted ALOHA time is divided into packet-length slots (packets are assumed to be of fixed length). A node can only begin its transmission at the beginning of one of these slots. With this modification, two or more nodes would only collide if they both have a packet to send in the same slot. The benefit of the slots can be intuited in two ways. The first way is to think of how much channel time is wasted by a collision; in slotted ALOHA this can be no longer that one slot length, whereas in pure

ALOHA this can be up to just shy of two packet lengths (equivalent to two slot lengths). The second way to intuit the benefit of the slots is to think of the "vulnerable" time of a packet. In pure ALOHA, a node's transmission will be unsuccessful if another node starts its own transmission anywhere between one packet time before it started to one packet time after it started. In other word's the vulnerable time is 2 packet lengths. In slotted ALOHA, that same interfering node would either have started its transmission in the previous slot (in which case it will not interfere) or it will start its transmission in the same slot. In other words, in slotted ALOHA the vulnerable time is only 1 packet length. Accordingly, the maximum throughput of slotted ALOHA is twice that of pure ALOHA.

The contention based protocol that would succeed ALOHA, as described by Tobagi et. al. in [5] was *CSMA* (Carrier Sensing Multiple Access). In the first part of his three-part paper series, Tobagi outlines several variants or CSMA "modes": non-persistent CSMA, 1-persistent CSMA, and p-persistent CSMA (of which 1-persistent is actually a special case). Each of the modes have the following properties: a terminal can be either sending or receiving at any given time, a terminal is "ready" if it has a packet to send, a simplifying assumption that time is slotted into slots of length τ (the propagation delay of the channel) holds and all nodes are perfectly synchronized.

Non-persistent CSMA obeys the rule that any terminal, upon becoming ready, will sense the medium at the start of the next slot and then either transmit if the channel was sensed idle or reschedule its transmission to some later time (making a retransmission behave much like a new arrival) if the channel is sensed busy. A p-persistent CSMA terminal, upon becoming ready, will sense the medium at the start of the next slot and either transmit if the channel was sensed idle or, with probability p, attempt to transmit during the next slot. The idea behind the p-persistence is that the probability p can be adjusted to give the best combination of idle time and interference (with a larger value of p decreasing idle time but increasing interference). 1-persistent CSMA has the negative property that if two terminals became synchronized and collided at some time slot t, would continue to be synchronized and collide again during time slot $t+1$, $t+2$ and so on. It is to prevent this situation that some degree of randomness

is desirable in contention-based MAC protocols.

Simulation results by Tobagi et. al. in [5] show that the most optimal p-persistent CSMA for a particular throughput will outperform non-persistent CSMA, however, not by very much. Both modes of CSMA outperform pure and slotted ALOHA as long as propagation delays are small (with large propagation delays, CSMA begins to perform poorly because it acts on stale information whereas ALOHA is immune to large propagation delays).

The ability to sense the channel before transmitting gives CSMA improved performance over ALOHA except in situations where its carrier sensing abilities fail. Carrier sensing fails in the presence of *hidden-terminals*. A hidden-terminal is present in the following scenario: there are three nodes arranged in a straight line, separated from each other by a distance equivalent to the propagation distance of their radios. The nodes on the ends of the line can only hear the node in the middle while the node in the middle can hear them both. The node on one end of the line is transmitting to the middle node when the node on the other end decides it too wants to transmit to the middle node. It senses the channel, finds it free, and proceeds to transmit causing a collision at the middle node.

Tobagi and Kleinrock evaluate the effects of hidden terminals on CSMA in [13] and show that the channel capacity of a CSMA controlled channel approaches that of a channel controlled by pure ALOHA in the presence of hidden terminals. They propose a protocol called *BTMA* (Busy Tone Multiple Access) to solve the hidden-terminal problem. The context in which BTMA is described is one of a single *station* and multiple *terminals* that can be hidden from each other. The communication channel is divided into two portions – a message portion and a busy-tone portion – which are not necessarily of equal size, i.e. the message portion gets most of the bandwidth. The station emits a busy-tone signal on the busy-tone portion of the channel when it detects a transmission from any of the terminals. Since all terminals are within range of the station, they can hear the busy-tone even if they cannot hear the transmissions of all other terminals. When a terminal becomes ready to transmit, it senses the busy-tone portion of the channel instead of the message portion as in CSMA, and hidden-terminal collisions are avoided. There

is a price to pay in BTMA for the busy-tone channel. The presence of noise on the busy-tone channel means that detecting the tone in this ”narrow-band” channel can be tricky. The wider the bandwidth given to the channel, the more detectable the tone will be, but the more bandwidth will be taken away from the message channel.

The literature is a little thin on where *CSMA/CA* has its roots, but the DCF (Distributed Coordination Function) mode of IEEE 802.11 builds upon its concept and is often referred to when CSMA/CA is talked about in the literature. The CA in CSMA/CA stands for collision avoidance and consists of two special messages: Request-to-Send (RTS) and Clear-to-Send (CTS). A sender precedes any transmission of data with an RTS message and proceeds to transmit its data only after receiving a CTS message. Even terminals hidden from the sender but within range of the receiver will hear the CTS and defer their own transmission, preventing a collision.

Phil Karn's *MACA* (Multiple Access with Collision Avoidance) [6] is strongly based upon CSMA/CA. In the amateur radio context in which Karn considers his protocol, there is a new challenge to overcome along with the hidden-terminal and that is the exposed-terminal. An exposed terminal occurs when a terminal that is too far away to interfere with another sender (i.e. it is out of range of that sender's receiver) defers its transmission because it can hear that other sender's transmission. In order to solve both exposed- and hidden-terminal problems simultaneously, Karn removes the carrier sensing (leaving only MA/CA) and makes MACA rely purely on its collision avoidance capability. MACA uses RTS and CTS packets to implement collision avoidance, with the sender including the amount of data it has to send in the RTS message and the receiver copying that information into its CTS message. Besides solving the hidden-terminal problem, the RTS/CTS exchange can solve the exposed-terminal problem because a ready terminal that hears another terminal's RTS and then does not hear a CTS can safely assume that it is out of contention range of that terminal and proceed to begin its own collision avoidance handshake and transmission. According to Karn, MACA is not a novel concept, merely a formalization of a commonly understood technique. He says it is essentially the same as BTMA except that the busy-

tone channel is multiplexed in with the data. MACA marks the appearance into the realm of wireless networks of the now popular binary exponential backoff algorithm for retransmission time.

Vaduvur Bharghavan and his research colleagues at Xerox Palo Alto Research Center (PARC) later enhanced Karn's MACA and, to honor the original name, called their protocol *MACAW* (Multiple Access Collision Avoidance for Wireless LANs) [7]. The first enhancement in MACAW was in the retransmission algorithm. MACA uses a simple binary exponential backoff algorithm in which the retransmission timer begins at 1 slots and is doubled after every unsuccessful transmission attempt and reset back to 1 after every successful transmission. This backoff scheme causes a high degree of unfairness in situations where one node grabs the channel and forces other contending nodes to repeatedly try to retransmit with smaller and smaller probability of success. in MACAW, the backoff scheme is augmented so that on transmission failure, the backoff timer is increased by a factor of 1.5 instead of 2 and upon success, instead of dropping down to 1 immediately, the timer is simply reduced by 1. Fairness is increased and backoff oscillation is reduced. In addition – given that the backoff timer's value is a function of the local congestion (and not just something at node itself) – nodes include the value of their backoff timer in their RTS and their neighbors adopt it.

Aside from improvements to the backoff strategy, the authors also implement multiple stream queues at each node, add an ACK (acknowledgment) packet to reduce the time taken to initiate retransmissions, add a DS (data sensing) packet to account for an exposed terminal situation where an RTS is sent to a node that is unable to respond with a CTS (and causes backoff counters to grow), and add an RRTS (Request for Request to Send) packet that can allow a node to contend on behalf of another node in certain situations. The authors note that the current scheme does not accommodate multicasting well and it remains an unsolved problem.

In 1997 the first draft of the IEEE 802.11 Wireless Local Area Network specification was completed. Crow et. al. wrote an article summarizing the features of the new standard [8]. In 802.11, a group of nodes that are communicating form the Basic Service Set (BSS). Within a BSS nodes are coordinated

using one of two modes: the Distributed Coordination Function (DCF) or the Point Coordination Function (PCF). The DCF is the more popular mode of 802.11 and is essentially CSMA/CA. The PCF mode is a polling technique that allows nodes to communicate in a contention-free manner under the management of a Point Coordinator (the access point). PCF is therefore not available in 802.11's ad-hoc mode, which most researchers are interested in.

The 802.11 standard defines three inter-frame spaces (IFS): the short IFS (SIFS), the PCF-IFS (PIFS) and the DCF-IFS (DIFS). During DCF mode, contending nodes must wait no less than a DIFS when contending for the channel. Once they have acquired it, they send their data packet and wait a SIFS for the ACK. A SIFS is the shortest IFS, so the ACK has "priority" over any new transmission. During the PCF mode, the point coordinator will wait a PIFS interval when contending for the channel. Since a PIFS is shorter than a DIFS, the point coordinator will beat out any other node to the channel. 802.11 has support for RTS/CTS exchange, not only to combat hidden-terminal problems, but also to prevent wasting time on collisions that involve large packets (the sender of which would continue sending until they sent the whole packet). RTS/CTS can optionally be turned on only for packets exceeding a certain size, for all packets, or not at all.

802.11 has two types of carrier sensing: physical and virtual. Physical carrier sensing works by analyzing all detected packets and via relative signal strength from other sources. In the header of every 802.11 packet there is duration information which receiving nodes use to update their Network Allocation Vector (NAV). The NAV indicates how long before the channel will be free. Virtual carrier sensing is done by looking at the NAV and deferring access if it is non-zero.

1.2.2 Schedule Based MAC Protocols

The *TRAMA* (TRaffic Adaptive Medium Access) protocol, is a schedule based protocol designed to reduce energy consumption[1] in sensor networks. Sensor networks are typically comprised of

[1] According to Rajendran et. al. TRAMA it is the first schedule-based protocol to do this.

large numbers of nodes of relatively low computational and storage ability that need to last for long periods of time solely on battery power. Wireless communications efficiency contributes in large part to the lifetime of a sensor network node. To achieve its high energy efficiency, TRAMA is designed to avoid collisions at receivers by electing a single transmitter at any time in a 2-hop neighborhood[2] and letting all nodes that aren't the intended receivers know they can go to sleep. In TRAMA, channel time is divided into two modes: a random access mode and a scheduled mode. Time is assumed slotted and nodes are assumed synchronized during both modes.

Random access mode is a contention-based mode. In order to accommodate topology changes, contention-based periods are present in all schedule-based protocols[3]. The random access mode is sized according to the expected density of nodes in the target application. During random access mode, nodes exchange their 1-hop neighbor information in beacons. After this process is finished, each node has 2-hop neighborhood knowledge.

The scheduled mode of TRAMA is when all data transmissions occur. In order to transmit data collision-free, a node determines its *SCHEDULE_INTERVAL*: the interval of time for which the node has data in its queue. The node then determines, with the help of a hash function similar to that used in [10], which slots during this interval it has highest priority for. It determines the receiver for each of these slots and encodes that information in its schedule message as a bitmap. Each winning slot gets its own bitmap, and each bitmap has a bit set for each of the node's neighbors that is an intended receiver of the transmission during the corresponding slot. Multicasting and broadcasting are easily supported. The last winning slot of the node is used to send the schedule information for the next interval and so on. There is a mechanism provided to give up unused slots so they are not wasted.

TRAMA's energy efficiency is compared to that of 802.11, CSMA, S-MAC and NAMA and found to be superior. It pays a big price in terms of delay. The delay incurred by TRAMA is almost

[2] A sufficient condition for collision freedom because it eliminates hidden terminal collisions.

[3] A notable exception is TDMA (Time Division Multiple Access) where the channel is divided into frames of N slots each, in a network of N nodes, and each node gets its own slot to use (or waste) at its discretion.

two orders of magnitude greater than that of the contention-based protocols. NAMA's delay is almost as large.

The most influential of the aforementioned schedule-based protocols to this work is *DYNAMMA* (DYNAmic Multi-channel Medium Access). With an attention to energy consumption similar to that in TRAMA, and more application independence than FLAMA, the DYNAMMA framework boasts the ability to adapt to application traffic patterns, provide collision-free multi-channel operation, and do it with reduced signaling overhead.

In DYNAMMA, time is divided into superframes, each of which is the same length and is broken up into periods of three different kinds of slots. The first of these is the signaling slot. Every superframe begins with a number of signaling slots. Each node gets its own signaling slot and uses it to announce its presence to its neighbors and exchange information about the traffic it has to send. A few of these slots are left vacant to be used by nodes joining the neighborhood. After the signaling slots come the base slots. A base slot is sized to fit the largest MAC frame size. For a maximum frame size of 4096 bytes, a base slot is the same size as 16 signaling slots. After the base slots the remainder of the superframe is filled by burst slots. These slots are larger in size than base slots and can fit multiple frames allowing a transmitter an opportunity to transmit multiple frames in a burst. Clearly, nodes in DYNAMMA have to be synchronized in order to conform to the superframe structure and among their other uses, the signaling slots provide this functionality.

As previously stated, all schedule-based protocols that wish to accommodate variable topologies must have a contention-based period. In DYNAMMA, signaling slots are the only contention-based element. When a node joins the network, it must choose a vacant signaling slot and begin transmitting its signaling packet in that slot in every subsequent superframe. It is easy to see that there exists a possibility that two nodes will join the network at the same time and choose the same signaling slot. DYNAMMA handles this situation in a manner very similar to that of the WiMedia MAC protocol: a node can detect that its signaling packet has collided by listening to its neighbors' signaling packets, and can then resolve

the collision by moving to a different signaling slot.

Once the nodes have settled into their neighborhoods, their signaling packets continuously propagate traffic information. Each node's signaling packet contains its own traffic information, as well as traffic information of each of its 1-hop neighbors. Traffic is treated as a series of 1-hop flows. Each flow has a source, a 1-hop destination, and a flow identifier. Furthermore, to be more adaptable to the application data characteristics, DYNAMMA has three flow classes: class 0 through class 2. Class 0 flows are allowed to compete for every data slot in a superframe, class 1 flows can only compete for half the slots, and class 2 flows can compete for a quarter of the slots. The determination of which class a flow falls in is a function of the number of flows contending in the neighborhood, the arrival rate of the flow, and the service rate of the flow. Flow information is summarized very succinctly using a bitmap format (previously seen in TRAMA). The flow bitmap has a bit for each neighbor of a node. The least significant bit of the flow bitmap corresponds to the first neighbor in the list of 1-hop neighbors. If the bit is 1, there is a flow for that neighbor, if 0 there isn't. The strength of the bitmap representation is that it lends itself very well to delivering multicast and broadcast information – previously mentioned as a lacking feature of all contention-based MAC protocols.

Once the traffic information is disseminated, each node can independently execute the distributed scheduling algorithm at the beginning of each data slot to decide whether it will become a transmitter, a receiver, or go to sleep during that slot. Channel selection is also done by the scheduling algorithm. To enable the distributed execution of this algorithm, DYNAMMA uses a pseudo-random function to generate a unique priority for each flow and select a channel. A very similar scheme is described by Garcia-Luna-Aceves et. al. and used in the NAMA, LAMA and PAMA MAC protocols [10].

The performance of DYNAMMA is terms of medium utilization and energy consumption is superior to that of IEEE 802.11 and TRAMA. The average delay packets incur is less than that of TRAMA but still significantly larger than that of IEEE 802.11 (playing a representative role of contention-based

MACs in general). It is here that TRANSFORMA finds motivation – to try to further reduce the average delay that still plagues schedule-based MAC protocols.

1.3 Contributions

In this section we summarize the contributions of our work.

The very first contribution of this work is an analysis of the behavior of a schedule-based MAC protocol that has accurate prediction. This prediction-simulating protocol is derived from the DYNAMMA framework and named DYNAMMA-PRED (for DYNAMMA with prediction). Chapter 2 describes the modifications made to DYNAMMA to form DYNAMMA-PRED and evaluates the results to show that, as expected, prediction can significantly reduce the delay suffered by the schedule-based protocol.

The dependence on traffic forecasting of our proposed Medium Access Control approach motivated us to seek a metric of predictability that we could ultimately use to quantify which types of network applications would be well served by our MAC. We developed a technique to generate "entropy fingerprints" of network applications that is described in Chapter 3. These entropy fingerprints have several characteristics that are unique to each network application and can be used to compare, among other things, the predictability of the application's traffic at various time scales.

We then designed a schedule-based MAC protocol we call TRANSFORMA (Traffic Forecasting Medium Access). In our protocol, each node uses a machine learning algorithm to predict the data rate of each outgoing flow. These predictions (or forecasts as they may otherwise be referred to in this thesis) are distributed to the neighboring nodes and a distributed scheduling algorithm that runs at each node uses this information to determine which flow will win any given slot. The protocol design and simulation evaluation are presented in Chapter 4.

Finally, we put together a fully functional TRANSFORMA network interface for a Linux ma-

chine. The implementation of this network interface consists of custom firmware for a Starix Technology development board, a companion Linux driver go with the firmware, and a forecaster application to perform all the traffic forecasting computation. The design of the implementation and experimental results are presented in Chapter 5.

Chapter 2

The Utility of Traffic Forecasting to Medium Access Control

2.1 Introduction

As a result of the wide availability and variety of wireless devices and the ubiquity of wireless communication infrastructure, a number of new applications and services have emerged – notably the "Smart Environments", which include the "Digital Home". One of the challenges imposed by Smart Environments in general, and the Digital Home in particular, is their high throughput and stringent quality-of-service (QoS) requirements.

As new physical layer technology such as highly integrated, small foot-print, single-chip radios combined with advanced coding and modulation techniques are able to support very high data rates, the fundamental limits challenging development and deployment of Digital Home applications have been shifting from the physical (PHY) layer to the medium access control (MAC) layer. Consequently, it is imperative to design efficient MAC techniques that will "expose" the underlying PHY's high data rates

to the applications while meeting their QoS requirements such as low delay and delay-jitter.

Random access (a.k.a., contention-based) channel access methods, such as IEEE 802.11's DCF, are known to have their performance deteriorate sharply as traffic load and network density increase. This degradation in performance is due to collisions that reduce channel efficiency and increase per-packet energy cost.

Schedule-based solutions to the medium access problem – such as those presented in [10], [15], [2] – eliminate collisions, improving channel efficiency substantially. Furthermore, using time slots to arbitrate medium access makes it possible to do more effective sleep-scheduling [2] and thus considerably improve energy efficiency. Dynamic schedule-based medium access approaches schedule data transmissions in response to application traffic. While these solutions maximize channel utilization by allocating channel time only to nodes that need it, they may exhibit longer data delivery delays. This extra delay is due to the fact that before data can be sent, traffic information needs to propagate so that nodes can schedule accordingly.

In this chapter we are interested in answering the following question: "If we use predictions of the behaviors of flows in the network, can we decrease the delay in schedule-based medium access control?" The main idea is to use traffic forecasting to **anticipate** transmission schedules instead of establishing them reactively, i.e., as traffic arrives at the MAC layer. Although not all applications generate forecastable traffic, we contend that many applications do, in particular most Digital Home services that have stringent QoS demands. Examples of predictable network traffic include Voice-over-IP (VoIP) applications such as Skype, iChat, and Google talk. Video streaming applications have lower QoS demands but also contain many predictable patterns. All of these applications are becoming increasingly commonplace in the home networks of today. As we move towards "smart" homes and offices, we argue that even more media-carrying, forecastable data streams will flow over the underlying networks.

The remainder of this chapter is organized as follows. We start by presenting an overview of related work in Section 2.2. Then, in Section 2.3, we present some preliminary results showing the

potential performance benefits traffic forecasting can bring to schedule-based channel access. We then describe a machine-learning based traffic forecasting technique using the expert framework ([16, 17]) in Section 2.4. We conclude with a discussion of open research issues and the direction of our future work.

2.2 Related Work

2.2.1 Schedule Based MAC Protocols

Contention-based medium access methods suffer from collisions that cause performance deterioration with increased load and also contribute significantly to the energy consumption of the radios. Schedule-based medium access methods on the other hand eliminate collisions and provide much higher delivery-ratio performance. Through structured use of the channel, they can also eliminate overhearing and idle-listening, which gives them very good energy saving properties.

The *TRAMA* (TRaffic Adaptive Medium Access) [11] protocol, is a schedule-based protocol designed to be bandwidth- and energy-efficient. TRAMA achieves energy efficiency by eliminating collisions and allowing nodes to sleep when they are not the intended receivers of the current transmission. While TRAMA achieves considerable energy savings (when compared to contention-based MAC protocols such as S-MAC), it incurs high delay.

DYNAMMA (DYNAmic Multi-channel Medium Access) [2], like TRAMA, gives attention to energy consumption and provides more application independence than FLAMA [12], a scheduled-access protocol designed specifically for sensor network applications. The DYNAMMA framework boasts the ability to adapt to application traffic patterns, provide collision-free multi-channel operation, while achieving reduced signaling overhead. DYNAMA's performance in terms of medium utilization and energy consumption is superior to that of TRAMA and contention-based protocols. However, while the average delay packets incur in DYNAMA is less than that of TRAMA, it is still significantly larger than that of IEEE 802.11 (which we use here as a baseline for contention-based MACs). Our goal is to

explore whether traffic forecasting can be used to reduce this delay commonly found in schedule-based MAC protocols. In Section 2.3 we use a modified version of DYNAMMA, which we call DYNAMMA-PRED, to show the potential improvement possible if traffic forecasting is used.

2.2.2 Traffic Forecasting

Dynamic schedule-based protocols achieve collision-freedom by exchanging some form of scheduling information in advance of actual data transmission. Transmission slots are allocated based on this information. Allocating more slots than the traffic needs wastes channel time, while allocating less slots than needed will increase the delivery delay. We propose to use a traffic forecaster at each traffic source that constantly adapts its forecast of how many slots and with what spacing will be required to best serve each data flow. The forecasts are disseminated to the two-hop neighborhood and used to schedule the medium access. A survey of the literature shows that traffic forecasting has not been used at the MAC layer in the way we are proposing.

In [18], the authors present a dynamic bandwidth allocation strategy that, combined with the use of the Renegotiated Constant Bit Rate (RCBR) service model can accommodate variable bit rate (VBR) video while reducing queue sizes and increasing network utilization. The method used is adaptive linear prediction minimizing mean square error. With the RCBR service model, a large frequency of bandwidth renegotiations corresponds to higher network utilization but also larger signaling overhead; the authors present some methods to control this tradeoff. In our forecast-enabled, schedule-based MAC we encounter a similar set of problems in terms of: allocating enough time slots to a flow to satisfy its bandwidth needs while trying to dampen fluctuations in the forecasts to reduce dissemination overhead.

In the work described in [19] the authors investigate several bandwidth allocation policies that can be used at the adaptation layer between ATM and IP networks. More specifically, the paper compares the performance of static algorithms (where bandwidth allocation does not change), periodic algorithms (where the bandwidth allocation is changed periodically) and adaptive algorithms (where the bandwidth

allocation is changed as necessary within some restrictions) is compared in the paper.

Koutsakis et al. have shown how traffic models can be used in call admission schemes to provide better quality of service when delivering video conferencing, MP3 downloads, and MPEG-4 streaming in cellular wireless networks [20],[21].

Traffic prediction or forecasting has also been used by Liu et al. in their work [22] to dynamically set the duration of a transmission opportunity (TXOP) in IEEE 802.11e in order to improve the quality of service given to variable bit-rate video.

2.2.3 Quality of Service

One of the benefits of a schedule-based MAC is the ease with which Quality of Service (QoS) guarantees can be made. IEEE 802.11 on the other hand struggles in this regard. There exists a large body of work concerned with adding QoS to IEEE 802.11. One notable example is the work described in [23] focusing on bandwidth management in ad-hoc networks. The solution presented there is a dynamic bandwidth management scheme that provides an application level solution to the problem of providing QoS in an ad-hoc network. Under the centralized control of a bandwidth manager entity, each node adapts its rate factoring in both the requirements of its applications and the available bandwidth. The whole framework runs on top of IEEE 802.11. Although the proposed dynamic bandwidth management is an effective method of providing QoS, it is complex and depends on a central management entity, which makes the system less flexible and prone to a central point of failure. The traffic forecasting, schedule-based MAC that we aim to develop as a follow up to this work will not only be able to deliver QoS guarantees without a central management authority, but it will be able to infer the QoS needs of applications and meet them without the need for any cross-layer awareness, greatly reducing the overall system complexity.

2.3 Benefits of Traffic Forecasting

In this section we quantify the impact of traffic forecasting on the performance of scheduled-based MACs. We do this by assuming that traffic can be forecast with 100% accuracy and evaluating the performance under this assumption. Assuming an optimal forecaster will give us an upper bound on the performance of forecast-based medium access.

Evaluation Methodology

We chose to use a simulation-based evaluation methodology to determine an upper bound on the performance of schedule-based medium access given perfect traffic forecasting. Specifically, the DYNAMMA protocol presented in [2] is adapted to use "oracle" traffic forecasting and its performance is compared to DYNAMMA's unmodified version.

DYNAMMA is a schedule based protocol. Each node that participates in the network synchronizes to its neighbors and broadcasts a beacon during its assigned beaconing slot at the beginning of each superframe (a construct commonly seen in schedule-based MAC protocols that defines a repeating pattern of time slots). When a node has traffic for one of its neighbors, it means it has an outgoing flow to that neighbor. Beacon packets are used to inform a node's neighbors of its own flows as well as those of it's neighbors. Using this method two-hop flow knowledge is established.

At the start of each data slot of the the superframe one or more flows are pseudorandomly selected to use that slot. Flows requiring more bandwidth are favored by the pseudorandom generator. Multiple flows can use the same slot only if their simultaneous transmissions will not cause a collision.

DYNAMMA-PRED uses the same framework as DYNAMMA with the addition of an "oracle" forecaster, which provides global queue state knowledge to the whole network. Unlike DYNAMMA, in DYNAMMA-PRED only flows that are known to have at least one packet to send are eligible to contend for a given slot. Although this method is not achievable in practice because it relies on all nodes

knowing instantly when a packet is ready to be sent at any of the nodes in their 2-hop neighborhood, the result provides a baseline of what is achievable with traffic forecasting.

Observations

To compare the performance of the two protocols we examine queuing delay, delivery ratio (total packets received divided by total packets sent), and percentage of time spent with radios in sleep mode using the Qualnet network simulator.

The scenario depicted in Figure 2.1 represents an infrastructure-less home network. An ad-hoc deployment will be essential to alleviating the burden of set-up and installation of smart devices on the consumer. The simulation scenario consists of 16 nodes arranged in a grid formation. They are spaced apart such that the diagonal spacing between two nodes is within radio range (25 meters) and creates a neighborhood pattern. The radio transmission rate is 53.3Mbps using a WiMedia physical layer model as defined in the ECMA-368 ultra wide-band physical and MAC layer standard [24]. Each node is a traffic source and randomly generates packets for random neighbors.

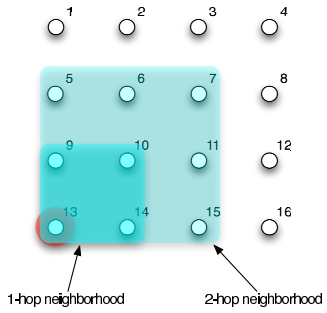


Figure 2.1: Simulation scenario: each node randomly generates traffic for its 1-hop neighbors.

The effect of the "oracle" forecaster on queueing delay is of particular interest. In Figure 2.2 we observe that the delay performance of DYNAMMA-PRED is better than that of DYNAMMA for inter-arrival times of $5ms$ and more. At high loads, when bandwidth demand exceeds availability

and MAC layer queues grow, DYNAMMA-PRED has comparable delay performance to DYNAMMA. DYNAMMA-PRED reaches a new delay floor of around $0.5ms$. To put this into perspective, the width of a slot is $638.125\mu s$ so the average delay is less than one slot length. DYNAMMA-PRED will try to schedule a flow for transmission during the slot after the packet arrives. The worst case delay in a lightly loaded system would be one slot length and the best case delay would approach zero, so on average the delay should be close to half a slot duration. The low delay of 802.11 at high load is misleading – the significant number of dropped packets are not accounted for in the MAC-layer delay measurement.

DYNAMMA is a multi-channel MAC protocol and we include results for 1–, 2– and 3– channel operation for both the original DYNAMMA and augmented DYNAMMA-PRED.

Figures 2.3 and 2.4 confirm that DYNAMMA-PRED retains the high delivery ratio and energy efficiency of DYNAMMA. It is not traffic forecasting but rather just the schedule-based nature of the medium access that provides good performance in these two aspects.

It is worth noting that the delivery ratio of schedule-based protocols drops only when the MAC-level queues fill up. Contention based protocols, on the other hand, begin to experience collisions even before the queues fill and so their delivery ratio steadily degrades with increased traffic load.

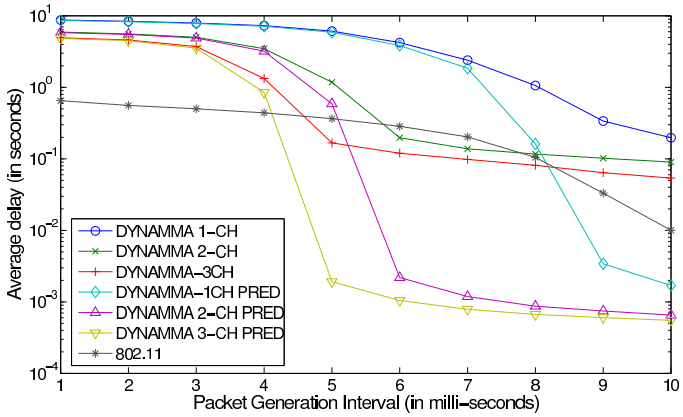


Figure 2.2: Average queueing delay of IEEE 802.11, DYNAMMA, and DYNAMMA-PRED.

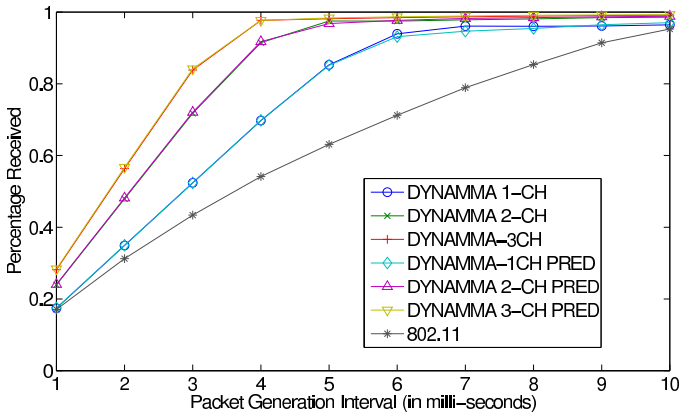


Figure 2.3: Average delivery ratio of IEEE 802.11, DYNAMMA, and DYNAMMA-PRED.

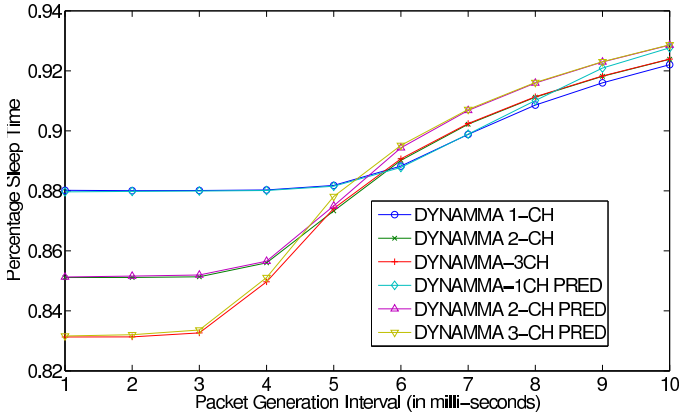


Figure 2.4: Percentage of time spent sleeping in DYNAMMA, and DYNAMMA-PRED.

2.4 Forecasting Traffic

The positive effect of the oracle forecaster can be seen in the results of DYNAMMA-PRED. Though "oracle" forecasting is not realizable, past traffic trends and patterns can be harnessed to anticipate future traffic behaviors.

There are several properties of a flow's traffic that may need to be forecast such as amount of throughput it will need and for what duration, the arrival time of its next packet, the size of the next packet or the average packet size. For the purpose of slot scheduling, we formulate the traffic forecasting problem in terms of the expected arrival time of the flow's next packet. In other words, the forecasting approach described in this section will focus on predicting a flow's expected packet inter-arrival time in order to schedule future transmission slots for that flow. The proposed traffic forecasting algorithm, which is based on the experts framework [17], selects the best inter-slot duration that a node should use to service a flow with minimal delay and minimal slot wastage.

Expert Framework Forecaster

The experts framework [17] is an on-line machine learning algorithm in which the output at each prediction trial depends on the individual outputs of a set of experts. All experts are given identical input data which they use to predict an output. The goal in the experts framework is to have the output of the overall algorithm track that of the best performing expert. An expert can be a complex algorithm that processes the input data to predict the output or it can be a simple function or even a constant that always gives the same prediction. Each expert has a weight that is multiplicatively updated at each trial of the algorithm. An expert's weight is reduced proportionally to how poorly it predicts. The output of the overall algorithm at each trial is the weighted average of the experts' outputs. The variable-share expert algorithm is used here to help experts that begin to forecast accurately quickly regain their weight.

The forecasting algorithm we designed operates on the assumption that packet inter-arrival

times are persistent over time. This assumption is based on empirical observations of traffic generated by applications that fall in the category described above, i.e., high date rate, real-time. Figure 2.5 shows cumulative distribution function (CDF) plots of the packet inter-arrival time and packet size properties of a Skype audio trace. The trace contains one direction of the Skype conversation and was recorded using the Wireshark [25] traffic monitoring tool at the source. The CDF plots reveal that only a few discrete values of inter-arrival time and packet size dominate the distributions. As home networks evolve, we expect to see many applications that, like Skype, are rich with application-dependent patterns that can be exploited by a traffic forecaster such as the one we outline here.

With MAC layer traffic forecasting, the time-scale of interest is rather small. It is determined by the size of a superframe in the protocol. Given that at least two superframe durations are required to propagate traffic information to a 2-hop neighborhood, the time scale of interest is around two superframe durations, e.g. in DYNAMMA it would be $320ms$.

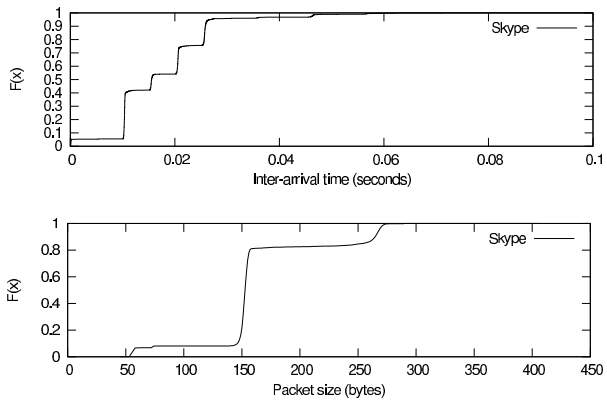


Figure 2.5: Cumulative distribution function plots showing packet inter-arrival time and packet size. There are around 5 discrete inter-arrival times and 3 discrete packet sizes.

The forecasting algorithm is designed around a schedule-based MAC's time structure. More specifically, the notions of superframe and time slot have been built into its operation. In a real implementation, the forecaster will run in real time and can be implemented in hardware for speed and

efficiency. Each forecast trial happens at the beginning of a superframe. At each trial, the algorithm evaluates the forecast of each expert and comes up with a new value for the best inter-slot duration. The experts' forecasts are evaluated using a loss function that takes into account both how many slots that forecast would waste and what the average per-packet delay would be. The forecast of each expert is a constant value in this algorithm and ranges from 1 to the number of slots per superframe. For example, a forecast of 1 corresponds to requesting every slot for the flow.

Forecasting Results

Our forecaster was run on the trace depicted in Figure 2.5. Figure 2.6 shows that the forecaster accurately predicts the slot period that will lead to best slot usage and lowest average delay. In this case, the best slot period corresponds to the inter-arrival time indicated by the the 0.01s dark line in the top plot of Figure 2.5.

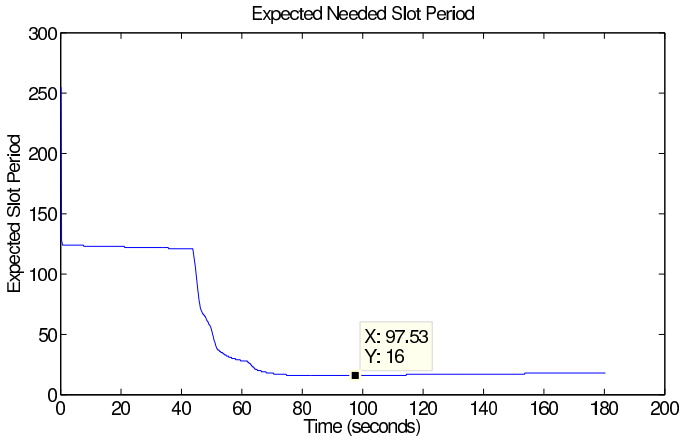


Figure 2.6: Forecaster output: prediction of slot period required to service a flow that produces lowest delay and wasted slots. Given that the slot duration was set at $638.125\mu s$, a slot period of 16 corresponds to $10.210ms$ and this is exactly the inter-arrival time indicated by the lowest dark cluster in the top plot of Figure 2.5.

The average delay incurred by a packet is depicted in Figure 2.7. When the forecaster con-

verges on the correct slot period, the average delay is around half the packet inter-arrival time. The forecaster constantly tries to reduce this delay while at the same time keeping the number of slots wasted low. Some amount of slot wastage must be traded in order to reduce the delay, and this can be tuned in the loss function that governs the operation of the experts algorithm. Figure 2.8 shows the slot usage of the algorithm to service the Skype audio flow. To reduce wasted slots, a backup transmitter can be elected along with the primary transmitter for each slot. The backup transmitter can use the slot if it does not detect the primary transmission within some timeout from the start of the slot.

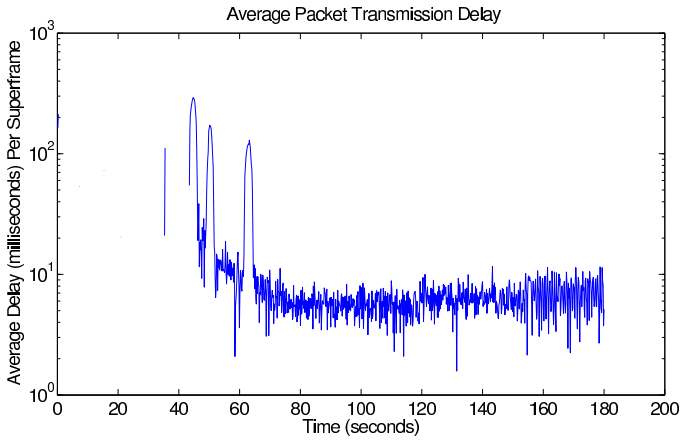


Figure 2.7: Average delay incurred by the experts forecasting algorithm.

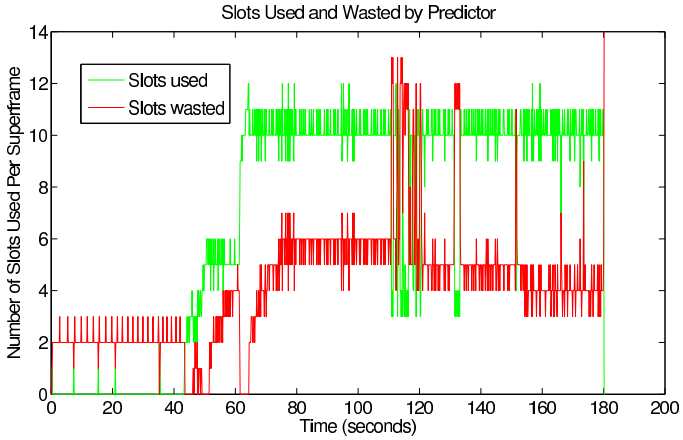


Figure 2.8: Slots used and wasted by the forecaster over time.

2.5 Challenges and Future Work

The forecaster presented in Section 2.4 works very effectively at forecasting the best slot period for a flow; it is therefore a promising approach to forecasting flow traffic at the MAC layer in order to improve schedule-based performance.

In this chapter we have shown forecasting working in a simplified scenario: a single node running a single application generating a single flow. In realistic scenarios each node may be host to flows from multiple network applications and in a multi-hop-capable network each node must forward traffic on behalf of other nodes. An example of the targeted applications are real-time multimedia applications whose traffic exhibits strong pattern and persistence. Furthermore, in the applications we are interested in, single-hop activity is more common making multi-hop aggregation less of a concern.

In traditional schedule-based medium access protocols, traffic information needs to propagate around these 2-hop neighborhoods in order for nodes to be able to set up schedules for traffic to be sent. Traffic forecast information needs to propagate around 2-hop neighborhoods in a similar fashion. The length of time required for this propagation to complete is implementation dependent, but in typical superframe-structured protocols it will take 2 superframes. The signaling done during the first superframe propagates the forecast to the 1-hop neighbors, and the signaling done during the second superframe propagates it to all the 2-hop neighbors. This non-negligible delay must be considered because if the traffic pattern does not persist for longer than this period, the forecast will be out of date by the time it is propagated.

Chapter 3

Characterizing Network Traffic Using Entropy

3.1 Introduction

There is no question that emerging and future Internet applications have become much more diverse and complex than the original Internet's "killer apps", namely e-mail, file transfer, remote login, and even early Web-based services. Application diversification will not only continue but will likely become even more accentuated as the Internet becomes the preferred medium for access to information, communication, and entertainment replacing or complementing the phone, TV, radio, movies, newspapers, books, etc. This trend is already visible today with services like Skype, YouTube, Hulu, and Netflix, to name a few. Teleconferencing and distance learning applications are also becoming more popular as well as media streaming, games, interactive TV, peer-to-peer and social networking. Although the user base of these applications is still growing, they already consume more than half of the total bandwidth [26]. And, as they become more popular, they will consume an even more disproportionate amount

of the Internet's overall resources.

We argue that, in order to engineer future internets such that they can adequately cater to their increasingly diverse and complex set of applications while using resources efficiently, it is critical to be able to characterize the load that emerging and future applications place on the underlying network. To this end, in this chapter, we explore ways to understand the correlation between the nature of an application and the complexity of traffic it generates. We investigated different metrics to characterize the complexity and behavior of application traffic in a systematic way. As a starting point, we explored self-similarity, which has become a well-known metric in the networking community and measures whether traffic preserves its burstiness at different time scales. We then looked into entropy, which, from Information Theory is defined as *a measure of information, choice and uncertainty* [1]. We found that, while self-similarity is not a strong indicator of traffic behavior, the entropy of packet inter-arrival times can generate application "entropy fingerprints" that can be used not only to clearly distinguish one application from another, but also provide a summary of the application's complexity over multiple time scales which can be used to quantitatively compare the complexity of one application's traffic to that of another.

Around the same time we were conducting our study, Riihijarvi et al. [27] also proposed the use of entropy as a complexity metric for network traffic. There are two distinctions between the works. First, Riihijarvi et al. [27] focus on aggregated network traffic where packets from multiple source-destination pairs and multiple applications are present. Our work targets per-(application) flow traffic, in which packets of a single application between a single source-destination pair are considered in isolation. Per-application flow characterization targets network control functions such as traffic scheduling and admission control at the edges of the network, which necessitates differentiating network traffic on a per-application basis. Riihijarvi et al., on the other hand, explore traffic model validation and anomaly detection applications to which aggregated network traffic is better suited.

While both efforts agree on the fact that self-similarity is not a strong indicator of traffic complexity (for both isolated and aggregated traffic), the second distinction between our approach and theirs

is the way the entropy analysis is conducted. We take an approach similar to that used in the neuroscience community to study neuron spike trains [28]. We map the packet arrival times of each trace to a binary series and estimate the entropy of this series. Riihijarvi et al., on the other hand, use the SampEn estimator [29] directly with the unprocessed time-series of packet inter-arrival times. We found that our approach of using the binary series in conjunction with our Plug-in Packet Timing Entropy (PPTEn) estimator captures more of the underlying application characteristics than the SampEn multiscale approach. In addition, the PPTEn estimator is capable of doing traffic forecasting with little to no modification, which can be beneficial for traffic scheduling.

The remainder of the chapter is organized as follows: in Section 3.2 we present the datasets that we use in this work. We present our finding on the applicability of self-similarity to traffic characterization in Section 3.3. Section 3.4 explains how entropy can be used as a complexity metric for network traffic and presents our PPTEn estimator and a brief overview of the SampEn estimator [29][27]. The results of our entropy analysis of the datasets are presented in Section 3.5. In Section 3.6 we show that PPTEn can be used as a traffic predictor as well and compare its performance to that of an ARMA predictor. Section 3.7 outlines some possible applications of this work and Section 3.8 concludes the chapter.

3.2 Datasets

In this section we describe application data we use in our study. We start by presenting a taxonomy of network applications that we feel (and a study by Sandvine Incorporated supports [30]) is representative of a large portion of today's network traffic in Table 3.1. In our taxonomy, applications fall into one of three categories according to their network traffic characteristics: streaming media, real-time or best-effort. We further subdivide the real-time category to differentiate voice over IP (VoIP), video conferencing, and remote access applications. For each application we indicate whether it uses

buffering, has traffic that tends to be bursty, has traffic that is affected by available bandwidth, has traffic patterns that depend on a application codec of some sort or has its traffic pattern greatly influenced by the presence or absence of user interaction.

Table 3.1: A taxonomy of common network applications. Applications whose traffic will be studied in this chapter are marked with "*".

Traffic class		Application	Transport protocol	Application protocol or codec	Buffered	Bursty	Bandwidth dependent	Codec dependent	User dependent
Streaming media		YouTube.com	TCP	H.264/MPEG-4 AVC [31]	•	•		•	
		Hulu.com *	TCP	H.264 [32]	•	•		•	
		Netflix.com *	TCP	VC1 Advanced Profile [33]	•	•	•	•	
		ABC.com *	TCP	TrueMotion VP7	•	•	•	•	
		Webcam stream *	UDP	RTP				•	
Real-time	VoIP	Skype *	UDP, TCP	ISAC, iLBC, G.729, iPCM-wb, EG.711A/U, PCM A/U, SVOPC [34]				•	
		iChat *	UDP	AAC-LD [35]				•	
		GoogleTalk *	UDP	PCMA, PCMU, G.723, iLBC, ISAC, IPCMWB, EG711U, EG711A [36]				•	
	Video conference	Skype *	UDP, TCP	TrueMotion VP7 [34]				•	
		iChat *	UDP	H.264/AVC [37]				•	
		GoogleTalk *	UDP	H.264 SVC, H.264, H.263-1998 [36]				•	
	Remote access	ssh	TCP	Secure Sockets Layer (SSL)					•
		VNC	TCP	Remote Framebuffer (RFB)				•	•
Best-effort		BitTorrent	TCP	BitTorrent protocol		•	•		
		File transfer	TCP, UDP	FTP/SFTP		•	•		
		Web browsing	TCP	HTTP		•	•		•

From the applications in the taxonomy, we chose from the ones whose traffic is not affected by user interaction to make up the traffic dataset that we use in the remainder of the chapter. We collected *tcpdump* [38] network traces and isolated the network traffic of the chosen applications. The application traces that form our dataset are listed in Table 3.2.

Table 3.2: Network traces that form our dataset

Real-time	VoIP	Skype iChat GoogleTalk
	Video conferencing	Skype iChat GoogleTalk
Media streaming		Hulu.com - “24” Hulu.com - “Chuck” Hulu.com - “American Dad” Netflix.com - “One Last Thing” Abc.com - “Castle” Webcam stream

We will analyze these application traces using entropy estimation algorithms and some of them using self-similarity in the remainder of the chapter. Before we proceed to the analysis, we describe the nature of each of the flows from the perspective of the cumulative distribution function (CDF) of their inter-arrival times and 5-second snapshots of their inter-arrival times. We make hypotheses in these two sections about the complexity of each application that we will come back to when we analyze the entropy estimates.

3.2.1 Real-time flows

The real-time flow group of traces consists of both voice over IP and video conferencing network traffic from Skype, GoogleTalk, and iChat. The traces were collected on the sender side so as to prevent deterioration of patterns due to network queueing. Durations of the flows are around 10 minutes and data rates range from $38kbps$ to $630kbps$. In the context of the real-time flows, we use the terms audio and VoIP interchangeably in this chapter.

From previous research on Skype traffic identification [39] [40] we know that we can expect to find patterns (and thus a high degree of predictability) in Skype audio flows. From the cumulative distribution function (CDF) of the packet inter-arrival times in Figure 3.1a, we can see that there are 4 distinct inter-arrival times in Skype VoIP flows. This supports the claim that there are patterns in Skype

audio traffic. The CDF for the Skype video conferencing flow in the same figure has similar distinct inter-arrival times to the VoIP one, with the addition of a new one at close to 0 seconds. The "staircase" in the Skype video CDF has smoother corners and non-horizontal steps, which means that there are inter-arrival times distributed throughou the [0,40]ms range. We expect the complexity of the Skype video conferencing flow will be higher than its VoIP counterpart, although the 5-second Skype flow snapshot in Figure 3.2b shows that some pattern is still evident.

The CDF of iChat audio (Fig. 3.1a) indicates that there is one fundamental packet inter-arrival time in the flow with the occasional extra packet. The 5-second flow snapshot in Figure 3.2a solidifies that observation. iChat audio has a more distinct pattern than Skype audio and can be expected to have lower complexity. iChat video has a similar packet inter-arrival pattern to iChat audio but has an additional distinct inter-arrival time of around 1ms as can be seen in both the CDF (Fig. 3.1a) and the flow snapshot (Fig. 3.2b).

GoogleTalk audio has the widest range of packet inter-arrival times according to its CDF in Figure 3.1a, but from the flow snapshot (Fig. 3.2a), it looks like the inter-arrival times around 60ms and 100ms dominate and as a result the flows behavior is less complex than that of Skype audio. GoogleTalk video seems to have nothing in common with its audio counterpart. Both the CDF and the flow snapshot show completely different behavior with no overlap in inter-arrival time concentrations. Due to the curved nature of its CDF "staircase", this flow is expected to be the most complex of the six.

3.2.2 Media streaming flows

We collected traces of three show episodes from Hulu.com, one episode from Abc.com and one movie from Netflix.com. Additionally, we collected a trace from a webcam streaming video using the Real-Time Protocol (RTP). The webcam streaming application is set apart from the others by the fact that it does not leverage client-side buffering and therefore doesn't have the burst-pause-burst network traffic pattern that is visible in the other traces.

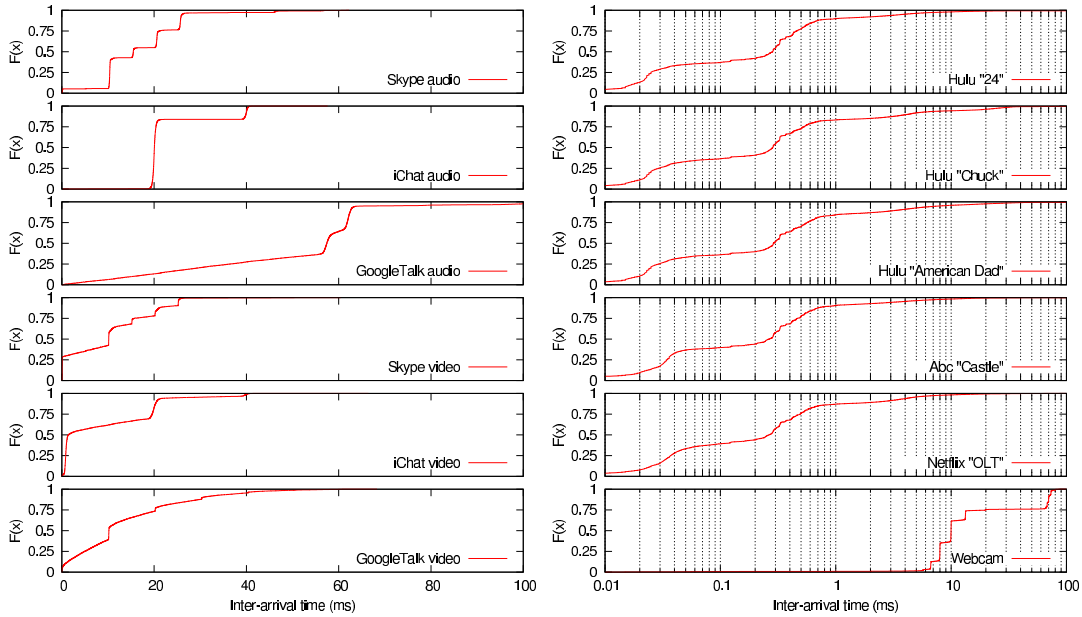
From the CDFs in Figure 3.1b it is difficult to distinguish between the Hulu, ABC, and Netflix flows, but the webcam flow is visibly different. In the flow snapshots in Figure 3.2c, the Netflix flow is easily differentiable, but the Hulu and ABC flows look very similar. This similarity is unexpected because the video codec used by ABC.com and Hulu.com is not the same (Table 3.1). Although the flow snapshots show the presence of a 2s inter-arrival time, its occurrence compared to the other inter-arrival times is so low that the CDFs don't show it (the 0-100ms CDF window appears to represent close to 100% of inter-arrivals).

Based on the CDFs and flow snapshots, our expectation is that the complexity of the webcam flow will be the lowest of the media streaming flows. The Netflix flow will have the highest complexity because it has less visible pattern than the Hulu and ABC flows. The remaining 4 flows will be very similar in complexity.

How the complexity of the media streaming flows will compare to that of the real-time flows is a more difficult estimation to make. Our hypothesis is that the larger amount of data in the media streaming flows will make their complexity higher than that of the real-time flows

3.3 Self-similarity

We considered self-similarity as our first candidate for a forecastability metric. Self-similar processes are processes that exhibit long range time dependencies. Consider a covariance stationary stochastic process $X = \{X_n : n = 0, 1, 2..., \}$, with variance σ^2 and autocorrelation function of the form $r(k) \sim k^{-\beta} L(n)$, for large k, where $0 < \beta$, and L is slowly varying for large n [41]. A process is self-similar if $\beta < 1$, implying that events in the distant past influence present outcomes. The Hurst coefficient $H = 1 - \beta/2$ is a measure of self-similarity, as $H \leq 1/2$ correspond to processes with short memory. Short memory has a precise characterization. Consider the block-aggregate process resulting from averaging blocks of m variables together, and its covariance $r^m(k)$. Self-similar processes have the



(a) VoIP and video conferencing

(b) Media streaming

Figure 3.1: CDFs of packet inter-arrival times for real-time flows and media streaming flows

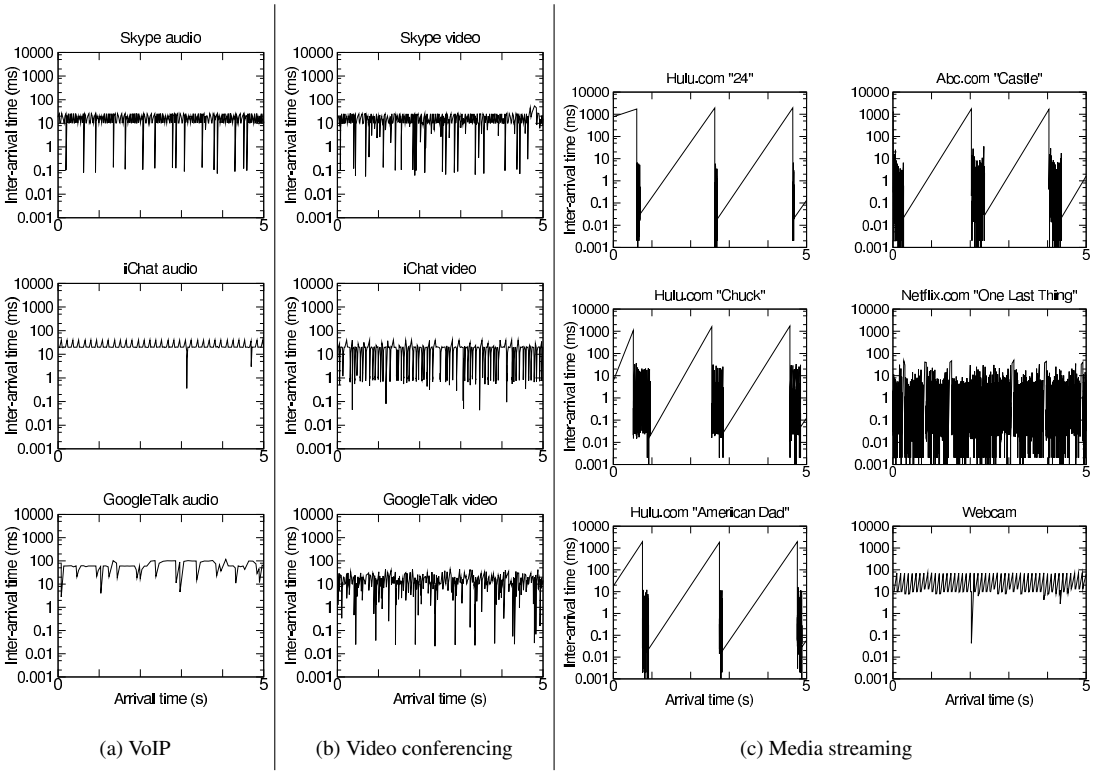


Figure 3.2: Packet arrival patterns for VoIP flows, video conferencing flows, and media streaming flows

property that at least $r^m(k) \to r(k)$ as $m \to \infty$. Usual processes instead have $r^m(k) \to 0$, so averaging removes dependency, which implies forecasting does not require looking at a very distant past.

3.3.1 Self-similarity related work

A traffic arrival process in a network can have different characteristics depending on the time scale. *Time scale* corresponds to the speed at which variations happen in the traffic arrival stochastic process. It is measured by aggregating packet arrivals at different discretized time interval lengths τ. Leland et al. [41] first showed the presence of statistical self-similarity in Ethernet traffic for packet-switched networks. Bursts of packet arrivals are present regardless of time scale, showing the poor applicability of poisson arrival models. Self-similarity has also been observed in World-Wide-Web traffic [42] and in Variable-Bit-Rate video traffic [43] among others. [44] refuted traditional network traffic models and [45] reconciled traditional traffic models with the self-similar behavior of real network traffic. Park et al. [46] show that self-similarity can be caused by the statistics of the file size distributions, more so in TCP flows. A self-similar dynamic process is less forecastable than a corresponding correlated random process. Statistical prediction methods require changes happening in lower frequency at some time scale, which is not satisfied by self-similar traffic. The Hurst parameter quantifies the degree of self-similarity. We aim to use the Hurst parameter as an indicator to identify whether certain types of flows are *not* likely self-similar and therefore better forecastable.

SELFIS [47] is a self-similarity analysis tool that implements various estimators of the Hurst parameter. The estimators calculate the Hurst parameter H_m for various time-scales m and declare self-similarity when $1/2 < H_m < 1$. [47] observe that self-similar sequences are likely to be declared so when most of the estimators agree. When H_m is very close to (below) $1/2$ or (above) 1, for the majority of the estimators, and various time-scales, the sequence is likely not self-similar.

To evaluate forecastability of the traffic process, we consider evaluating self-similarity from average flow rate, as average flow rate implies large variation of inter-arrival times at various time scales.

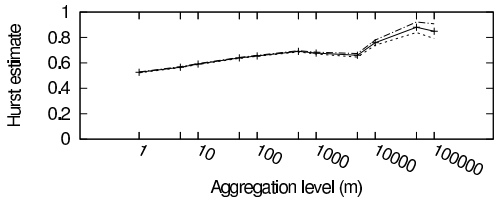
To evaluate the self-similarity itself we run SELFIS for various aggregation levels m and plot the corresponding estimates for various different estimators. If for most m and most estimators, $1/2 < H_m < 1$, then we strongly believe the sequence to be self-similar.

3.3.2 Self-similarity results

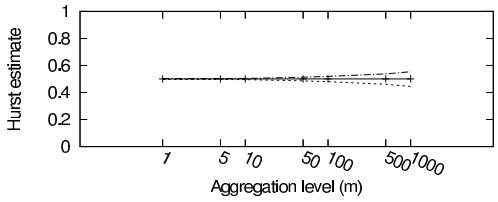
Internet backbone traffic is strongly believed to be self-similar [45]. We consider one minute of traffic captured at an Equinox datacenter in San Jose, CA on January 15 at an OC192 backbone link of a Tier1 ISP between San Jose and Los Angles [48], referred to here as CAIDA flow. There are approximately 13 million packets during the 1 minute interval, necessitating a small base aggregation interval of 1μs . Figure 3.3(a) plots Whittle Hurst estimates for various aggregation levels, and shows strong evidence that data exhibits self-similarity. Figure 3.4(a) shows that the Abry Veitch Hurst estimator confirms this observation. The dotted and dashed lines correspond to bounds on the estimate. Table 3.3 summarizes the observations by all the estimators in the SELFIS tool for all the traces studied. In the table, we use a (+) to indicate self-similarity if at all aggregation levels the Hurst estimate is in the range (1/2,1), a (-) if at all aggregation levels the Hurst estimate is outside this range, and a 0 if a it is a mixture of both.

Table 3.3: Summary of Hurst estimator results. For each flow we run 7 Hurst parameter estimators over a variety of aggregation levels. The results are plotted and in this table each plot is summarized by a (-), (0), or (+), indicating **no** self-similarity, **maybe** self-similarity, or **yes** self-similarity, respectively. A (*) indicates that $< 50\%$ of the points fell outside of the 0 to 1 range, and a (**) indicates that $> 50\%$ of the points fell outside of the 0 to 1 range (this is an indication that the estimator is not robust for this data and calls into question its output)

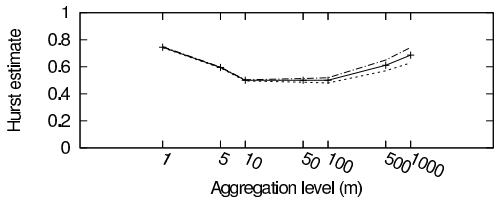
Flow	Aggregate Variance	R/S	Periodogram	Absolute Moments	Variance of Residuals	Abry Veitch	Whittle
CAIDA	+	0	+	0	+	+	+
GoogleTalk Audio	+	-	0	+	+*	0	0
iChat Audio	-	-	-	-	-*	-	-
Skype Audio	+	+	0*	+	0**	0	+
iChat Video	-	-	0	0	+*	0	+
Skype Video	+*	+	0*	+	+**	0	0



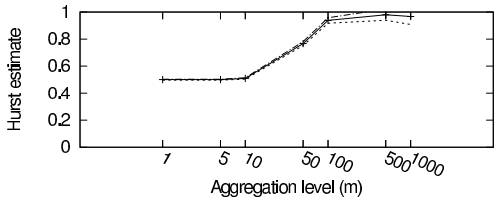
(a)



(b)

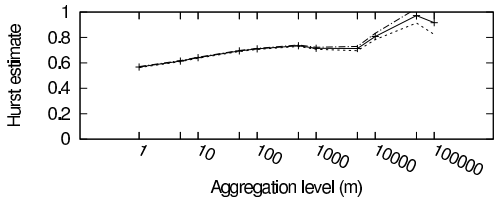


(c)

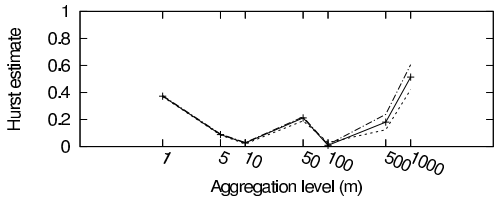


(d)

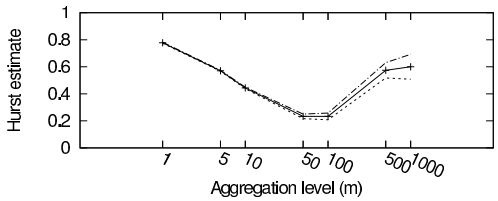
Figure 3.3: Whittle estimator Hurst estimates of (a) CAIDA flow (known to be self-similar), (b) iChat audio flow, (c) iChat video flow and (d) Skype audio flow.



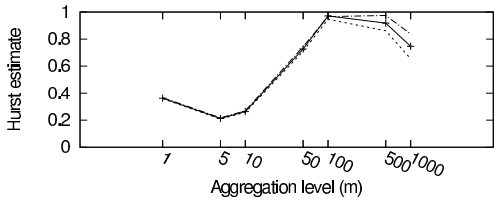
(a)



(b)



(c)



(d)

Figure 3.4: Abry Veitch estimator Hurst estimates of (a) CAIDA flow (known to be self-similar), (b) iChat audio flow, (c) iChat video flow and (d) Skype audio flow.

We then consider the traces of Skype, GoogleTalk, and iChat, both audio and video. Data rates for these flows range from $4.7kB/s$ to $78kB/s$, so a bigger base aggregation interval of 1ms is chosen. Figures 3.3 and 3.4 show plots of Hurst estimates of some chosen flows from this set for the Whittle and Abry Veitch estimators, respectively. The Whittle estimates in the range of $(\frac{1}{2}, 1)$ for the CAIDA flow indicate a presence of self-similarity in that trace. For the iChat audio, iChat video and Skype audio flows, the conclusion is weaker with many aggregation levels exhibiting Whittle's lowest estimate of $\frac{1}{2}$. The Abry Veitch estimates are in line with those of the Whittle estimator, though the Abry Veitch estimator has a wider range. Table 3.3 summarizes the results for various classes of flows. Notice that Skype audio and video are not believed self-similar, but for certain aggregation levels, Hurst coefficients in the range $(\frac{1}{2}, 1)$ were observed. This can be caused by both jitter and outliers. Contrary to what we hoped, as Riihijarvi et al. also found [27], self-similarity does not offer a conclusive indication of complexity and thus will not make a good complexity indicator.

3.4 Packet timing entropy estimation

In this section we describe why entropy can be used as a complexity metric, we present our Plug-in Packet Timing Entropy (PPTEn) estimator and the theory behind it and we briefly describe the SampEn estimator used by Riihijarvi et al. [27].

3.4.1 Entropy rate

Given a stochastic process $X = (X_n : n = 0, 1, ...)$ taking values in a discrete domain $\mathcal{D}$, its entropy rate $H(X)$ is defined as:

$$H(X) = \lim_{n\to\infty} \frac{1}{n} H(X_1, ..., X_n), \tag{3.1}$$

where $H(X_1, ..., X_n)$ is the entropy of the set of random variables $\mathbf{X}_n = \{X_1, ..., X_n\}$ with joint probability $P(X_1, ..., X_n)$, and is given by

$$H(X_1, ..., X_n) = -\sum_{\mathbf{X}_n \in \mathcal{D}^n} P(X_1, ..., X_n) \log P(X_1, ..., X_n).$$

Standard information theoretic results [49] show that $0 \leq H(X) \leq 1$, and for a *stationary* stochastic process, the rate is given by the residual entropy:

$$H(X) = \lim_{n \to \infty} H(X_n | X_{n-1}, ..., X_1), \tag{3.2}$$

where conditional entropy is defined with respect to conditional probabilities. Notice that this result also suggests that a stationary Markov process with memory length l has rate given by $H(X) = H(X_n | X_{n-1}, ..., X_{n-l+1})$, and that in some sense non self-similar processes have this property as the correlation to X_r for $r \ll n$ becomes small quickly. The importance of this property is that entropy rates can be reliably estimated for such processes, using finite memory estimators [50].

The entropy rate of a sequence is a measure of *how predictable* a sequence is based on past observations. An intuitive interpretation is how much new information the outcome of X_n brings, when we have observed the past. A constant sequence has entropy rate 0, and a sequence of independent fair coin tosses has entropy rate 1, as every new coin toss brings one whole bit of information. Sequences with lots of repeated patterns have low entropy rates, as previous pattern help to predict more recent outcomes. In fact a periodic sequence, without randomness, has entropy rate 0 once it has been determined to be periodic.

3.4.2 Multiscale plug-in packet timing entropy estimator

Markov plug-in estimator. A standard estimator for the entropy rate of an independent and identically distributed (i.i.d.) sequence X is given by Maximum Likelihood Estimator (MLE) of the empirical entropy loss:

$$\hat{H}(X) = \max_P -\frac{1}{n} \sum_{r=1}^{n} \log P(X_r), \tag{3.3}$$

where P is the discrete distribution over the m atoms of the discrete domain that defines each element of X. The probability distribution $P = \hat{P}$ that solves the above optimization problem is the empirical estimator [51]

$$\hat{P}(k) = \frac{1}{n}\sum_{r=1}^{n} \mathbf{1}(X_r = k), \tag{3.4}$$

that counts the number of occurrences of the k-th member of the discrete domain in the sequence. The *plug-in* entropy estimate is obtained by plugging in $\hat{P}$ in the definition of $\hat{H}$ in place of P. The plug in estimator has some important properties when the sequence is independent and identically distributed. It is biased so that

$$\mathbb{E}[\hat{H}] - H = -\frac{m-1}{2n} + O(1/n^2), \tag{3.5}$$

and it has asymptotic variance given by $\text{Var}[\log P(X_1)]/n^2$. A simple extension can be used for estimating the *entropy rate* in Eq. 3.1 when X is not independent, but is Markov, by using a conditional plug-in estimator $\hat{P}(X_n = k|X_{n-1} = r)$ following the standard empirical estimator. The residual entropy formulae (Eq. (3.2)) then shows that inserting this as a plug-in in the conditional entropy definition

$$\begin{aligned} H(X) &= \sum_r \hat{P}(r) H(X_n|X_{n-1} = r) \\ H(X_n|X_{n-1} = r) &= \sum_k -\hat{P}(k|r) \log \hat{P}(k|r), \end{aligned} \tag{3.6}$$

where $\hat{P}(k, r) = \hat{P}(X_n = k|X_{n-1} = r)$. In general we can consider Markov processes X with memory length l, where X_n is conditionally independent of the past given $X_{n-1}, ..., X_{n-l}$. Extending the proof in [51] we obtain

Theorem 1. *The conditional entropy estimator is a biased, consistent and asymptotic normal estimator for a* stationary *Markov process X with memory length l, with bias given by*

$$\mathbb{E}[\hat{H}] - H = -\frac{m^l \times (m-1)}{2n} + O(1/n^2). \tag{3.7}$$

Improvements in performance are obtained by using a *coverage adjusted* entropy estimator

[52], that rescales $\hat{P}$ appropriately, although for large n relative to m, both estimators are very similar and moreover, the resulting bias is unknown. To abbreviate the name of the procedure, we call it PPTEn.

Packet timing entropy estimator. A sequence of packet arrival times $t_1, ..., t_n$ characterizes the packet arrival process. Typically we are interested in learning if there is some finite memory predictability for this process. The entropy rate captures such behavior. Unfortunately, packet arrival times usually belong to an unbounded integer domain, and therefore the estimator suggested in Eqn (3.6) may fail since it has a bias error proportional to m, the size of the domain (see Eqn. (3.7)). Furthermore, a standard discrete distribution implies that two elements x_1 and x_2 of the domain are not comparable, independent of a notion of distance between both. For example, small random jitters can increase entropy substantially. Finally, packet bursts can also lead to higher entropy, although we desire a process composed of periodic bursts have small entropy independent of the number of packets in the burst. Thus we separate packet timing from number of packets at a given time scale.

We consider a *rescaled* representation of the timing sequence to address these issues. Divide time into bins of size τ, the time scale unit. Create a timing pattern sequence s_k, such that $s_k = 1$ if there exists some t_r in the interval $[k\tau, (k+1)\tau)$ and $s_k = 0$ otherwise. As you make the intervals larger, the likelihood that small jitters move to adjacent bins decreases a lot. Moreover, if a few jitter samples occur, you still will get a good entropy estimate. It is only if jitters are frequent that it is an issue. In fact, the binning is equivalent to a smoothing filter. An alternative way to seeing this is that the random variable $\tilde{\tau} = \lfloor \tau/T \rfloor * T$, under the model $\tau = P + e$, where P is a fixed constant period, and $e < M$ is a bounded random jitter error, has zero variance as soon as $T > M$. In fact, the variance decreases quickly with T.

Now using the plug-in estimator Eqn. (3.6) compute $\hat{H}(\tau, l)$, the entropy of the timing pattern sequence for time scale τ and memory length l. Notice that $m = 2$ since the sequence is binary. Theorem 1 shows that the estimator is guaranteed to be near consistent for memory lengths $l - 1 \ll \log_2 n$. For other memory lengths it may depend on the effective size of the conditioning sequences. Notice that the theoretical guarantees are usually conservative, and in practice performance may be

better.

The multi-scale plug-in packet timing entropy estimator has low computation complexity in general, and in particular can be quickly computed for scales such that $\tau = 2^b\tilde{\tau}$, where $\tilde{\tau}$ is some reference smallest scale. This is an important and appealing property that allowed us to explore datasets in a more comprehensive way.

3.4.3 SampEn entropy estimator

The SampEn estimator works on the principle that the entropy rate of the sequence $\{t_1, ..., t_n\}$ for memory length l can be approximated by counting the number of vectors of size l selected as a contiguous subsequence of $T' = \{t_2 - t_1, ..., t_n - t_{n-1}\}$ within some distance r. The notion of time-scale is introduced by creating a sequence as a running averaging of T' of size τ. The estimator itself is not easy to compute, but it is shown to exhibit good behavior in various empirical sets.

PPTEn provides a complimentary view to SampEn by separating packet intensity and packet timing. Its simpler computation allows for determining quickly the time-scales of interest, and can be used as an input to a SampEn analysis. Furthermore, the bias tradeoff faced by PPTEn is favorable compared to SampEn. Finally, if coverage adjusted entropy is used, PPTEn is unbiased for finite samples but SampEn is not [52].

3.4.4 Entropy estimation related work

The entropy rate of a stochastic process is a measure of how predictable the process is, as it measures the amount of associated uncertainty or *information* [1]. In networking, entropy rate measurements have been used for attack detection and network behavior profiling [53], [54], [55], [56], [57]. We aim to use entropy as an indicator of the degree of predictability associated with a traffic process. The neuroscience community has investigated various estimators for the entropy rate associated with the arrival of neural spikes [28], i.e., the computation of the entropy of a sequence of 1s and 0s. If 1s are

associated to a packet arrival, and 0s to no packet arrival for a discrete time interval, a packet flow maps to a spike train. Entropy then measures the presence of *patterns* in the arrival process, as patterns reduce entropy. We use a variation of the plug-in estimator that we developed to demonstrate that certain flow processes have *memory* and thus reduced entropy.

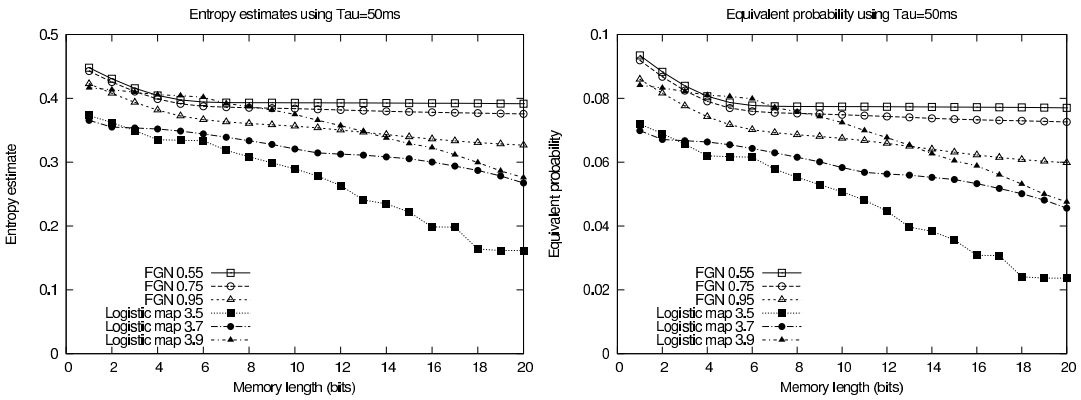
3.5 Entropy-Based Traffic Complexity Analysis

3.5.1 PPTEn estimator results

The raw data that we extract from the flow traces consists of packet arrival timestamps and packet sizes. Though entropy analysis can apply to both packet arrival times and packet sizes, we focus on arrival times.

Inspired by the neuron spike encoding used in [28], our approach is to encode packet arrivals in a simple binary sequence. In this approach, time is divided into bins of some size, τ and the binary value of each bin represents whether there was a packet arrival during that bin or not. The bin size, τ, is clearly an important parameter in this approach. With bin sizes too large, we risk losing information (multiple packet arrivals are treated the same as a single packet arrival). With bin sizes too small, no information is lost, but it turns out that the entropy estimates suffer because the abundance of empty bins drown out the effect of the few non-empty ones.

In their paper, Riihijarvi et al. [27] use several synthetic processes to verify that the output of their entropy estimator is in agreement with the expected complexity of the processes. We picked two of the same processes – the fractional gaussian noise process and the logistic map process – to verify that there is agreement between our entropy estimator and theirs. The plots in Figures 3.5a and 3.6a show that, in agreement with the SampEn estimator, the entropy ordering from our estimator of the 6 flows in order from lowest to highest entropy is:



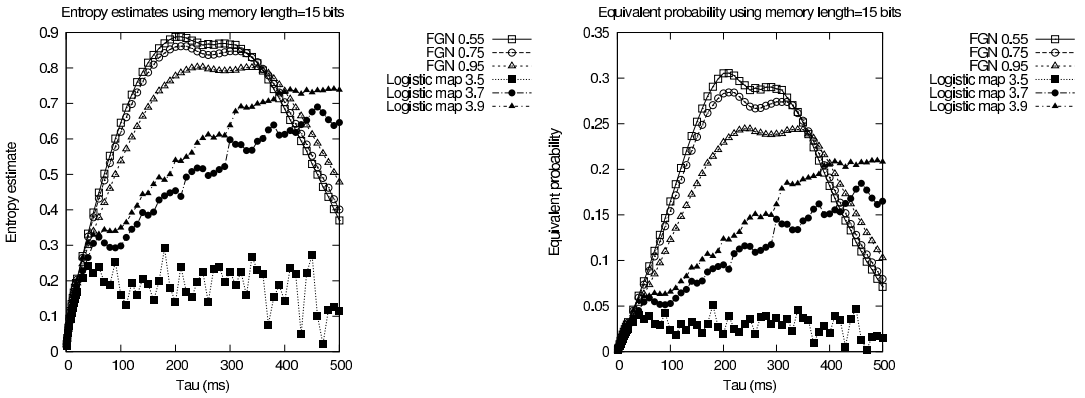
(a) (b)

Figure 3.5: Entropy estimates (a) and equivalent probability (b) of 2 types of synthetically generated flows. Using larger word lengths reduces the entropy estimate of the dataset because packet arrivals in a flow are not independent of each other.

Lowest entropy	Logistic map, $R = 3.5$
	Logistic map, $R = 3.7$
	Logistic map, $R = 3.9$
	Fractional gaussian, $H = 0.95$
	Fractional gaussian, $H = 0.75$
Highest entropy	Fractional gaussian, $H = 0.55$

Effect of word length

Equivalent probability is an alternative way of presenting the entropy information presented in Figure 3.5a. Given an entropy estimate of a binary sequence, the equivalent probability is $p < (1 - p)$, that would yield that same entropy. Let's also assume that the entropy estimate is being used as a means of predicting what the next symbol in the sequence will be — a "1" or a "0" — and we pick whichever has a higher probability according to the estimator. In this scenario, the equivalent probability, p, becomes



(a) (b)

Figure 3.6: Entropy estimates (a) and equivalent probability (b) of synthetic flows as a function of time interval, τ.

the probability that you will be wrong if you pick the symbol with higher probability. To restate that **in terms of word length: the more history you take into account when making your prediction, the smaller the chance your prediction will be wrong** (Figure 3.5b).

Effect of Time Interval

Next we fix the word length at 15 bits and investigate the effect of time interval, τ, on the estimator output. Figure 3.6a shows the entropy estimate as a function of time interval.

Looking at the equivalent probability plots (Figure 3.6b) as we did in the previous section, provides some intuition on the effect of τ on the entropy estimates. In general, **if τ is more than m times bigger or smaller than a flow's packet inter-arrival time, the effect will be a reduction in the probability of a wrong prediction**. Considering that we are predicting whether a packet will arrive or not during the next time interval, τ, a large τ will almost guarantee a packet arrival. Similarly, for a very small τ we can almost guarantee the absence of a packet arrival. This behavior is also readily observable in the application traces discussed below.

Ultimately, what τ will show a peak in the entropy estimate will depend on the flow itself. With low values of τ such as $1ms$, the entropy of the flow is likely to be low if the rate of packet arrivals is low (i.e. on the order of 1 packet every $20ms$) because in this case the bit string is mostly 0-bits with the occasional 1-bit, and the entropy estimator picks this up as a low entropy because the 1-bits are overpowered by the 0-bits. As τ increases we can reach the other extreme: a bit sequence with mostly 1-bits and the occasional 0-bit. This second case occurs when τ is approximately equal to the largest inter-arrival time in the flow.

For the remainder of the entropy estimates, we fixed the memory length of our estimator at $m = 15$ and varied τ between $1ms$ and $200ms$ for the real-time flows and between $0.001ms$ and $1000ms$ for the media streaming flows. The results are summarized in Figure 3.7 for real-time flows and Figure 3.8 for media streaming flows.

Real-time flow complexity

Depending on the packet arrival pattern of the flow being estimated, a different value of τ may be appropriate. It is necessary to observe the entropy estimates over a range of τ values to get a more complete picture. Figure 3.7 presents the entropy estimates of the VoIP and video conferencing traces that come from Skype, GoogleTalk and iChat.

From close inspection of the packet inter-arrival CDFs and the flow snapshots in Section 3.2, we set the expectation for the trends that the entropy results should match. Flows with lower complexity should have lower entropies, therefore we expect the entropy of VoIP flows to be lower than that of video conferencing flows and we expect that the iChat audio flow will have the lowest entropy and GoogleTalk video will have the highest.

First inspection of the entropies in Figure 3.7 may lead you to conclude that the first expectation — that VoIP flows have lower entropy than their video conferencing counterparts — has not been met. For τ in the range [8,18]ms, the entropy of GoogleTalk audio is larger than the entropy of GoogleTalk

video. Similar discrepancies exist for Skype. However, if you note the entropy value of each flow at its highest point, you'll notice that the ordering that we expect is exactly what the estimator gives us:

Lowest entropy	iChat audio
	GoogleTalk audio
	Skype audio
	iChat video
	Skype video
Highest entropy	GoogleTalk video

This observation means that we can't pick some value of τ that will be appropriate for all the flows and compare their entropies using that single value of τ. The location of the peak entropy is related to the distribution of the inter-arrival times of the flow. For example, Figure 3.1a shows that GoogleTalk audio has a larger inter-arrival time than the other real-time flows and correspondingly, we see a peak in its entropy for a larger value of τ.

Due to the nature of the way we create the binary sequences that are fed to the estimator, all fluctuations in inter-arrival time smaller than τ are filtered out. Increasing τ increases the time scale at which the entropy estimates apply. Multiple peaks in the entropy of a flow mean that the flow exhibits complex behavior at multiple time scales. For example, the GoogleTalk video flow has a peak at $\tau = 4ms$ and then another at $\tau = 20ms$. The peaks correspond to the two larger sections of the flow's CDF in Figure 3.1a.

Media streaming flow complexity

The PPTEn estimates for the media streaming flows are shown in Figure 3.8. We have used a wider range of τ values for these flows because their CDFs (Fig. 3.1b) and flow snapshots (Fig. 3.2c) indicate that they have inter-arrival time patterns at two very different time scales.

An examination of the peak entropies of the flows yields the following ordering:

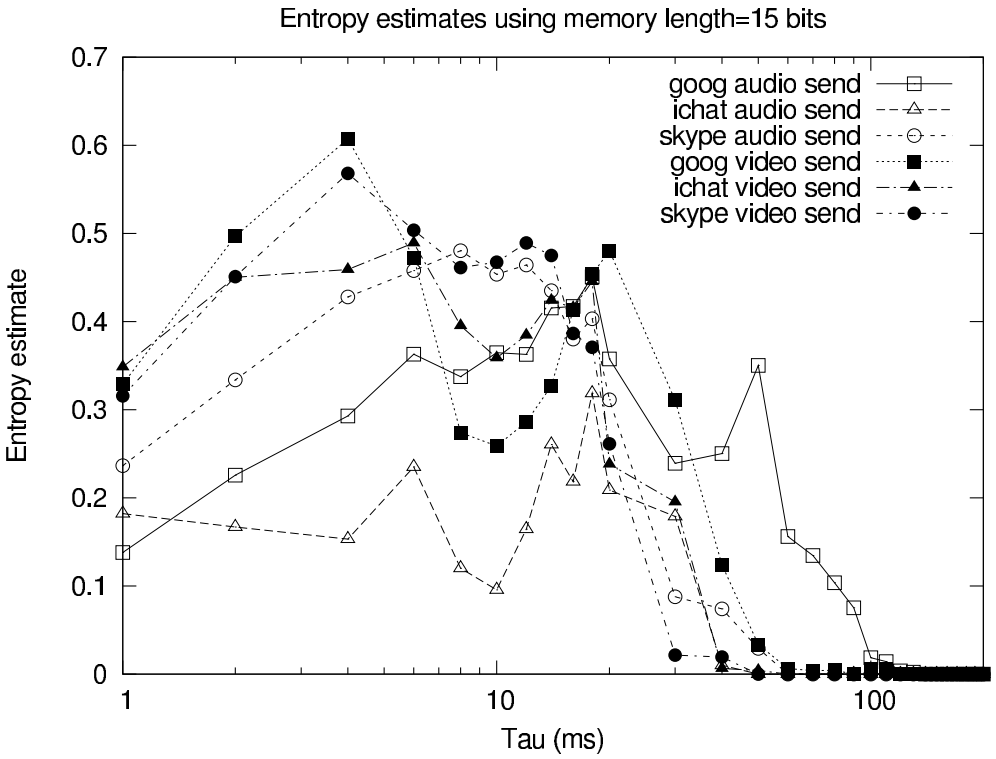


Figure 3.7: Entropy estimates of VoIP and video conferencing flows using $m = 15$.

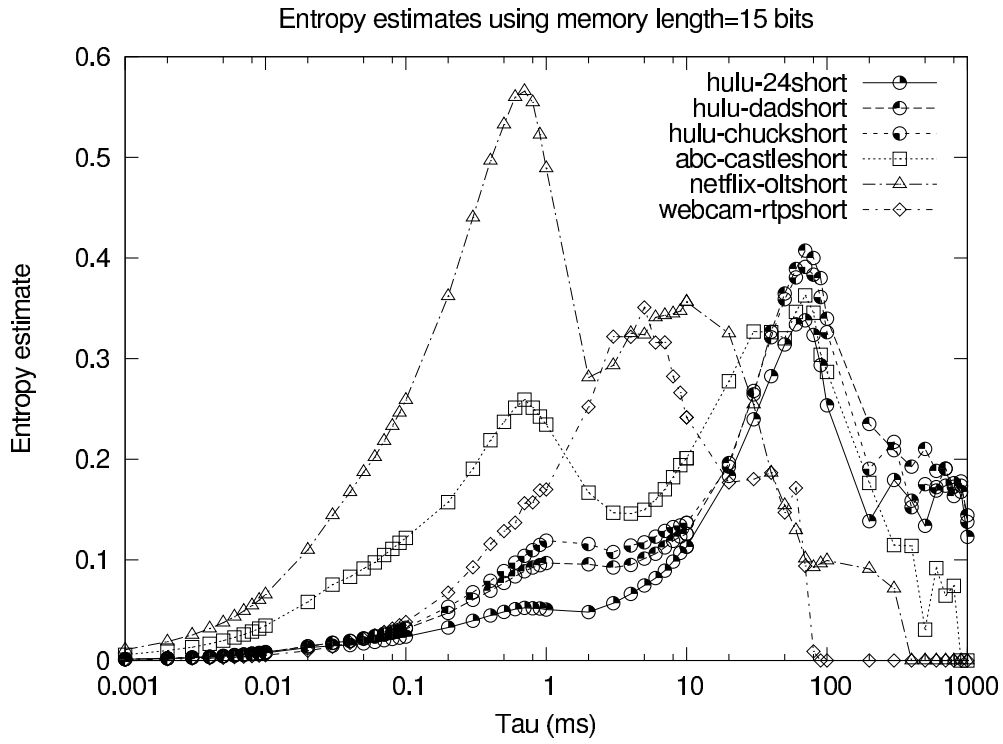


Figure 3.8: PPTEn estimates of media streaming flows using $m = 15$.

Lowest entropy	Hulu.com “24”
	Webcam
	ABC.com “Castle”
	Hulu.com “Chuck”
	Hulu.com “American Dad”
Highest entropy	Netflix.com “OLT”

The Netflix flow has the highest entropy as expected. Although the webcam flow was expected to have the lowest entropy, it is higher than one of the Hulu flows. Furthermore, a comparison between Figures 3.7 and 3.8 shows the peak entropies of the media streaming flows are lower than the peak entropies of the video conferencing flows and on par with the VoIP flows. It might be expected that the media richness of the media streaming flows should make them more complex than the VoIP flows, but

that does not account for the buffered nature of one versus the real-time nature of the other. By using the coverage adjusted entropy estimator approach mentioned in Section 3.4.2 we can improve the sensitivity of the estimator and may be able to improve the strength of this result.

In our discussion of the media streaming dataset in Section 3.2.2 we mention that by just looking at the CDFs in Figure 3.1b it is difficult to distinguish between the Hulu flows and the ABC and Netflix flows. Furthermore, though the flow snapshots in Figure 3.2c show that Netflix has significantly different behavior than the ABC and Hulu flows, distinguishing the Hulu and ABC flows is still impossible. An inspection of the PPTEn estimates, however, shows that the ABC flow's behavior is quite different from that of the Hulu flows.

Not only do the peak entropies allow us to rank the complexity of application network traffic, but the location of the peaks also gives us valuable information about the scale at which we can expect the traffic to be most unpredictable. The location and height of the peaks forms a fingerprint for each application. It is left as future work to see how sensitive to the media content, such as speaker's voice or conversation content in VoIP or which television show is being streamed by a media streaming application, this fingerprint will be.

3.5.2 SampEn estimator results

We ran the SampEn estimator on the time series of packet inter-arrival times extracted from each application flow. The multiscale SampEn results are shown in Figure 3.9 for the real-time flows and in Figure 3.10 for the media streaming flows. A scale of 1 on the x-axis corresponds to the original time series, a scale of 2 means that every pair of samples in the original series has been averaged together to make up one sample, and so forth. This scaling approach, shown by Riihijarvi et al [27] allows entropy estimates over larger timescales.

Unlike our PPTEn estimates, the multi scale SampEn estimates do not exhibit telltale peaks. It isn't clear what the effect of increasing the time scale is on the operation of the SampEn estimator, as it

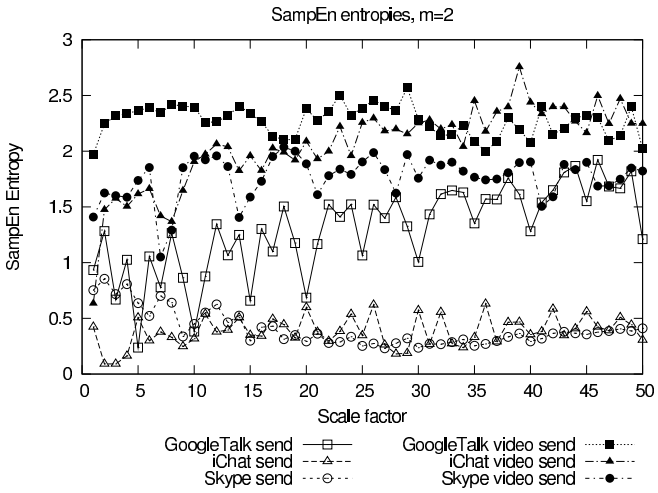


Figure 3.9: SampEn estimates of real-time flows with SampEn parameter $m = 2$

was for PPTEn.

The trends in Figure 3.9 indicate that Skype audio has the lowest complexity and iChat video and GoogleTalk video seem to have the highest. Strangely, the entropy of Skype video is lower than that of GoogleTalk audio and for larger time scales as low as that of Skype audio. The trends are not consistent with those shown by our PPTEn estimator or with the expectations that we outline in Section 3.2.1. The effect of increasing m does not affect the overall trends but adds to the variability already evident when we use $m = 2$.

For the media streaming flows in Figure 3.10, one trend that is immediately apparent is that the SampEn estimator sees the webcam RTP stream as the most complex of the media traces. This is exactly opposite to our expectation and the result from our PPTEn estimator. The SampEn estimator correctly shows the Hulu traces having similar entropies and distinguishes them from the ABC trace, as did the PPTEn estimator. For the media streaming, with the exception of the webcam flow, the SampEn estimator provides the same entropy based ordering as PPTEn. A big drawback of SampEn in this application is that the concept of scale is not intuitively related to the behavior of the application whereas with PPTEn, it can be seen how the τ value of the entropy peaks relates to the network traffic patterns.

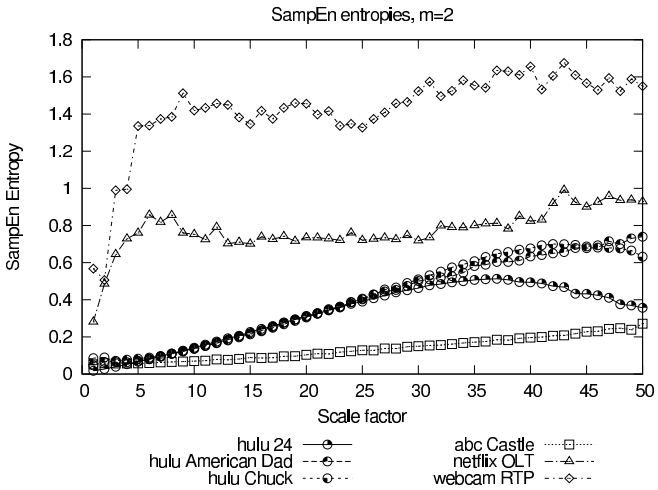
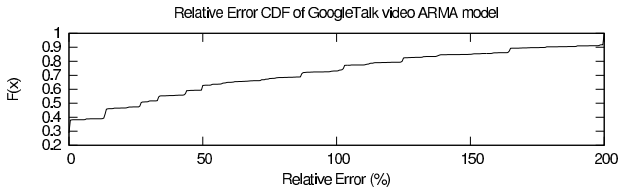


Figure 3.10: SampEn estimates of media streaming flows with SampEn parameter $m = 2$

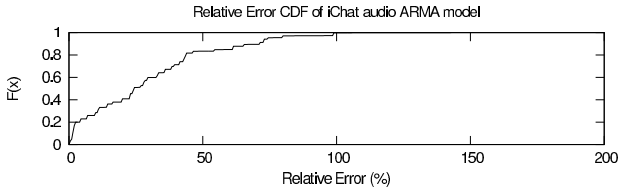
3.6 ARMA and Entropy based Predictors

In this section we will briefly explore a couple of simple approaches to forecasting packet arrivals. The first approach is similar to one taken by Sang et al. [58]. In their work, an ARMA model is used to forecast the bandwidth requirement during the next interval. In our work, we are more concerned with packet arrival time because it is more adequate for the types of applications we target, e.g., traffic scheduling. The ARMA model approach can be used to forecast packet arrivals, but there is one caveat: we cannot apply a low-pass filter to the packet arrival data as in [58]. Instead, we use the time scale τ to obtain a similar effect: reduce high frequency variations. By quantizing the inter-arrival times to multiples of τ, and discarding all times t_i such that $t_i < \tau$, we are able to observe a multiple-time scale effect for this process as well. We estimate the ARMA model with a 10th order moving average model, no feedback filter tap, and an autoregressive model of order 10 for the error remaining after using the forward predictor.

Poor performance of the approach does not necessarily indicate that the traffic flow is not forecastable, only that the model does not fit the data. In particular, ARMA models are good at capturing low frequency changes in the data and can represent periodic signals without error, but jitter and outliers



(a) GoogleTalk Video



(b) iChat Audio

Figure 3.11: Cumulative distribution functions of the relative error using ARMA predictors.

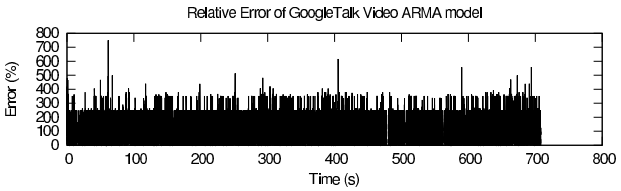
in the data cause the ARMA model to have large errors. Despite this, its performance is reasonably good, with the VoIP and video-conference flows showing low prediction error and the Crawdad flows showing high prediction error (see Table 3.4 and Figures 3.11 and 3.12).

Table 3.4: ARMAX model-based prediction quality

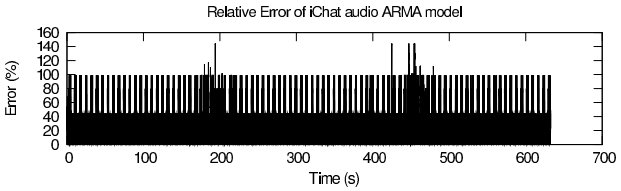
Flow	Median error (in ms)	85%ile error (in ms)
GoogleTalk audio send	7.54	33.15
GoogleTalk video send	8.13	49.41
iChat audio send	8.36	18.45
iChat video send	0.78	2.34
Skype audio send	0.03	1.53
Skype video send	0.0022	12.86

The entropy based predictor on the other hand does not assume a model. It has a table of bit patterns of length equal to the memory size (typically 15 in the results presented in this chapter). During the training stage, the table is filled and with each bit pattern there is an associated probability of the next bit being a 0 or 1. It is using that probability that the entropy estimator predicts, and we evaluate its performance during both the training and testing phases in Table 3.5.

The results are in agreement with the ARMA model predictor results. The GoogleTalk video



(a) GoogleTalk Video



(b) iChat Audio

Figure 3.12: ARMA estimated relative error.

test error being higher than the training error is most likely due to jitter in the data.

Table 3.5: Train and test errors of Entropy based predictor

Flow	Train error	Test error
GoogleTalk audio send	6.28%	11.66%
GoogleTalk video send	7.50%	13.62%
iChat audio send	6.02%	5.97%
iChat video send	4.22%	4.48%
Skype audio send	2.76%	3.20%
Skype video send	5.15%	0.16%

3.7 Applications of entropy estimator

The "entropy fingerprints" that we get from the Plug-in Packet Timing Entropy (PPTEn) estimator summarize many important characteristics of network application flows. Not only can we compare applications on the basis of peak entropy, but we can also categorize them according to number of entropy peaks, the bin size τ of the entropy peaks, and even the range of τ over which the entropy is above some threshold (which is smaller, for example, for real-time applications than media streaming appli-

cations). Therefore, these entropy fingerprints find applications in a number of contexts where traffic categorization or classification is important functionality and/or can improve performance. Below we elaborate on a few examples.

Intrusion detection: Previous work ([53, 54, 55, 56, 57, 27]) have shown uses of entropy estimation in traffic anomaly and intrusion detection. Although our approach is focused at estimating entropy at the per-flow level, it can be applied to intrusion detection as well. For example, suppose that a company wants to filter all egress traffic, e.g., to protect against corporate espionage. It puts in place a firewall to only allow outbound e-mail and HTTP traffic. However, an insider has found a way to hide a file transfer in an HTTP stream. If the corporate firewall uses the proposed PPTEn fingerprints of all egress flows that traverse it, it will be able to red-flag these "non-conforming" flows as their entropy fingerprint will look very different from that of regular HTTP traffic.

Admission control: PPTEn fingerprints can also be used in controlling admission of application flows into the network. For example, in typical infrastructure-based wireless networks, network traffic traverses the access point(s). PPTEn entropy fingerprints can be used by an access point to identify the traffic class of each traversing flow. If conditions allow, the access point can admit the flow by forwarding its packets. If the flow presents an unsustainable load on the network, the access point can dismiss the flow by dropping its packets. The entropy fingerprints enable admission control policies to be carried out by network ingress devices such as access points transparently to end hosts. It is important to note that, by using PPTEn entropy fingerprints, no details about the nature or characteristics of the network traffic are required from the application – instead they are inferred from the dynamically generated entropy fingerprints using a database of known traffic fingerprint characteristics.

Traffic scheduling, traffic engineering, and bandwidth allocation: Similarly, network appliances such as routers, switches, or access points can perform traffic scheduling and bandwidth allocation on the basis of PPTEn fingerprints. The router or access point can use the entropy fingerprints to group active flows into different traffic classes and divide available bandwidth according to some set

of rules. Medium access scheduling can also be done on the basis of the bandwidth allocation for each traffic class and each traffic source on the network.

3.8 Conclusion

In this chapter we showed that entropy estimation works well for measuring the complexity of per-application network traffic. We presented results of using two entropy estimation approaches – our own PPTEn estimator and the SampEn estimator – and showed that the output of our PPTEn entropy estimator provides more information on the application behavior and can more readily be used to compare per-flow network traffic complexities.

The PPTEn estimates corresponded almost exactly to the network traffic complexity ordering we came up with based on visual analysis of the network traffic from Skype, GoogleTalk, iChat, Hulu.com, ABC.com, and Netflix.com. In addition, the PPTEn estimates over a range τ highlight many application characteristics, some of which are even too subtle for visual observation. We refer to the entropy estimates as "entropy fingerprints" because of how closely related to the flow characteristics they are.

The fact that so many application characteristics are reflected in its estimates along with its forecasting potential makes PPTEn well suited for use in the realm of traffic scheduling and admission control.

Chapter 4

Design of a Traffic Forecasting Medium Access Control Protocol

4.1 Introduction

Prior research has shown that schedule-based medium access control (MAC) approaches provide efficient channel utilization, lend themselves well to reducing energy consumption by eliminating idle-listening, and ensure a deterministic level of service to the users of the medium ([60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70]). However, one drawback of schedule-based MACs is that their average packet delivery delay is higher than that of their contention-based counterparts. This chapter describes the TRAffic FORecasting Medium Access protocol, or TRANSFORMA, a novel schedule-based MAC protocol that employs per-application traffic forecasting to significantly reduce delivery delay. Through its traffic forecasting capability, TRANSFORMA provides each data flow transmission opportunities that are: (1) commensurate with the flow's data generation patterns and (2) exhibit adequate temporal distribution.

TRANSFORMA is motivated by the clear upward trend in the number of communication and entertainment applications available over the Internet. This trend is visible today with services like Skype, YouTube, Hulu, and Netflix, to name just a few. Teleconferencing and distance learning applications are also becoming more popular. With the proliferation of smart phones, ambient computing applications that have hitherto only existed in research circles are soon likely to become mainstream as well.

IEEE 802.11 is the wireless MAC currently prevalent in these application domains due to its cost effectiveness and wide availability. However, as high data-rate, real-time applications continue to evolve, they will put more strain on the wireless infrastructure, especially at the edges of the Internet. With an eye to the future, the MAC framework we present in this chapter aims to fill a future niche wherein applications that transmit and receive media-rich, delay-sensitive content can rely on a robust and decentralized ad hoc wireless network substrate that delivers good performance and degrades gracefully and predictably under load.

Several studies of residential broadband traffic have shown that the two largest bandwidth consumers are peer-to-peer traffic and HTTP traffic [71],[72]. However, HTTP is no longer a protocol used just to deliver Web pages with text and images. HTTP traffic is made up of many distinct types of application traffic including video and interactive Web applications. Thanks to Adobe Flash and the HTML 5 standard, the multimedia content of the Web is constantly increasing. To receive adequate medium access service, expecting the application (in this case the Web browser) to inform the MAC layer of the traffic characteristics of each of its flows is not realistic. Instead, the MAC layer should detect the properties of each flow transparently and adapt its level of service accordingly.

TRANSFORMA attempts to do exactly that by observing an application flow, learning its pattern (if one exists), and "forecasting" the flow's future behavior based on the observed one. In its current implementation, TRANSFORMA's forecaster examines the packet arrival process of each application flow and determines the corresponding per-flow inter-packet arrival times. It will then use this informa-

tion to establish the flow's medium access schedule. TRANSFORMA operates under the assumption that applications that place more stringent requirements, e.g., higher data rates and delay sensitivity have forecastable network usage patterns. It turns out that many current applications fall under this category: Skype, iChat, and Google Talk are VoIP and video-conferencing applications and naturally exhibit this kind of forecastable behavior. Additionally, non-real-time media streaming applications such as Hulu, Netflix, and iTunes do also.

The simulation results show that given a heterogeneous collection of flows TRANSFORMA can detect the periodicity of each flow and prioritize the channel access in such a way as to keep delays smaller than inter-packet times. The results also show that TRANSFORMA can use data rate forecasts of application flows and provide resources proportionally to every flow even as load increases. As a result, TRANSFORMA's delay performance is superior to DYNAMMA whenever flows are non-homogenous and superior to 802.11 at higher network loads.

The rest of this chapter is organized as follows: in Section 4.3 we describe the design of TRANSFORMA. In Section 4.4 we evaluate TRANSFORMA's performance through simulation, and Section 4.5 concludes the chapter.

4.2 Related Work

The energy efficiency potential of scheduled medium access is the guiding motivation behind a number of MAC protocols, predominately in the area of wireless sensor networks [60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70].

DYNAMMA is a schedule-based MAC protocol that adapts slot allocation to the current traffic at each node [70]. Through beacons exchanged at the start of each superframe, nodes in DYNAMMA establish 2-hop traffic information. Based on traffic information it collects, a node performs per-slot distributed elections of a transmitter and receiver. Although TRANSFORMA shares some common features

with DYNAMMA, (e.g., it uses the same superframe structure which is inspired by the WiMedia MAC standard [3]), it has very distinct performance characteristics by design. In particular, TRANSFORMA yields lower delays by allocating periodic transmission opportunities to flows at a rate corresponding to their modal packet inter-arrival time. This turned out to also give TRANSFORMA the ability to prioritize traffic in a manner that is favorable to "real-time" flows such as Skype, iChat and GoogleTalk.

Application awareness at the MAC layer has been advocated in the design of Rendezvous [73] and Sticky CSMA/CA [74]. Rendezvous is a MAC protocol that tries to take advantage of the periodic characteristics of "real-time" traffic. It is a contention based MAC in which the inter-arrival time of an application's packet is used to set up periodic reservations. Once reservations are established, a node gets priority on the channel during those times. Data can be transmitted outside of reservations using opportunistic, i.e., random access mode which uses the same approach as 802.11 to arbitrate access to the channel. By using knowledge about the application, Rendezvous can achieve better energy efficiency while maintaining adequate levels of service. While Rendezvous relies on the upper layers to provide the MAC with information about the application's traffic, TRANSFORMA is designed to extract it directly from the flow of packets. This also ensures that TRANSFORMA automatically adjusts to traffic changes.

In Sticky CSMA/CA [74], it is assumed that real-time flows are periodic by design or traffic-shaping. Once such flows acquire the wireless medium, they "stick" to a periodic schedule. Their neighbors can detect this schedule and avoid interfering with it. Delay-insensitive traffic only attempts to acquire the medium outside of these schedules thereby granting real-time traffic a higher level of service.

Despite their application awareness, Rendezvous and Sticky CSMA/CA are contention-based MAC protocols and thus susceptible to the performance limitations that go with that class of protocols. In designing TRANSFORMA, our goal was to have a MAC that is schedule-based and thus possesses all the benefits thereof while leveraging application awareness to improve delay performance.

Although often dismissed as impractical because of their complexity, the existence of a variety of real implementations – such as Soft-TDMAC [75], MadMAC [76], FreeMAC [77] and OverlayMAC

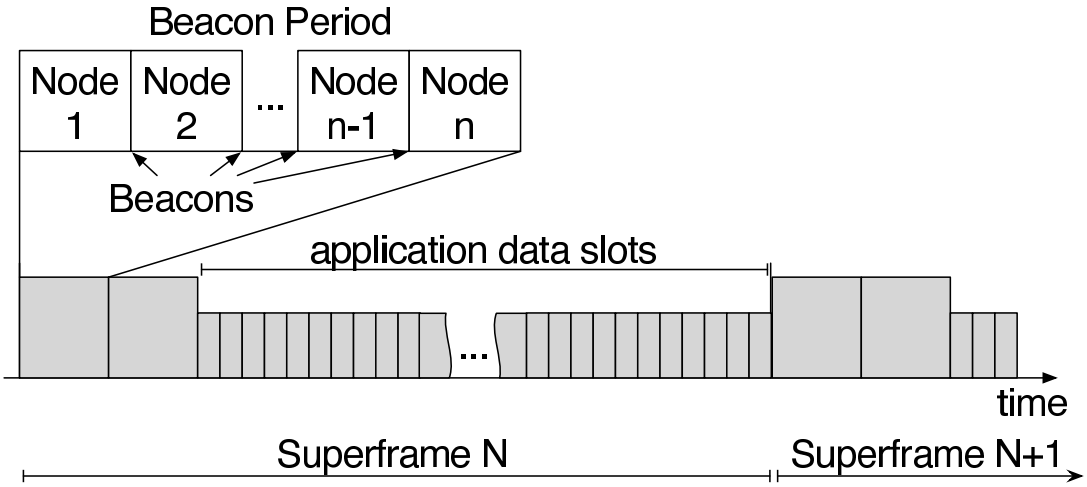


Figure 4.1: TRANSFORMA's superframe structure

[78] – show that schedule-based protocols can be realized in practice.

4.3 TRANSFORMA

4.3.1 Protocol Overview

TRANSFORMA provides collision-free medium access using a distributed scheduling algorithm based on forecasts of per-flow packet inter-arrival times. Consequently, TRANSFORMA's schedules not only adapt to application-level traffic but also do so in a proactive fashion. In other words, TRANSFORMA tries to anticipate the workload at each node and sets transmission schedules accordingly. This is in contrast to "traditional" scheduled access protocols (e.g., DYNAMMA [70]), which set schedules reactively and thus incur considerable delay. TRANSFORMA's proactive approach to scheduling is accomplished using a traffic forecaster which determines packet inter-arrival times for each application flow at every node. Nodes periodically exchange 2-hop traffic-forecast information. This information is used to select one or more non-conflicting transmitter-receiver pairs for each data transmission slot. Nodes periodically exchange 2-hop neighborhood traffic-forecast information. This

information is used to select one or more non-conflicting transmitter and receiver pairs during each data transmission slot.

TRANSFORMA assumes that a single channel is shared between control and data packets. The channel is time slotted and slots are grouped into `superframes`, each of which starts with 2 `beacon-periods` (illustrated in Figure 4.1). During a beacon period every node transmits information about its own flows and those of its 1-hop neighbors. Information propagates one hop per beacon period, thus after two beacon periods 2-hop information reaches all nodes in a 2-hop neighborhood. Subsequent slots in the superframe are used for transmission of application data and are arbitrated by TRANSFORMA's Flow Selection Algorithm (detailed in Section 4.3.4). More details on TRANSFORMA's time slot organization will be presented in Section 4.3.2.

In order to address the specific bandwidth needs of each application and yet maintain network utilization at adequate levels, TRANSFORMA employs a novel approach to medium scheduling based on traffic forecasting. TRANSFORMA's forecaster examines each packet arrival of a flow to adjust its forecast of that flow's data rate. The resulting data rate forecast is then used to provide the corresponding application flow the right amount of slots. In TRANSFORMA, a flow is defined by its source and destination addresses as well as transport layer port numbers. Section 4.3.3 goes into more detail on how the forecaster works and how it is integrated into the operation of TRANSFORMA.

TRANSFORMA can be broken down into three main components: the **Control Plane**, the **Traffic Forecaster**, and the **Flow Selection Algorithm**. We will discuss these in detail in Sections 4.3.2, 4.3.3 and 4.3.4 respectively. The **Control Plane** provides the infrastructure within which the other components of TRANSFORMA operate. It defines the layout of the superframe and the messaging that is used to disseminate traffic forecasts among the 2-hop neighborhood. The job of the **Traffic Forecaster** is to forecast the data rate of each flow and provide this forecast for dissemination to other nodes to be used for scheduling slots. The **Flow Selection Algorithm** is responsible for taking the per-flow forecasts and using them to provide a collision-free arbitration of the medium.

4.3.2 TRANSFORMA's Control Plane

TRANSFORMA defines basic rules for accessing the channel and uses one control message: the TRANSFORMA beacon. TRANSFORMA's channel access rules are:

- Nodes must send a beacon during beacon periods to be able to access the channel.
- Nodes must receive all beacons (barring errors) sent during beacon periods to be able to access the channel.
- Nodes must select a color not used by any other node in their 2-hop neighborhood and advertise it in their beacon (explained below).
- Nodes can only send a beacon during their assigned beacon slot in a beacon period.
- Nodes must execute the **Flow Selection Algorithm** during each data slot to determine whether they will receive, transmit, or sleep during that slot.
- Reception begins at the start of a slot.
- Transmission begins some guard time after the start of a slot.

TRANSFORMA uses a time-slotted channel access approach, the structure of which is illustrated in Figure 4.1. Slots are grouped into superframes, which repeat in time. At the beginning of each superframe are two beacon periods during each of which all nodes get an opportunity to send their beacon.

The purpose of the beacon, pictured in Figure 4.2, is to facilitate synchronization among nodes as well as topology– and traffic discovery. Like DYNAMMA [70], TRANSFORMA's use of beacons for topology management and synchronization is inspired by/emulates the WiMedia MAC [3]. In order to join the network, a node listens to the channel for at least two superframe durations or until hearing a full beacon period. It picks a random available beacon slot and transmits its beacon in that slot starting

in the next beacon period. A node can detect a beacon collision by inspecting the beacons of its 1-hop neighbors, who should mention the node in their beacons. If a node does not appear in its neighbors' beacons, or if there is disagreement among them, then there has been a beacon collision and the node must choose a new beacon slot.

Nodes in TRANSFORMA have colors. Colors are chosen such that no two nodes in the same 2-hop neighborhood can have the same color. This means that nodes of the same color are free to transmit simultaneously without causing collisions and this fact is used in the scheduling algorithm. To keep color assignment optimal, each node must choose the lowest possible index color that is not already used in its two hop neighborhood.

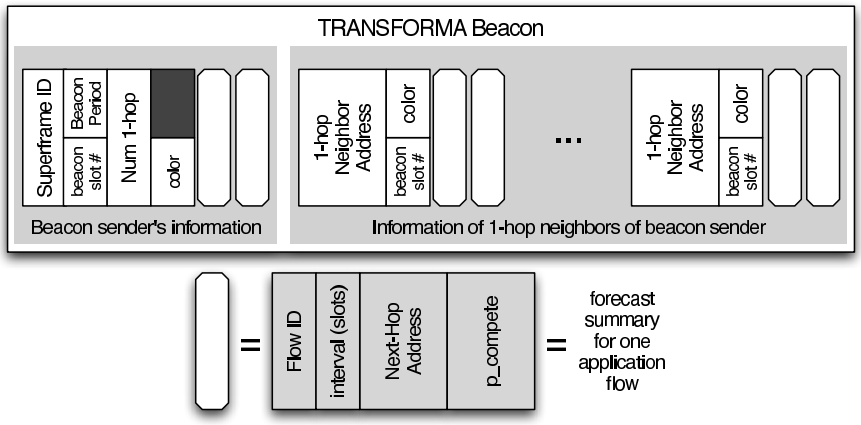


Figure 4.2: The TRANSFORMA beacon holds neighborhood information of the sending node and forecast summaries that need to be known by all nodes within 2-hops of the sender.

When the network layer passes an outgoing packet down to TRANSFORMA, the corresponding application flow to which the packet belongs is identified using the packet's source address, source port number, destination address, and destination port number) 4-tuple and the traffic forecaster is notified of the new packet arrival. If the packet does not have a source and destination port (i.e. no TCP or UDP header), it belongs to a special flow that each node can have up to one instance of. The forecaster makes available the most recent data rate forecast for each application flow. Each node maintains

a list of outgoing application flows, $\mathbf{OF}[\,]$. Each entry in the $\mathbf{OF}[\,]$ list has these properties: *slots per superframe*, the *source node address*, the *source application port number*, the *destination address*, the *destination port*, the flow's *next hop*, and a unique *flow ID*. For protocol scalability, nodes include a limited number of flow advertisements in each beacon – in TRANSFORMA's current implementation we use 2 flow advertisements per node. For scheduling purposes, flow advertisements include *flow ID*, *slots per sf*, *competition probability* (Section 4.3.4), and *next hop address*. When a node has more than 2 flows in its $\mathbf{OF}[\,]$ list, they are advertised in a round robin fashion.

Nodes maintain a second list of schedulable flows, $\mathbf{SF}[\,]$, which are populated by flow advertisements received from neighbors. The **Flow Selection Algorithm** described in Section 4.3.4 operates based on information in $\mathbf{SF}[\,]$.

4.3.3 Traffic Forecaster

Channel access in TRANSFORMA is scheduled based on the disseminated per-flow data rate forecasts. The TRANSFORMA protocol, however, is independent of the particular forecasting algorithm used. In our current implementation we use the well known *share algorithm* [79] which has shown excellent performance in a variety of on-line problems [80], [81], [82], [83]. One direction of future work we plan to explore is to investigate alternate forecasting approaches and compare the resulting performance.

The objective of the *share algorithm* is to pick from a set of experts, $\{x_1, x_2, ..., x_n\}$, the one whose output gives the smallest *loss*. The algorithm maintains a weight for each expert, $\{w_1, w_2, ..., w_n\}$, the value of which determines the impact that each expert's output has on the global output of the algorithm. The *share algorithm* redistributes the weight of experts whose *loss* is high to those experts with low *loss*. As a result, the algorithm quickly adapts to changes in the input (Figure 4.3).

In TRANSFORMA, the output of both the individual experts and the *share algorithm* represent a data rate forecast expressed in units of slots per superframe. In our implementation of the forecaster, the

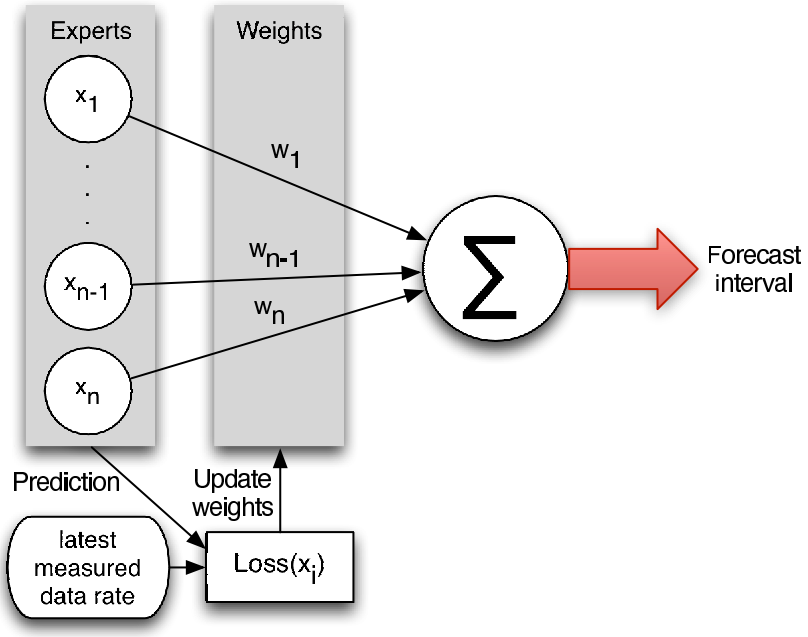


Figure 4.3: The **share algorithm** used by TRANSFORMA

values of the experts, $\{x_1, x_2, ..., x_n\}$, are distributed linearly between a data rate of one slot per super-frame and the maximum forecastable rate and consequently the output of the forecaster also falls within that range. The *loss* function penalizes experts proportionally to the difference between the measured data rate, λ, and their value, x_i.

TRANSFORMA maintains for each application flow its own forecast interval, $\hat{\lambda}$, and experts' weights, $\{w_1, w_2, ..., w_n\}$. When a packet arrives from the network layer, TRANSFORMA matches the packet's identification, i.e., (source address, source port number, destination address, and destination port number) tuple to the application flow it belongs to and performs the following actions:

1. Computes the latest frame's data rate, λ:

$$\tau = t_{\text{(latest packet arrival)}} - t_{\text{(previous packet arrival)}}$$

$$\lambda = \frac{\text{frame size}}{\tau}$$

2. Calculates the *loss* of each expert, x_i:

$$Loss(x_i) = \begin{cases} \left(\frac{0.75(\lambda - x_i)}{\text{max rate}}\right)^2 & \text{if } \lambda \leq x_i \\ \left(\frac{(\lambda - x_i)}{\text{max rate}}\right)^2 & \text{if } \lambda > x_i \end{cases}$$

3. Reduces weights of poorly performing experts:

$$w_i' = w_i e^{-\eta Loss(x_i)}$$

4. Shares some of the remaining weights:

$$pool = \sum_{i=1}^{n} w_i'(1 - (1-\alpha)^{Loss(x_i)})$$
$$w_i'' = (1-\alpha)^{Loss(x_i)} w_i' + \frac{1}{n} pool$$

5. Calculates new forecaster output:

$$\hat{\lambda} = \frac{\sum_{i=0}^{n} w_i x_i}{\sum_{i=0}^{n} w_i}$$

The 0.75 constant in the loss function is an empirically determined value whose purpose is to penalize an expert that estimates too low more than an expert that estimates an equal amount too high. The reasoning is that allocating a few extra slots will help to absorb any bursty behavior whereas allocating too few slots can only cause buffers to grow. Figure 4.4 shows how the forecast data rate compares to the actual data rate of a Skype flow. The Skype trace was captured using `tcpdump` [84]; we then extracted the packet arrival times and packet sizes from the trace. In the plot, the forecast intentionally does not follow the instantaneous data rate of the flow; we can only use one forecast per superframe so it's not beneficial to have a more rapidly changing forecast. The η parameter controls the rate at which the algorithm adapts to the input and a value of 10 has been found empirically to provide

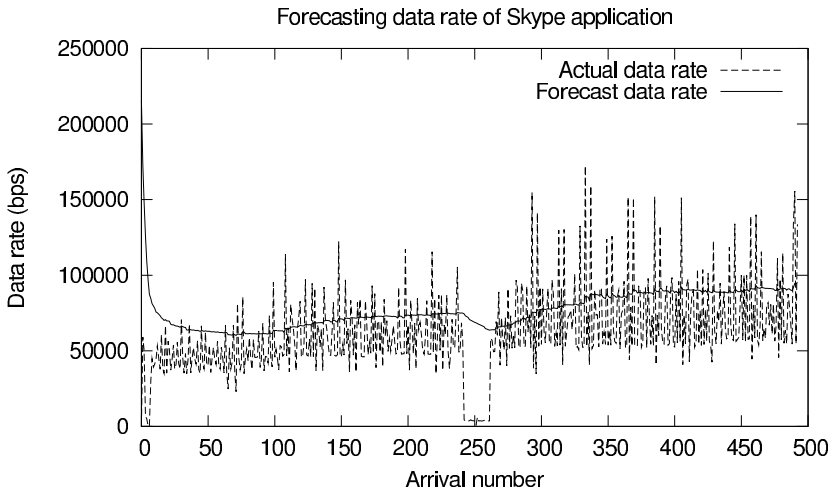


Figure 4.4: The data rate forecast compared to the measured instantaneous data rate of a Skype flow.

a good balance between a smooth yet responsive forecast. The α parameter controls how much of the losing experts' weight is shared with the winners. Setting this parameter too high causes weight to quickly shift from one expert to another and makes the forecast less smooth, but sharing is beneficial to the responsiveness of the forecaster, so there is a balance. We found that a value of 0.04 works well.

4.3.4 Flow Selection: Scheduling Medium Access

Once per-flow rate forecast information has been distributed around the network, the next challenge is how to use it to most effectively schedule the medium in a distributed manner – each node uses only local information to make decisions that do not contradict those of its neighbors. The scheduling algorithm in TRANSFORMA is designed with the goal of providing as many slots to each flow as its traffic forecast requires. When the load on the network makes this impossible, the scheduler shares the slots in a fair manner among all competing flows.

Flow Selection Algorithm

After a beacon exchange, each node in the network has a list of schedulable flows, $\mathbf{SF}[\,]$, in its 2-hop neighborhood. For each flow, the node knows the *source*, *flow ID*, *rate forecast*, and *competition probability*, P_c. Computation of P_c will be discussed in the next section. Every node knows the colors of its 2-hop neighbors and consequently the color of all the flows (a flow has the color of its source).

The first step is to compute for each flow in $\mathbf{SF}[\,]$ two pseudo random numbers based on a hash of the flow's information; *rand1* is a number in the range $[0, 1]$ and *rand2* is an integer in the range $[0, 2^{32} - 1]$.

$$\text{rand1} = \text{hash1}(\#_{sf} \oplus \#_{slot} \oplus \text{flow color} \oplus \text{flow ID} \oplus \text{seed})$$

$$\text{rand2} = \text{hash2}(\#_{sf} \oplus \#_{slot} \oplus \text{flow color} \oplus \text{flow ID} \oplus \text{seed})$$

As the hashes are being computed, any flow with $P_c \geq$ rand1 is placed in the set of *competing flows*, $\mathbf{CF}[\,]$.

The second step is to select from $\mathbf{CF}[\,]$ the flow(s) with the largest *rand2*. All of these flows must have the same color. If not, only those with the smallest color index are kept. The coloring scheme ensures that no two nodes in the same 2-hop neighborhood will have the same color, therefore the transmitters of these flows can safely transmit concurrently without causing collisions. If the node running this instance of the algorithm is the sender or receiver of a winning flow, it puts its radio in transmit or receive respectively. Otherwise it can sleep for the duration of the slot.

Computation of Competing Probability

We express the likelihood of a flow, f_i, winning a slot using the following equation:

$$P(f_i \text{ wins}) = P(f_i \text{ competes}) \cdot P(f_i \text{ wins} \mid f_i \text{ competes}) \tag{4.1}$$

The goal is to compute the value of $P(f_i \text{ competes})$ that would give us a $P(f_i \text{ wins})$ which corresponds to the forecast data rate for the flow. This value is dependent on all the flows that are trying to share the

medium.

If we define the random variable X to represent the number of flows competing for the slot, the conditional probability on the right side of Equation 4.1 can be expressed as

$$P(f_i \text{ wins} | f_i \text{ competes}) = \frac{1}{E[X |\ f_i \text{ competes}]} \tag{4.2}$$

The random variable $\mathbf{X}$ can be expressed as a sum of random variables, $\mathbf{I}_i$, each representing the contribution of flow f_i to $\mathbf{X}$.

$$E[I_j | f_i \text{ competes}] = \begin{cases} P(f_i \text{ competes}) & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \tag{4.3}$$

If there are n flows, knowing $E[X|f_i \text{ competes}] = \sum_{j=1}^{n} E[I_j|f_i \text{ competes}]$, Equation 4.2 becomes:

$$P(f_i \text{ wins} \mid f_i \text{ competes}) = \frac{1}{\left(\sum_{j=1}^{n} P(f_j \text{ competes})\right) - P(f_i \text{ competes}) + 1} \tag{4.4}$$

Now we rewrite Equation 4.1 as:

$$P(f_i \text{ wins}) = P(f_i \text{ competes}) \cdot \frac{1}{\varepsilon - P(f_i \text{ competes}) + 1} \tag{4.5}$$

$$\varepsilon = \sum_{j=1}^{n} P(f_j \text{ competes})$$

One final constraint enables us to solve for P(f_i competes): we find the flow with the greatest $P(f_i \text{ wins})$, call it f_{max} and set $P(f_{max} \text{ competes}) = 1$. Then we solve for ε:

$$\varepsilon = \frac{1}{P(f_{max} \text{ wins})} \tag{4.6}$$

Rearranging Equation 4.5, and having ε allows us to solve for P(f_i competes):

$$P(f_i \text{ competes}) = \frac{P(f_i \text{ wins})(1 + \varepsilon)}{1 + P(f_i \text{ wins})} \tag{4.7}$$

Each node uses Equations 4.7 and 4.6 to compute the $P(f_i \text{ competes})$ of its flows once per superframe.

Scaling P(win)

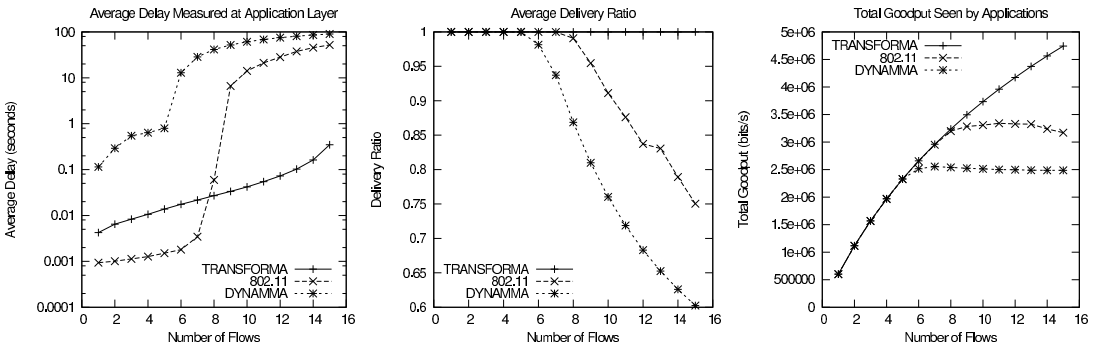
Although we could directly compute $P(f_i \text{ wins})$ from the data rate forecast of a flow and use that to compute the P(f_i competes), this wouldn't give the desired result in situations where the sum of the $P(f_i \text{ wins})$ of all the flows exceeds 1. To keep slot allocation working fairly under such circumstances, we always scale the probability such that $\sum_{i=1}^{n} P(f_i \text{ wins})_{scaled} = 1$.

4.4 Performance Evaluation

4.4.1 Experimental Setup

We evaluate the performance of TRANSFORMA using version 4.0 of the Qualnet [85] network simulator. As performance baseline, we chose one schedule-based and one contention-based MAC protocol to compare against TRANSFORMA. We selected DYNAMMA to serve the role of the schedule-based baseline protocol because it stands out as a general-purpose MAC protocol that has been shown to perform competitively against other schedule-based protocols [70]. IEEE 802.11 DCF [86] has been extensively used and studied and was therefore an obvious choice as the contention-based baseline.

When designing TRANSFORMA we targeted the niche of local area and enterprise networks such as those found in the home, office buildings and hospitals. For example, today's wireless home networks most often have an access point that serves as an internet gateway and all wireless communication goes through this gateway. It is not unlikely that as the number of devices in homes increase, a less centralized topology may serve them better. As the amount of multimedia and the ways in which to access it increase, the home network will increasingly be used to move data between devices in the home rather than just to and from the internet. With this in mind we devised two network topologies for our experiments (Figure 4.7). The first topology reflects a traditional hotspot network – we have 15 nodes distributed in a circle around a central node. We have spaced the nodes such that hidden terminals exist



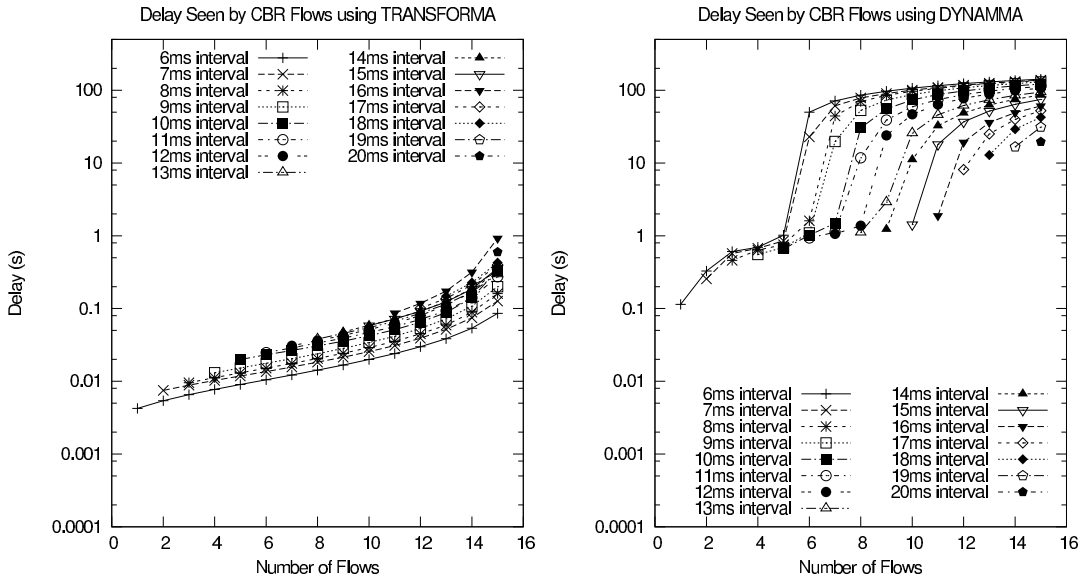
(a) (b) (c)

Figure 4.5: Average delay (a), packet delivery ratio (b), and total goodput (c) for heterogeneous flows in hot spot topology

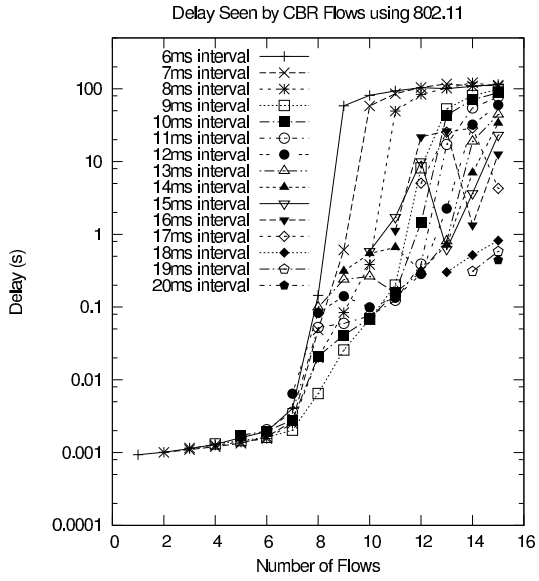
in the network. The second topology is a multihop grid with diagonal spacing set to the radio range. This topology represents a decentralized network and is large enough to provide some possibilities of spatial channel reuse. The grid topology also has multiple 2-hop neighborhoods, which is good for the purposes of stressing the schedule-based protocols.

In our simulations, all MAC protocols use the 802.11a physical layer operating in the 2.4GHz range with a data rate of 6.0Mbps. Qualnet does not have an 802.11g physical layer implementation. Instead using the 802.11a PHY in the 2.4GHz band approximates 802.11g. The 802.11 MAC is configured with all the default settings. DYNAMMA and TRANSFORMA are both configured with a 1024byte slot size and as close as possible to 1s superframe duration. Small differences in header sizes between DYNAMMA and TRANSFORMA mean that the slot duration is 1.422ms for TRANSFORMA compared to 1.458ms for DYNAMMA and that TRANSFORMA has 703 slots per superframe while DYNAMMA has 700. Both DYNAMMA and TRANSFORMA are configured to fit as many packets as possible into each slot, provided they belong to the same flow.

All experiments shown in this chapter use the UDP transport protocol. To represent a real-time flow such as Skype, UDP is the appropriate transport to use. For the background traffic and heteroge-



(a) (b)



(c)

Figure 4.6: Per flow delays using TRANSFORMA (a), DYNAMMA (b), and 802.11 DCF (c) in hot spot topology

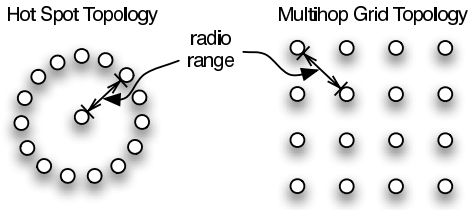


Figure 4.7: Two topologies used in our experiments.

neous flows using UDP gives us control over how heavily we load the network (TCP would back off under heavy load). Despite the feedback loop TCP uses in its congestion control function it operates without problems on top of TRANSFORMA. TRANSFORMA's traffic forecaster is tuned to ignore oscillations in the data rate and is slower to reduce the forecast data rate than it is to increase it. This prevents the forecaster from starving a TCP flow that has gone into congestion avoidance.

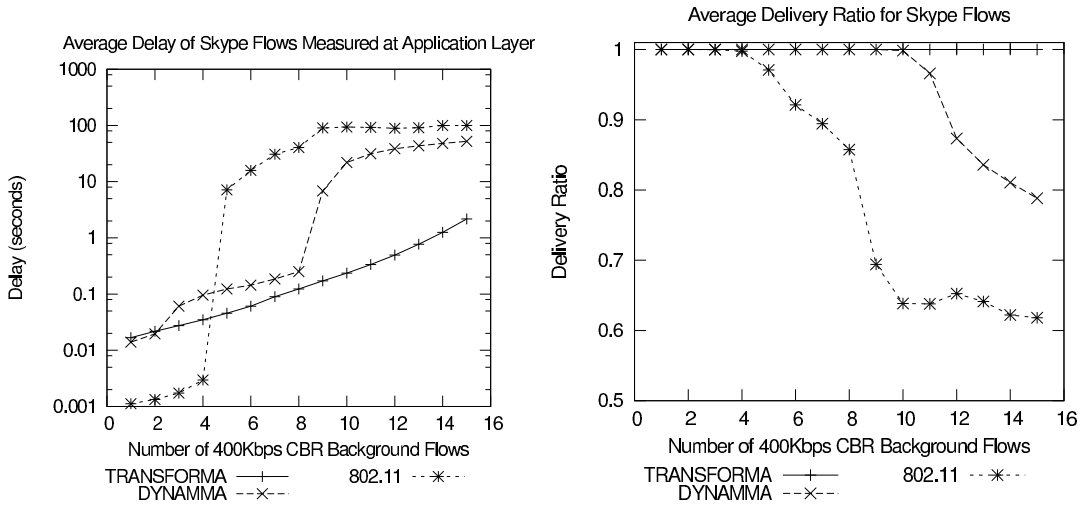
4.4.2 Hot Spot Topology

Heterogeneous flows

The network traffic in the first experiment consists of a number of heterogeneous CBR flows. We vary the load on the network by adding flows one by one. All the flows have a packet size of 450 bytes, but each flow has a different packet arrival interval. The first flow's packet arrival interval is 6ms and each successive flow has an interval 1ms larger than the previous one. To get all the nodes in the network involved, each node in the ring is the source of a single flow, and as flows are added they are distributed uniformly around the ring. The central node is the destination for all the flows.

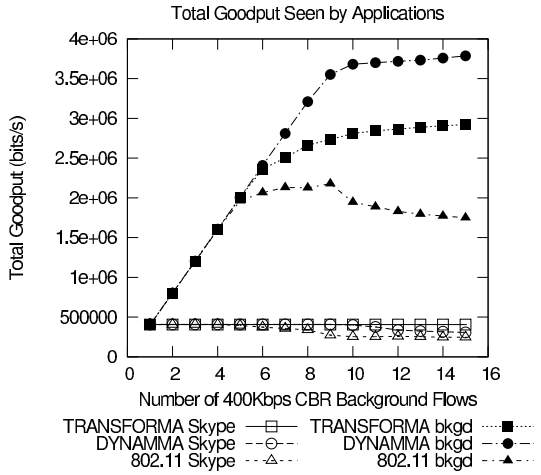
To measure the performance of TRANSFORMA and the other MACs, we use four metrics: average delay, per-flow delay, delivery ratio, and total goodput[1]. One of our main objectives is to minimize delay, so that metric is self evident. Delivery ratio is a strong indicator of the ability of a given MAC to cope with the traffic load. Goodput is a metric that reflects both delivery ratio and delay so it is very

[1] We define goodput as the number of bytes successfully received at the application layer divided by the time between the first and last packet.



(a)

(b)



(c)

Figure 4.8: Traffic delay (a) and delivery ratio (b) for Skype foreground traffic. Goodput (c) for Skype (foreground) flows and background flows.

useful in evaluating performance. We included a per-flow metric because the flows are heterogeneous and we want to compare how TRANSFORMA, DYNAMMA, and 802.11 deal with each of them. The total goodput is an indirect way of measuring channel utilization. In this scenario the maximum theoretical goodput is 6.0Mbps (the physical data rate) because flows are one hop and successful simultaneous transmissions are impossible. Transport, network and MAC layer headers make it impossible to reach this maximum, but we can still draw conclusions based on how close each MAC gets.

We ran the experiment with 10 seeds and present the averaged results in Figures 4.5–4.6. TRANSFORMA's average delay (Figure 4.5a) is one order of magnitude less that DYNAMMA's under low load and around two orders of magnitude lower as the load increases. TRANSFORMA's delay is higher than that of 802.11 until the 8th flow is added, at which point 802.11 begins to struggle with the load. Figure 4.6 shows that as the load increases, the flows that begin to suffer the most when using both 802.11 and DYNAMMA are the ones with the highest data rate. TRANSFORMA, on the other hand, scales back the number of slots each flow wins in equal proportion. This leads us to believe that in DYNAMMA, despite the 3 traffic classes, as the load increases all the flows end up in the highest class where they all win slots with equal likelyhood. This means that flows with more traffic will end up suffering first. In TRANSFORMA, the forecast of the flow's data rate and the total load in the network determine how many slots the flow will win each superframe; the more slots a flow wins, the closer together the slots will be on average and the lower the delay.

The plots of delivery ratio (Figure 4.5b) and total goodput (Figure 4.5c) show that TRANSFORMA outperforms DYNAMMA and 802.11 under high load and is able to use use a larger percentage of the available bandwidth. TRANSFORMA outperforms DYNAMMA because its level of control over how many slots each flow gets is greater than that of DYNAMMA. It outperforms 802.11 because it is collision free and thus better able to deal with high load.

Skype flows

We designed the second experiment to observe the effect that background traffic has on real-time flows when using TRANSFORMA and the two other MAC protocols. Instead of using a synthetic traffic generator to model real-time traffic, we used a real trace of a Skype phone call captured using `tcpdump` [84]. We then extracted the packet arrival times and packet sizes from the trace and fed them into Qualnet using its trace-based traffic generator. TRANSFORMA's traffic forecaster was able to identify the modal packet inter-arrival times for each of the three applications.

In this experiment there are 3 Skype calls that we term "foreground" traffic, and an increasing number of CBR "background" flows. Each Skype call is made up of two separate trace-based flows: one going into the center node and one going away from it. Here, as in the previous experiment, the center node plays the role of the internet gateway. Each background CBR flow has 200byte packets spaced at an interval of 4ms.

The delay plot in Figure 4.8a shows that TRANSFORMA is able to maintain a low delay for the foreground traffic while the delays of DYNAMMA and 802.11 increase sharply as the background load increases. Figure 4.8b shows that 802.11 drops packets due to high contention and DYNAMMA drops packets because it's buffers begin to overflow, whereas TRANSFORMA is able to keep its buffers from overflowing by allocating slots to each flow commensurately to its forecast; in other words, by allocating the right amount of resources to the right flows and being collision-free, TRANSFORMA's is able to outperform the other MACs.

4.4.3 Multihop Grid Topology

In this topology we decided to again use the heterogeneous flows traffic scenario. The challenge with the grid was adding flows in a systematic fashion so that the addition of a single flow didn't suddenly change the dynamics. Each application flow in this experiment traverses 3 hops to get from one

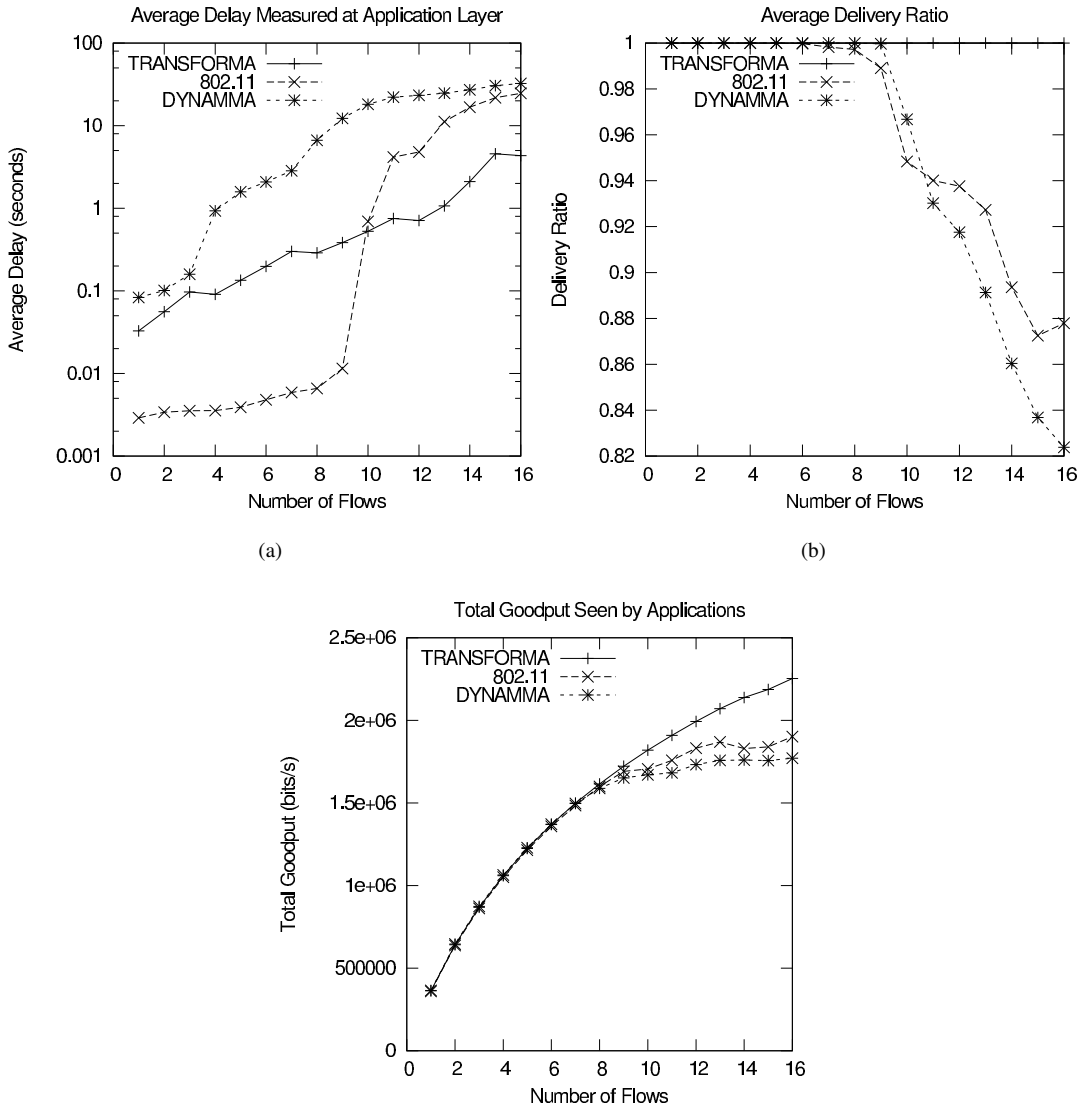


Figure 4.9: Average delay (a), packet delivery ratio (b), and total good put (c) of all heterogeneous flows using grid topology

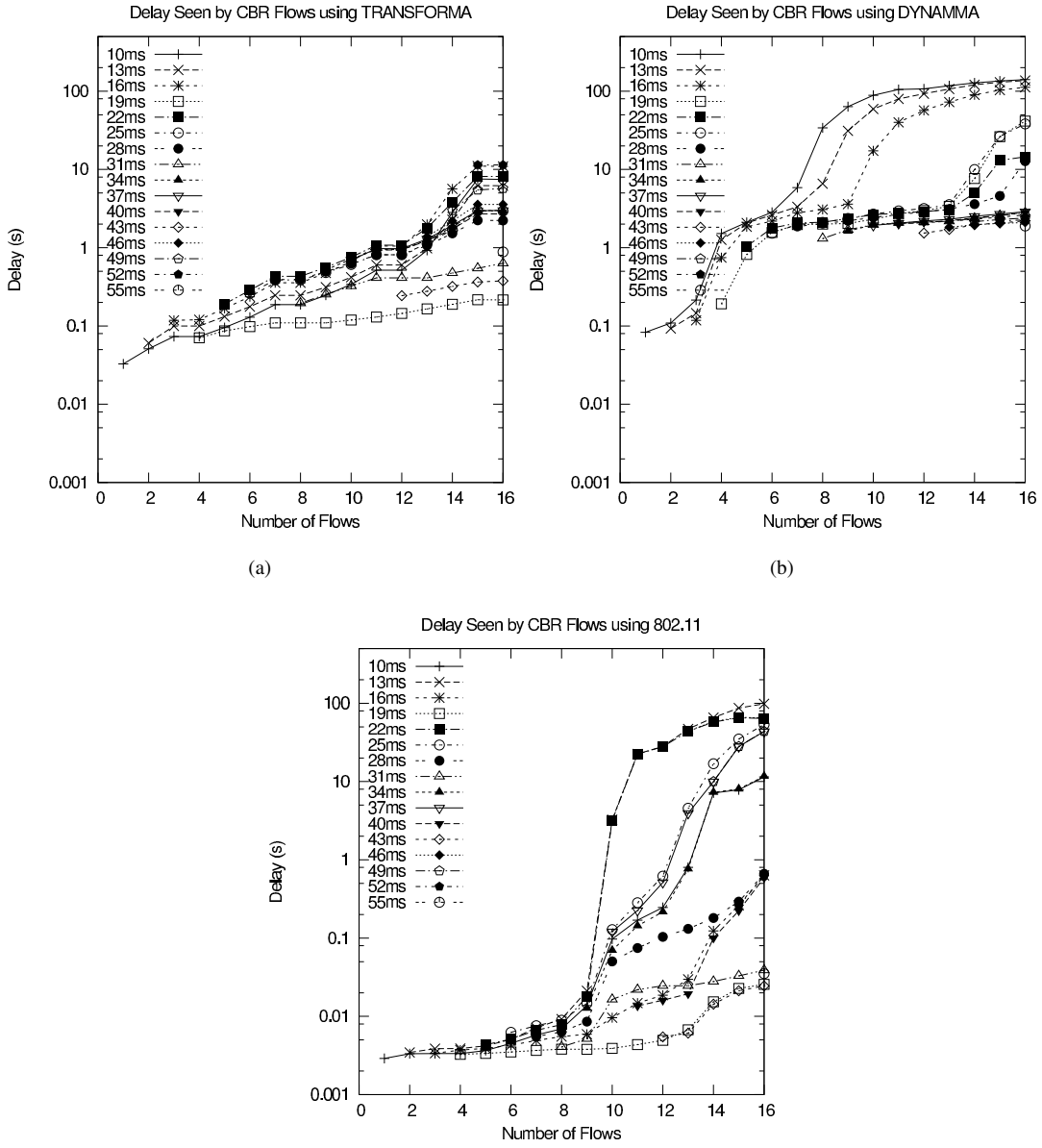


Figure 4.10: Per flow delays using TRANSFORMA (a), DYNAMMA (b), and 802.11 DCF (c) in multihop grid topology

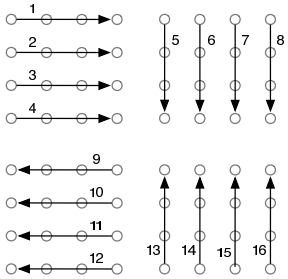


Figure 4.11: The order in which flows are added to the 4 by 4 grid

side of the 4 by 4 grid to the other. The flows were added in the order shown in Figure 4.11. Similarly to the hot spot topology version of this experiment, we ran the simulation with 10 seeds and evaluated the three MACs based on average delay, per-flow delay, delivery ratio, and total goodput (Figures 4.9–4.10). In this scenario, the first flow had a packet interval of 10ms and each successive flow had a 3ms larger interval. Taking into account the fact that each flow traverses 3 hops, the total load offered to the network is 6.8Mbps. This load is manageable because simultaneous transmissions are possible in this topology.

The results show that in the multihop grid topology TRANSFORMA once again outperforms DYNAMMA by an order of magnitude at almost all loads. The delay metrics shown in these graphs are application-layer delays, so all protocols will see their delays increased due to the multi hop nature of these flows: 802.11 will have to contend three separate times to get a packet from source to destination and DYNAMMA and TRANSFORMA have to schedule slots at each node along the way. When the load is low, contending has lower overhead than scheduling, so 802.11 does considerably better than either schedule-based approach. As the load increases, however, the overhead for contention surpasses that of scheduling. Figure 4.10b clearly shows that with DYNAMMA the flows with highest data rate feel the effects of the higher data rate first. TRANSFORMA once again more appropriately allocates slots so that all the flows share the effects of the increasing load. The relationship between flow rate and delay that was so clearly evident in TRANSFORMA's curves in the hot spot topology (Figure 4.6a) has been obscured in this topology by the interplay between flows. TRANSFORMA's use of node color

when computing competition probabilities for each flow and subsequently computing flow winners for each slot has the side effect that low rate flows with the same color as high rate flows can sometimes experience lower than expected delays.

In this experiment TRANSFORMA once again retains its high delivery ratio while 802.11 drops packets due to contention and DYNAMMA drops them due to buffer overflow of the high rate flows. Consequently, the maximum achievable total goodput of TRANSFORMA is the highest of the three. Because all the flows in this experiment traverse 3 hops, the total goodput can be multiplied by 3 to get a lower bound on how much data has to be transmitted to achieve that goodput.

4.4.4 What about 802.11e?

802.11 EDCA provides quality of service enhancements to the standard 802.11 DCF. These enhancements allow 802.11 EDCA to prioritize contention based on 4 access categories, each of which has a different priority. It is the responsibility of the higher layers to select the access category for each packet and place it in the corresponding queue. We assert that 802.11e will perform similarly to 802.11 under high loads and further, a fair comparison with TRANSFORMA was not possible given that 802.11e does not determine the access category of traffic on its own.

4.5 Conclusion

In this chapter we presented TRANSFORMA, a collision-free, scheduled-based medium access control protocol that employs a novel approach to medium access based on traffic forecasting. TRANSFORMA's traffic forecaster identifies patterns in application flows and uses this information to schedule access to the medium most effectively. By doing so, TRANSFORMA tries to anticipate the workload at each node and sets transmission schedules accordingly. This is in contrast to "traditional" scheduled access protocols (e.g., DYNAMMA [70], which set schedules reactively and thus incur con-

siderably higher delays.

We showed through simulations that TRANSFORMA is able to identify the traffic patterns of various kinds of flows and use that information to schedule them, assuring each flow a packet delay on the order of its packet inter-arrival time. Our results also showed that TRANSFORMA is able to schedule real-time flows alongside background traffic with less adverse effects on the real time flows' delay than 802.11 and DYNAMMA.

Moving forward, our future work plans include: performing live experiments on a testbed as a way to cross-validate our simulation results and developing an analytical model to derive performance bounds for TRANSFORMA and as a way to validate our simulations.

Chapter 5

Implementation of TRANSFORMA

For the culmination of this dissertation work, TRANSFORMA was implemented using the STX1201 wireless modem development platform from Starix Technology [87]. The implementation consists of custom firmware for the STX1201, a matching Linux network device driver and a forecasting daemon.

5.1 The Starix 1201 Development Platform

The Starix 1201 Development Platform is built around the RTU7105 single-CMOS chip, which contains:

- Ultra wideband baseband and RF transceiver
- USB, UART, I2C, and general purpose I/O interfaces
- Microprocessor (132MHz)

The MAC of the RTU7105 is largely implemented in firmware, which runs on the RTU7105's processor, with hardware support for the very low level operations such as transmitting beacons. Out of the box, the board implements the WiMedia MAC [3]. The fact that the firmware of the chip defines in

Figure 5.1: The Starix 1201 Development Platform

large part how the MAC operates, combined with the Software Development SDK, makes the platform flexible and well suited to MAC protocol development. Given that a lot of the infrastructure of the WiMedia MAC such as beaconing, joining the network and synchronizing node clocks is directly usable by TRANSFORMA, the platform is very well suited to our implementation.

5.2 Implementation Goal

Our goal with this implementation was to make a fully usable network interface that runs TRANSFORMA, enabling experimentation with TRANSFORMA on real network applications.

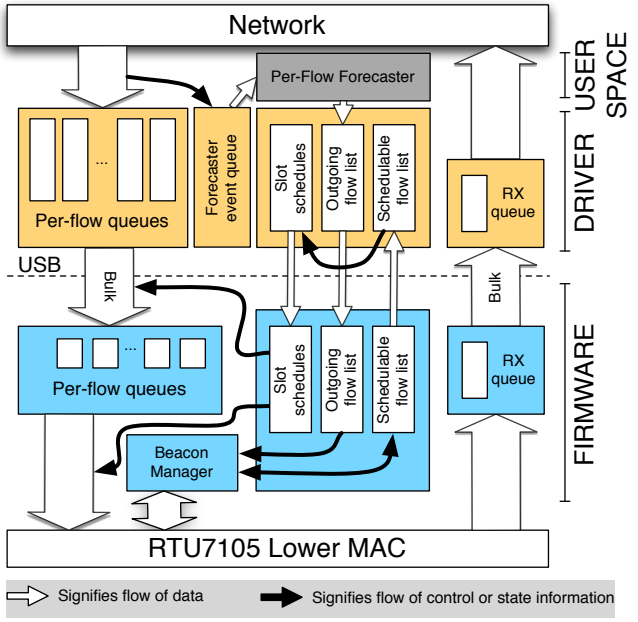


Figure 5.2: Implementation block diagram

The overall architecture of our implementation was driven by the capabilities of the STX1201 modem platform. The computation and memory resources of the STX1201 platform are insufficient to implement TRANSFORMA entirely in firmware, so our implementation is a combination of firmware, a Linux driver and a user-space daemon (Fig 5.2). The role of the driver is to maintain the per-flow queues of TRANSFORMA and carry out most of the computation required to elect a winner for each slot. The user-space daemon is needed to perform the floating point computations of the forecaster that cannot be done in kernel-space. The firmware's main task, aside from sending and receiving data, is to advertise outgoing flow information in its beacon and collect neighborhood flow information from neighbors' beacons to build the schedulable flow list, which is used by the driver to compute the slot schedules. Each of these three pieces of TRANSFORMA will be described in Sections 5.3, 5.4, and 5.5 and then Section 5.6 will elaborate on how they all work in concert to enable data to flow through a TRANSFORMA-powered wireless network interface.

5.3 Firmware

The STX1201 modem's firmware plays a key role in enabling this implementation of TRANSFORMA. It is responsible for executing the more time sensitive operations of the protocol such as sending beacons every superframe and transmitting data at precisely the right time in each 256 μs data slot. The STX1201 was designed to run a scheduled MAC protocol and the software APIs provided with it significantly ease the effort needed to implement TRANSFORMA.

The firmware cannot implement all of TRANSFORMA on its own. It has limited memory for buffering data and was not designed to be able to perform any kind of computation intensive task other than moving data around. The functionality performed by the firmware can be divided into three main areas: communicating with the USB host that it is connected to, exchanging beacons with nodes within radio range, and transmitting data to and receiving data from neighboring nodes. The STX1201 modem completes these tasks using a combination of hardware and firmware. The low-level details are abstracted away through the API provided in the SDK that comes with the modem, however an good understanding of USB and the WiMedia MAC that the modem was designed to implement are required for successful use of the platform.

5.3.1 USB Communication

The STX1201 modem is connected to the host computer using a High Speed USB connection (USB 2.0) [88]. The maximum throughput of this link is 480 Mbps, not accounting for protocol overhead. This total throughput is further shared between IN and OUT (of the host) data directions and may be shared by multiple devices that are connected to the same USB host controller. While detailed information on the USB 2.0 specification can be found at the www.usb.org website, a brief overview will be provided here to aid the reader's understanding of this aspect of the implementation design.

The USB protocol defines virtual *pipes*, each end of which terminates at an *endpoint*. A col-

lection of these endpoints forms an *interface*. There are 4 kinds of pipes:

1. Control
2. Bulk
3. Interrupt
4. Isochronous

With the exception of the Control pipe, which is bidirectional, pipes are unidirectional – they are either *IN* to the host or *OUT* of the host. Every USB device must have at least a Control pipe because it is through this pipe that the host gets enough information about the device to be able to match it to a driver. Each type of pipe has characteristics that make it suited for certain types of data traffic.

The Bulk pipe is the most commonly used type of pipe. It is a stream pipe that provides reliable (in order) delivery of data but no guarantee on latency. This type of pipe can use any available bandwidth but reserves none. A simple network adapter can be implemented with just a pair of Bulk pipes (an IN and an OUT).

Interrupt pipes are meant for small amounts of data or indications that are sent infrequently but with some latency guarantees. Every interrupt pipe specifies a period with which it must be serviced and the host controller will set up a reservation to ensure that the pipe gets serviced at least once per period.

Isochronous pipes have latency guarantees, similar to Interrupt pipes, but they are intended to sustain higher data bandwidth. What differentiates them from Bulk pipes is that they do not provide data delivery guarantees. They are commonly used for video and audio streams of webcams, an application where some data loss is tolerable as long as the latency is bounded.

In the design of modem-to-driver communication of TRANSFORMA we decided that the benefits of the Bulk pipes outweighed their drawbacks when it comes to delivering data to and from the driver. Had we chosen to use Isochronous pipes, we would be forced to address the issue of retransmitting

data between the driver and the firmware at a layer above USB – an approach that is not without its own drawbacks. For the time sensitive signaling that needs to happen between the modem and driver we used a pair of Interrupt pipes.

5.3.2 Establishing Superframes

TRANSFORMA uses the same superframe structure as the WiMedia MAC that the STX1201 modem was designed to implement. This allows us to leverage the hardware facilities already in place that allow tight time synchronization between wireless peers and coordinate beacon exchange during the beacon period of each superframe. The superframe structure is depicted in Figure 5.3. One difference between the superframe pictured here and the one used in our simulations is that in the STX1201 implementation we use a single beacon period per superframe. Although the back-to-back beacon period can provide single-superframe propagation of 2-hop neighborhood information, it is difficult to alter the STX1201 to support it, and given the short superframes (64ms), the benefit is small. The number of slots in each superframe taken up by the beacon period depends on the density of the wireless neighborhood. It is around 16 slots in the experiments we have carried out. The WiMedia MAC protocol dictates that beacons always be sent at 53.3 Mbps, and we did not modify this. The data rate used during data slots can be increased up to 200Mbps on the STX1201. We chose a cautious 80Mbps. This allows us to send 2000 bytes per data slot, leaving some guard space. We did not want to pack the slots completely because we did not have the wireless sniffer that would be necessary to troubleshoot data loss due to overlapping slots if it was to occur.

Beacons in the WiMedia MAC protocol contain blocks of data called *Information Elements* (IEs). Some IEs are used by the MAC itself to coordinate Beacon slot assignments and expansion or contraction of the beacon period. Other IEs can be application-specific. To implement TRANSFORMA, we use two such IEs in each node's beacon. The first holds information about the node that is sending the beacon such as:

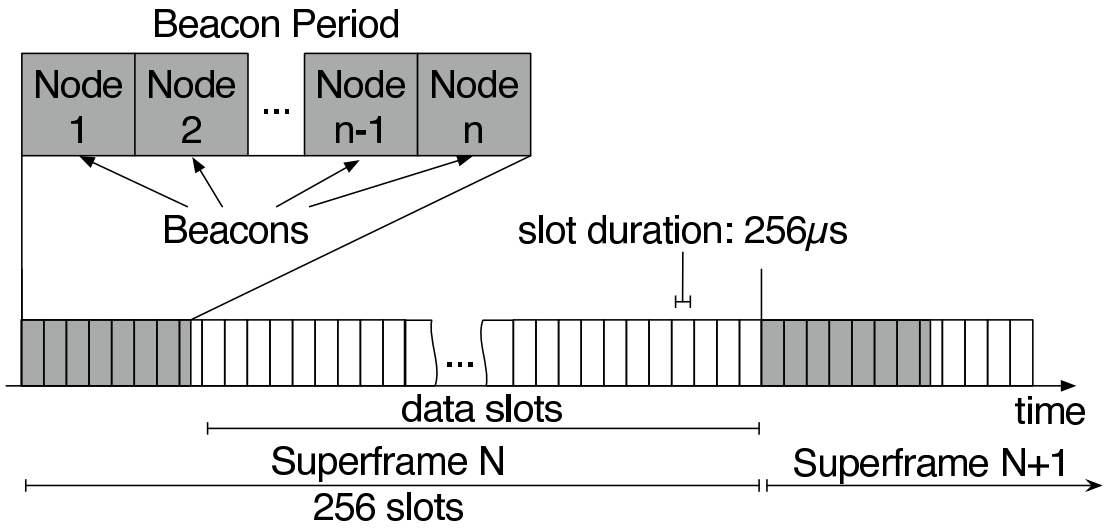


Figure 5.3: TRANSFORMA's superframe structure

- Device ID
- Superframe ID
- Node color
- Some number of flow advertisements up to max per IE (currently set to 2):
  - Flow ID
  - Flow forecast (in units of slots per second)
  - Flow $P_{contend}$ value
  - Flow next hop device ID

The second of the IEs holds information about the neighbors of the sending node. The beacons received from those neighbors have the IE described above, and the second IE packages all that information and passes it along. This ensures that all the information in the first IE of each node propagates to the 2-hop neighborhood of that node. All nodes increment their current Superframe ID to match the largest

Superframe ID received in any beacon. All flows outgoing from a given node are implicitly colored with the color advertised in that node's beacon.

Each node maintains two lists of flows: the *outgoing flow* list, which contains the flows that are outgoing from that node, and the *schedulable flow* list, which contains all flows in the 2-hop neighborhood of that node. The *outgoing flow* list is used to populate the first of the two IEs in each node's beacon, while the *schedulabe flow* list is used by the distributed scheduling algorithm to elect a flow winner for each slot of every superframe.

The firmware does not have enough CPU cycles to spare to perform the schedule computation, so it must be done in the driver. At the end of each beacon period, the firmware takes its updated list of schedulable flows and sends it via the Interrupt IN pipe to the driver. The Interrupt pipes are serviced with a period of 8ms, so this information is guaranteed to be delivered to the driver within 8ms of the beacon period being over. Once the driver receives the updated schedulable flows list, it runs the flow selection algorithm to elect a flow winner for each slot in the upcoming superframe. It also computes updated $P_{contend}$ values of each flow, and sends all this back to the modem via the Interrupt OUT pipe. This information will get to the modem before the next beacon period, so the updated flow information can be used to refresh the contents of the IEs of each node. The slot schedule is ready to be used in the upcoming superframe.

5.3.3 Data Transmission

The receive path of the modem is straight forward. When a packet arrives over the air, it gets immediately queued up on the Bulk IN pipe to be sent up to the driver. The complexity is in the transmit path. Every time a USB transfer completion event occurs for the Bulk OUT pipe, indicating that the driver has sent over a new packet of data, the firmware looks in the header of the packet to see which flow it belongs to and puts it in the wireless queue for that flow. Asynchronously, a timer fires at the beginning of each data slot and the firmware refers to the slot schedule for that superframe to find out

which of the outgoing flow, if any, will win the slot. If it find a matching flow, it sends the packet at the head of its queue. If there is no matching flow for this node, this means that a flow from some other node has won the slot and the node should listen for incoming transmission.

The memory constraints of the STX1201 present an interesting challenge here. To buffer enough data for a full superframe would require 480KB (2000 bytes per slot and 240 data slots per superframe). We have only around 1/10 of that, which means that the driver needs to get the data to the modem in the right order and it needs to try to get it there as close to the slot during which it will be sent out as possible. The driver uses the Interrupt packet received from the modem right after each beacon period as a reference point. Assuming a worst case delay of 8ms between the end of the beacon period and the reception of this packet, the driver estimates the first data slot of the next superframe to be in $duration_{superframe} - 8ms$.

5.4 Driver

A substantial portion of the implementation of TRANSFORMA lies in the driver. The advantage of implementing a portion of TRANSFORMA in the driver is that it has more computational and memory resources available to it than the modem. The disadvantage is that the driver's only connection to the modem is the USB link. The loosely bounded nature of the latency of this link combined with the fact that there are time sensitive operations the driver needs to perform during each superframe to keep the protocol operating make the implementation challenging.

The TRANSFORMA driver is built around the Linux *usbnet* driver, which is used by most USB network devices. The *usbnet* driver is designed to perform all the USB operations required to implement a USB network interface such as a USB Ethernet adapter or USB WiFi card. Normally, the per-device specifics are handled by a device-specific "mini-driver" which leverages the *usbnet* driver for the heavy lifting required to enumerate the device and keep data flowing through the Bulk IN and OUT

pipes. It was not possible to implement TRANSFORMA as a mini-driver because it requires more of the driver than typical devices and the outgoing data flow is somewhat more complex. We started by taking the entire *usbnet* driver, merging it together with one of the mini-drivers, and then making significant alterations to support the per-flow queues, traffic forecaster, and flow-selection computations that make up TRANSFORMA.

5.4.1 Handling Outgoing Data

When the Linux network layer has data to send that will be going out through the *tran0* network interface instantiated by our driver, the kernel calls the *.ndo_start_xmit* function of the driver with a pointer to the *sk_buff* that holds the IP packet. If this was a standard USB Ethernet device, the contents of this packet would be placed in an URB (USB request block) and the URB would then be queued for transmission through the Bulk OUT pipe to the USB device. In TRANSFORMA, several things need to happen before a packet can be sent over the air, so we must temporarily store the packet somewhere while the chain of events its arrival causes are completed.

TRANSFORMA sorts packets by flow, and each flow has a packet queue. The first thing that must happen when a packet arrives is to identify the flow this packet belongs to. TRANSFORMA identifies what flow a packet belongs to using the source IP address, destination IP address, and if the packet is UDP or TCP, the source and destination ports as well. These properties of the packet are used to look it up in a hash table of outgoing flows. If the flow's state structure is not found in the hash table, it is added. The flow structure contains an sk_buff queue used for queuing outgoing packets as well as a forecaster event queue. The event queue is used to queue events such as packet arrivals, departures, and, when a flow hasn't received packets for a while, flow deletions. These events are passed in batches to the user space forecaster. The forecaster uses these events to update its forecasts and state for each flow.

5.4.2 Interfacing To Forecaster

As mentioned earlier, TRANSFORMA's traffic forecaster is implemented as a user-space process. This means that it needs to a method of communicating with the driver. The method that was chosen is a *char* device interface. The driver instantiates a unique char device for each TRANSFORMA instance for which a device file is created in /dev. For example, the *tran0* TRANSFORMA interface has a matching *char* device file called *fcast_tran0*. A char device file is opened and accessed like any other file on the system using *open*, *read*, *write* file I/O system calls. In the case of TRANSFORMA, the driver ensures that only one process can open the file at a time.

When the user-space forecaster process is launched, it opens the *fcast_tranX* file and immediately executes a blocking read request. In the driver, the first thing the char device read operation handler does is wait for a flag to be set that indicates there is new data for the forecaster. When this flag is not set, the system scheduler puts the calling process (in this case the user-space forecaster) to sleep until that flag is set. The next time the driver sees a packet arrival, departure, or flow timeout, it will enqueue a new event and set the flag, causing the user-space app to get scheduled and process the new events. When the events have been processed, the updated forecasts for all active flows are given to to the drive via a write call to the char device. The event queue mechanism gives the system the flexibility to deal with higher load scenarios where multiple events can occur before the user-space process has a chance to read any of them out.

5.4.3 USB Communication

As mentioned in Section 5.3.3, due to the memory constraints of the modem the driver needs to send it data packets in the order in which they will be sent over the air. This order is often not going to be the order in which the packets were received from the network layer – this is the reason we can't simply package each packet arriving from the network layer into an URB and queue it to the Bulk OUT

pipe. What the driver does instead is wait until the last possible moment before a flow's slot, and then pull out as many packets out of the flow's queue as will fit in the slot and queue them for transmission.

To achieve this, there is a transmission timer in the driver that fires as close as possible to once per slot. In actuality, due to the standard time-tick resolution of the Linux kernel, this timer fires every 4ms (15.625 slots). This is acceptable because, to account for the latency of the Bulk pipe, the driver queues the data for transmission half a superframe in advance of the slot during which it will be sent over the air. Each time the timer fires, the driver goes through the list of schedules for upcoming superframes and searches for any slot that will be scheduled within the next 32ms. The arrival time of the Interrupt IN URB (Section 5.3.1) is used as a reference point to let the driver compute where in the superframe it currently is; it is known that this URB will be received within 8ms of the end of the beacon period in each superframe. The driver aggregates packets to generate payloads up to 2000 bytes in size. The modem does not unpack these payloads, but simply passes them on over the air to their next hop. The driver on the other end unpacks them before passing them up to the network layer.

At the start of every superframe the STX1201 modem exchanges beacons with other modems around it during the beacon period. Once the exchange is complete, the modem sends an Interrupt URB to the driver with an up-to-date *schedulable flow* list. The driver uses this information to run the competition probability and flow selection algorithms described in Section 4.3.4. The output of these algorithms is an up to date *outgoing flow* list and a schedule for the next superframe. The outgoing flow information is stored as part of the flow state that is kept for each flow and the schedule is added to the end of a schedule list. Both pieces of information are also placed inside of Interrupt URBs and sent back to the modem.

5.5 Forecaster Daemon

Thanks to the fact that the forecaster daemon runs in user-space and not kernel space, the same code that is used in the Qualnet simulations can be reused for the implementation. The forecaster algorithm is described in detail in Section 4.3.3. The daemon executes the following operations in an endless loop: read events from driver, process events, write forecasts to driver.

When the daemon does a read operation on the *fcast_tranX* device, it returns an event queue for each flow. The events can be one of: packet arrival, packet departure, flow deletion. Arrivals are passed to the forecaster's arrival handler, which updates the forecast for the corresponding flow. Packet departure events go to the departure handler, which does not affect the forecast, but is necessary to compute the arrival-to-departure delay and queue occupancy statistics. These statistics are periodically written to a file and provide useful information for debugging and tuning the forecaster.

5.6 The journey of an outgoing packet

To solidify the reader's understanding of how driver, firmware and forecaster daemon all work together in TRANSFORMA, this section will describe all the steps that happen in order to enable the first packet of a flow to be sent over the air. Figure 5.4 illustrates the steps that take place.

It all begins when a packet arrives at the driver from the network layer. Since this is the first packet of this flow, a new flow state instance is created for it. The packet is queued in the packet queue of this flow state instance and the arrival event for the packet goes into the forecaster event queue for the flow. As soon as the flag is set to indicate that there are events for the waiting forecaster daemon to process, the system scheduler schedules its process and it reads in the pending events. The forecaster creates a new flow state instance for this flow because it, too, didn't have one since this is the first packet and then updates the forecast. This being the first packet, the forecast is set to the maximum number of slots per second. This helps to compensate for the fact that the first few packets of the flow need to wait

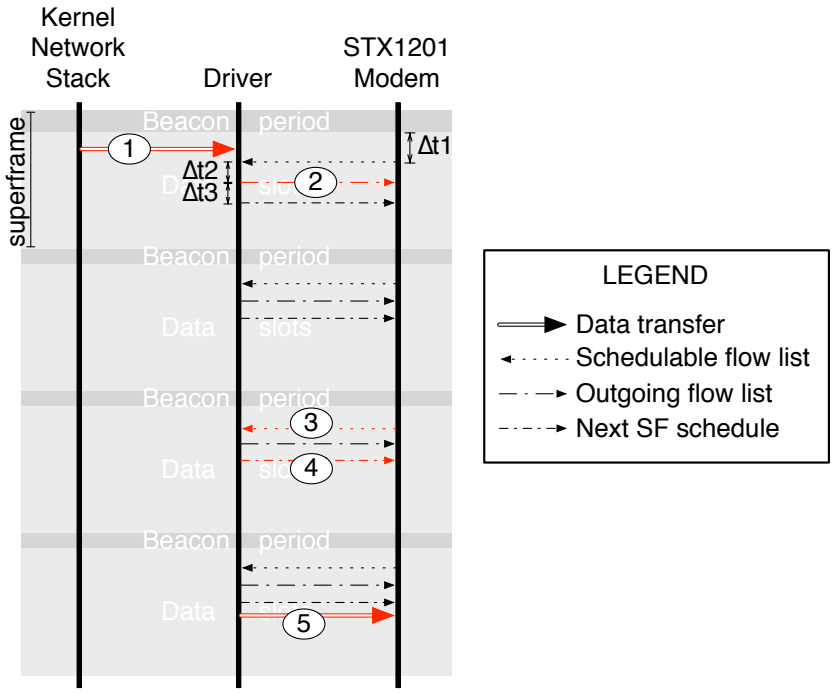


Figure 5.4: The journey of the first outgoing packet in a flow.

a little longer than the following packets. The new forecast is written back to the driver, which uses it to update the state instance of the flow.

Meanwhile, after each beacon period, the modem sends the driver the latest schedulable flow list. This time around, the schedulable flow list does not include the flow that was just created, so it cannot be included in the schedule for the next superframe yet. However, in response to the schedulable flow list, the driver sends its outgoing flow list, which does include the new flow. Note that due to the period of the Interrupt pipes being set to 8ms, the values of $\Delta t1$,$\Delta t2$,$\Delta t3$ are in the range (0ms,8ms).

In the next beacon period, the modem will advertise this new flow, and in the beacon period of the following superframe, the modem will hear its neighbors advertising the new flow. At that point, the new flow will become part of the schedulable flow list which the modem passes up to the driver. The driver will then include the new flow in the schedule for the subsequent superframe, during which the packet will be sent down to the modem and then in turn over the air.

It is clear that some delay is incurred by the first packets until the flow is established. Following that, the delay is much smaller and is a function of how many slots per second the flow is allocated and whether that amount is sufficient for all its packets or not.

5.7 Experiments

5.7.1 Heterogeneous Flows

The first experiment we carried out with the TRANSFORMA implementation emulates one of the experiments in Section 4.4: the heterogeneous flow experiment. In this experiment we systematically add flows of varying data rates one by one and observe the performance seen by each flow as the load increases. The iperf tool [89] for bandwidth performance testing was used to generate the CBR flows for this experiment. The achievable data rate, given we can fit 2000 bytes per slot, 240 slots per superframe, and 64ms superframe duration, is 60Mbps. The rates of the heterogeneous flows in this experiment are

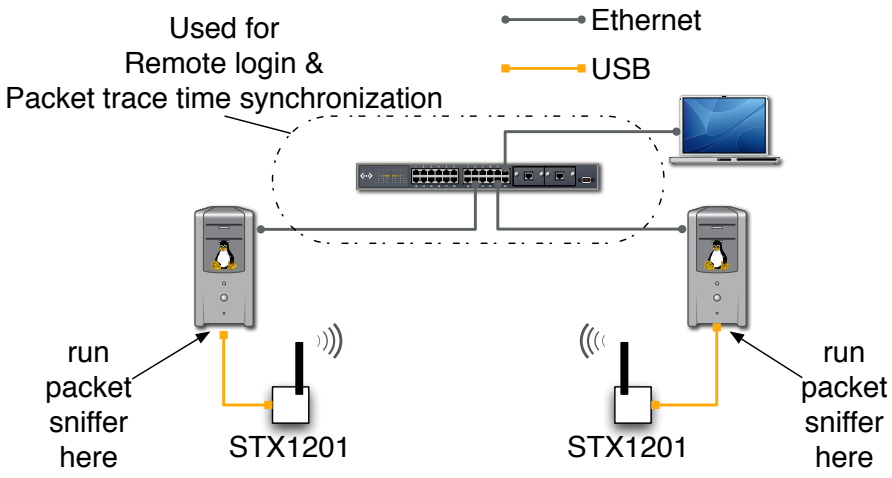


Figure 5.5: Heterogeneous flow experiment network setup

chosen so that the total load approaches this maximum when all the flows are running. We configured iperf to use a packet size of 950 bytes (not including headers) so that 2 such packets will fit in one data slot. Figure 5.5 illustrates the experiment setup. Each node is connected to two subnets, one is for the TRANSFORMA link, and the other is for Network Time Protocol (NTP) time synchronization and remote access into the nodes. NTP synchronizes node clocks to within less than 1 ms, allowing us to correlate packet departure times on one node with the packet arrival times on the other and compute end to end delay.

In Figures 5.6,5.7,5.8 and 5.9, each X-axis value is generated by running that number of flows, and then taking the average delay, delivery ratio or good put. During every such run, flows are started one by one in order from highest data rate to lowest data rate, staggered by 1s. Each flow runs for 60s. In addition, we ran 10 iterations of all the runs, and each point on the plot represents an average of these 10 iterations.

All of our experiment results are extracted from packet traces taken from the *tran0* network interface on each node. End to end delay, delivery ratio, and goodput are computed by combining the packet traces from the sending and receiving interfaces. This would not be possible without the time

synchronization afforded by the NTP protocol, which keeps the nodes system clocks less than 1 ms and as little as 0.1ms apart.

Figure 5.6 shows the delay plots of a subset of the total flows (values including standard deviations can be found in Table 5.1). The trend we expect to see from our simulation experience is that the flow with the largest throughput should experience the shortest delay. This is due to the fact that the more slots a flow can use, the smaller the average inter-slot time becomes. In addition, as the number of flows increases, the end-to-end delay for each flow will increase because more flows are now sharing the same number of slots so the inter-slot time for each flow decreases. The second trend is apparent in this figure, but the first is missing. The reason turns out to be the fact that the delay a flow's packets experience is not constant over the lifetime of the flow. Specifically, when the flow is being established, the packets that arrive during the first few superframes must wait approximately 4 superframe durations before they are sent out. If the flows queue builds up at this time, subsequent packets may also experience some additional delay until the queue empties. Available bandwidth is divided amount the active flows according to their forecast ratios, so as long as the offered load is less than the sustainable load, the queues are guaranteed to empty. Figure 5.7 illustrates what the average delays look like if we disregard the packets arriving during the first 5s of each flow. In this figure the expected trend is visible: the flows with higher throughputs are allocated more slots and, as a consequence, experience lower delays.

Figure 5.8 shows that the delivery ratio of the TRANSFORMA link is approximately 100% – a characteristic that comes from the schedule based nature of the protocol. Figure 5.9 shows the goodput seen by each flow. The values are about 6% higher than the application level throughput because this goodput is measured at the network layer and includes IP and UDP header overheads.

5.7.2 VoIP With Background Traffic

The second experiment we carried out showcases a real VoIP application operating on top of the TRANSFORMA link. The setup is similar to the one in the previous experiment, but the routing

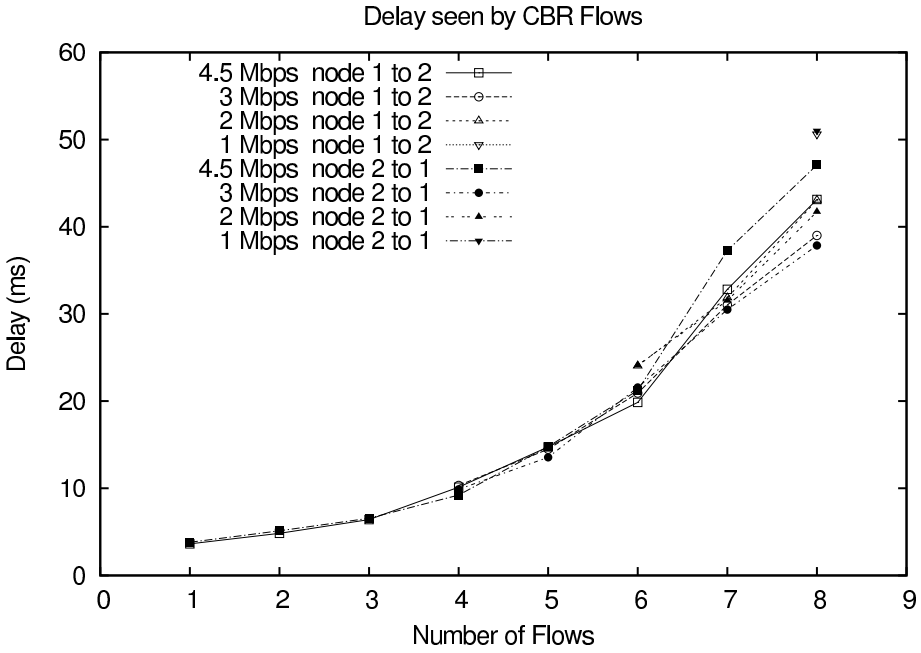


Figure 5.6: Average delay of heterogeneous flows in each direction going over a TRANSFORMA link.

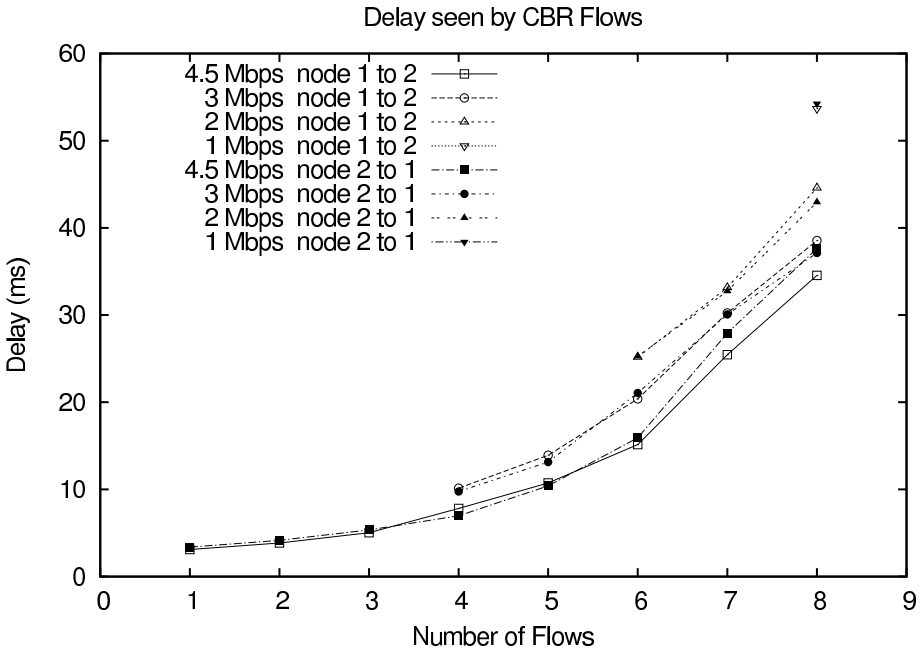


Figure 5.7: Average delay excluding first 5s of heterogeneous flows in each direction going over a TRANSFORMA link.

Table 5.1: Delay values of Figure 5.6

	Delays of flows from node 1 to node 2 Units are milliseconds							
Flow Count	1 Mbps	1.5 Mbps	2 Mbps	2.5 Mbps	3 Mbps	3.5 Mbps	4 Mbps	4.5 Mbps
1								3.6±0.4
2							4.4±0.3	4.8±0.3
3						6.1±0.3	6.3±0.3	6.4±0.8
4					10.3±0.7	10.1±0.7	9.3±0.6	10.1±0.8
5				15.1±1.1	14.5±1.1	14.2±1.2	14.6±1.2	14.7±2.2
6			24.0±1.7	22.7±1.9	20.9±1.2	18.6±1.6	18.3±2.0	19.9±2.8
7		36.3±1.0	31.9±1.5	37.7±4.6	31.0±2.2	28.5±2.5	28.4±1.5	32.8±3.5
8	50.6±3.7	50.1±4.6	43.1±4.7	41.3±3.9	39.0±6.6	35.7±4.3	37.2±3.4	43.1±3.1

	Delays of flows from node 2 to node 1 Units are milliseconds							
Flow Count	1 Mbps	1.5 Mbps	2 Mbps	2.5 Mbps	3 Mbps	3.5 Mbps	4 Mbps	4.5 Mbps
1								3.8±0.4
2							4.7±0.3	5.1±0.2
3						6.5±0.4	6.4±0.3	6.6±0.4
4					9.9±0.5	9.1±0.5	8.4±0.6	9.2±0.6
5				15.1±0.7	13.5±0.6	13.4±0.9	12.9±1.0	14.8±2.4
6			24.1±1.5	21.9±1.5	21.6±1.9	19.6±1.3	18.1±1.1	21.2±1.9
7		36.4±1.1	31.5±1.2	37.9±1.9	30.5±2.2	31.0±2.5	31.5±3.8	37.3±3.8
8	51.0±1.3	51.6±3.4	41.7±4.7	37.3±2.5	37.9±3.7	37.1±2.6	40.6±7.6	47.1±9.7

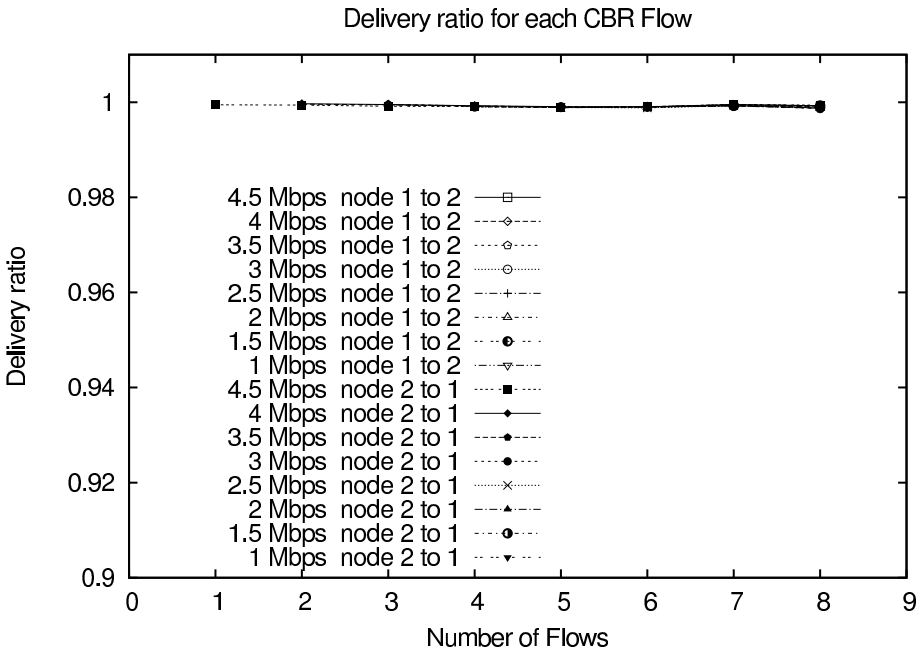


Figure 5.8: Delivery ratio of heterogeneous flows in each direction going over a TRANSFORMA link.

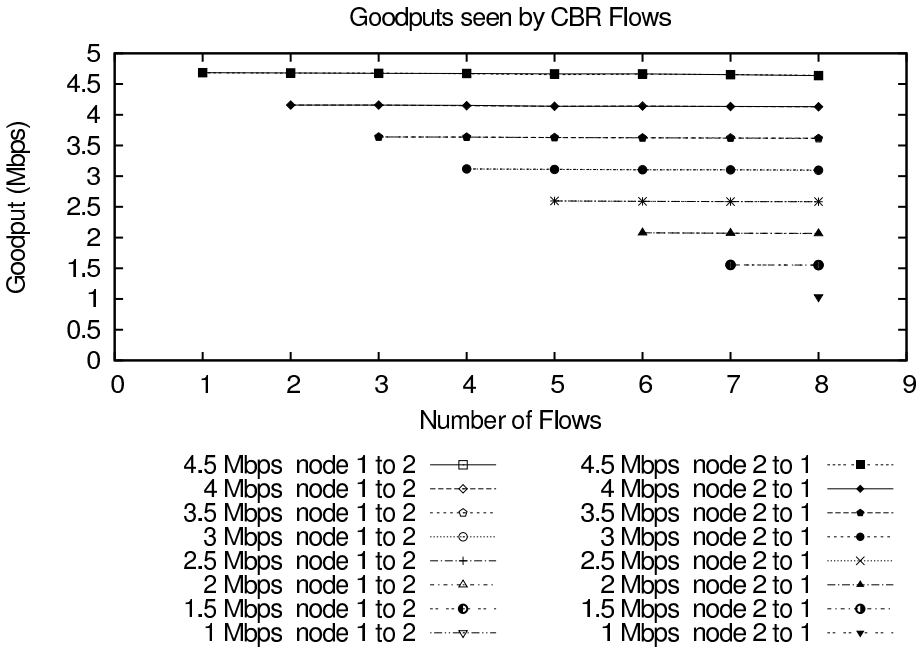


Figure 5.9: Goodput of heterogeneous flows in each direction going over a TRANSFORMA link.

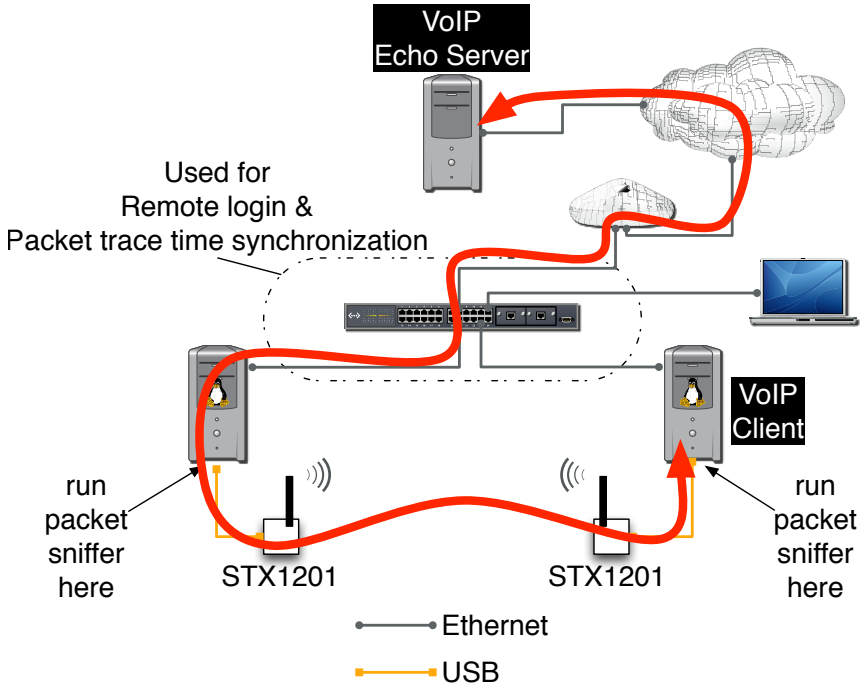


Figure 5.10: VoIP experiment network setup

tables on the nodes and the internet router were adjusted to provide internet connectivity to one of the nodes only through the TRANSFORMA link. In this experiment, a call is made using the Ekiga Linux Softphone application from the VoIP client node in Figure 5.10 to an echo server. Everything heard by the microphone is transmitted over the TRANSFORMA link, through the internet, to the server on the other end. The server then echoes it back and it is heard a split second later over the earpiece. This results in two VoIP flows over the TRANSFORMA link – one originating from behind the TRANSFORMA link, and one originating from a remote server and routed back over the TRANSFORMA link. While this call is in progress, we increase the traffic over the TRANSFORMA link by adding 6 4Mbps flows one at a time at 10s intervals.

For this experiment, the delay and goodput of each active flow is plotted over time – instead of being averaged over the duration of the flow they are averaged over a shorter period of 0.5s so that we

can see the change in these quantities as the load changes. Figure 5.11 has a plot of the delay for the 6 CBR flows on top and a separate one for the VoIP flows on the bottom. These were separated to make the plots a little more readable. Each flow starts out with a higher delay due to the necessary flow setup. Once the flow enters the schedules of the 2 hop neighborhood, the packets that were queued are quickly sent out and the forecast settles. As flows are added, the forecasts of other flows are not affected, but the number of slots that are allocated to each flow get reduced, though they are still allocated in proportion to the forecasts.

The data rate of the background flows is much higher than that of the VoIP flows as can be seen in Figure 5.12. As a result, the delay of the VoIP flows should be higher, although it is still at a respectable level by VoIP standards. The "in" VoIP flow does not get the same forecast as the "out" flow because it has been slightly shaped by the Internet link that it traverses. When faced with varying packet inter-arrival times, the TRANSFORMA forecaster errs on the side of caution and puts greatest value on the smallest inter-arrival time when forecasting the data rate of the flow. This means that if an otherwise periodic flow, such as VoIP has it's inter-arrival times altered such that some packets arrive closer together, the resulting forecast will be higher than it would otherwise.

Although delivery ratio plots have been omitted, the call audio quality was good and the delay between words was not perceived to be higher than that of the same VoIP call to the echo server done over an Ethernet link.

5.8 Conclusion

We were able to implement TRANSFORMA using the Starix STX1201 with custom firmware and a matching Linux network device driver. The end result was a network link that faithfully executes the TRANSFORMA wireless MAC protocol and can be used to study its behavior away from the confines of a simulated setting. We have presented two experiments here that illustrate some of the performance

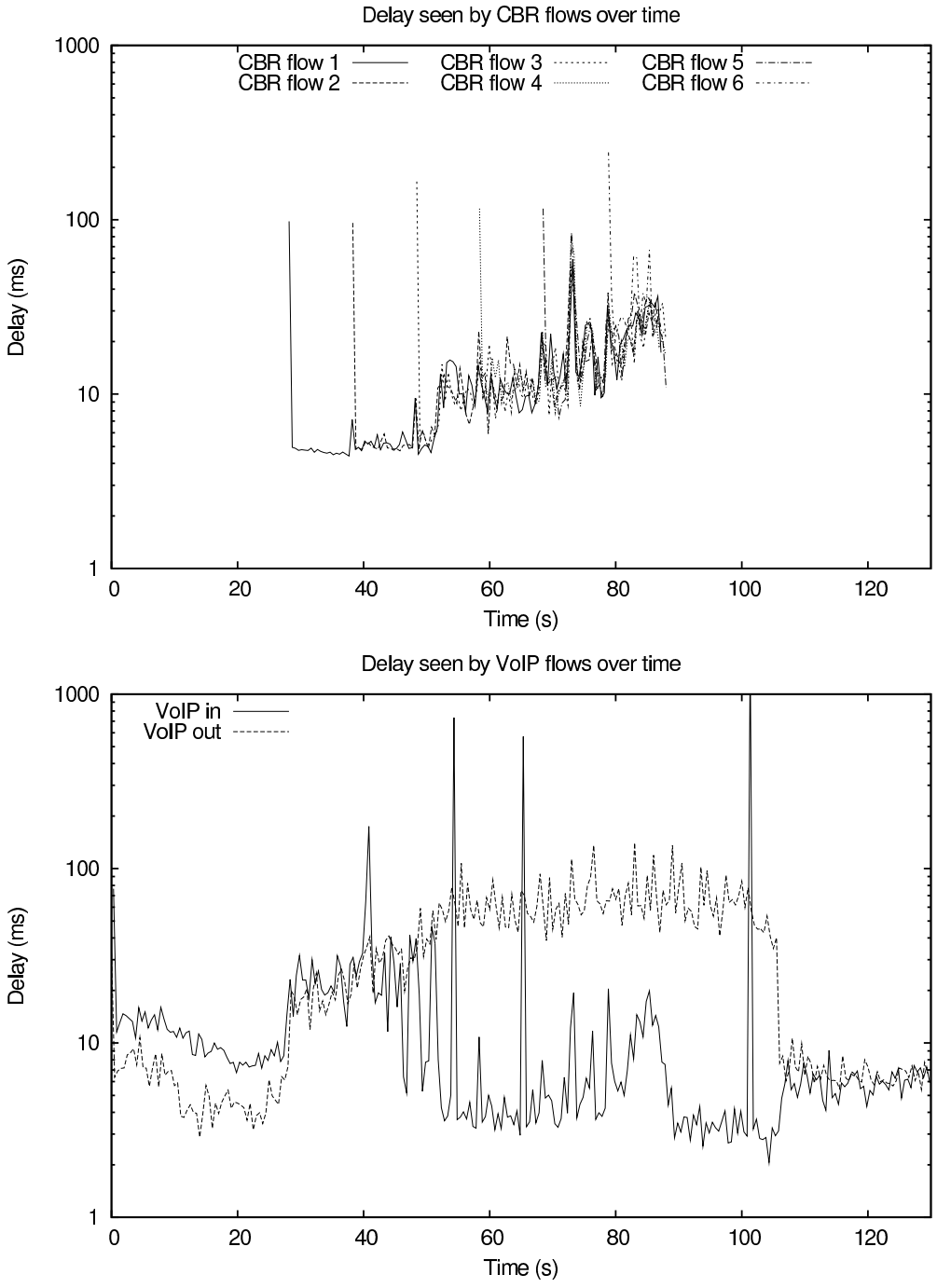


Figure 5.11: Average delay across TRANSFORMA link seen by flows over time

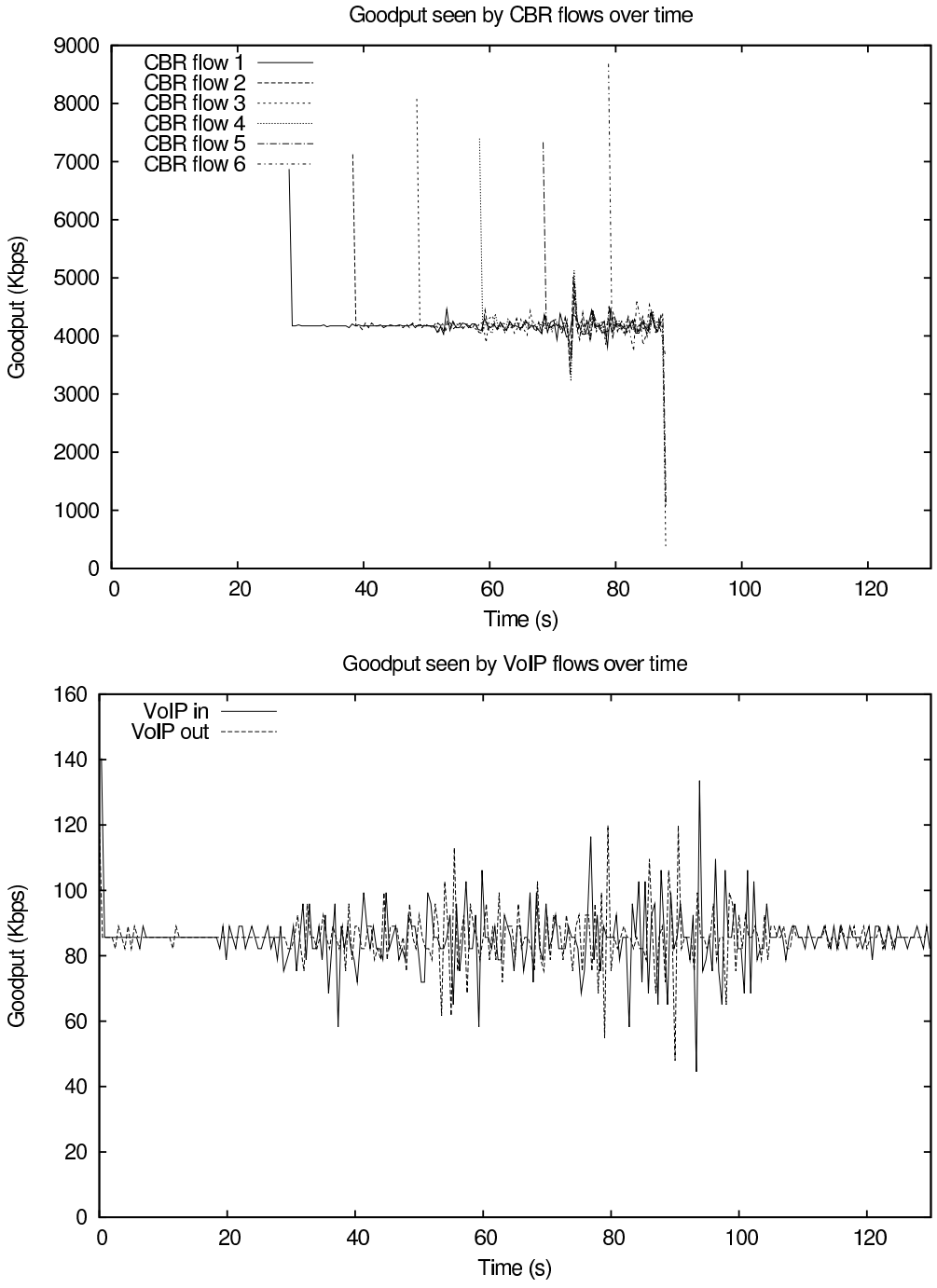


Figure 5.12: Average goodput across TRANSFORMA link seen by flows over time

capabilities and behaviors of the protocol in various settings. The experiment results validate those we have seen in our simulations. Although there are aspects of the implementation that can be improved upon, it is fully functional and demonstrates that the TRANSFORMA MAC works not only on paper but also in practice.

Chapter 6

Conclusion and Future Work

In the early work presented in Chapter 2 we found that the expected benefit of traffic forecasting for a schedule-based medium access approach is a reduction of the delay incurred by the protocol under lower load scenarios. Given that wireless networks often operate at less than full load, this benefit is very useful. While carrying out this work, an early version of the forecaster that would later be used in TRANSFORMA was developed and shown to work effectively at forecasting the packet inter-arrival time most likely to meet a network flow's latency requirements.

Next, in the work presented in Chapter 3 we studied the use of entropy to quantify and compare per-application traffic complexity. We presented results of using two entropy estimation approaches – our own PPTEn estimator and the SampEn estimator – and showed that the output of our PPTEn entropy estimator provides more information on the application behavior and can more readily be used to compare per-flow network traffic complexities.

The PPTEn estimates corresponded almost exactly to the network traffic complexity ordering we came up with based on visual analysis of the network traffic from Skype, GoogleTalk, iChat, Hulu.com, ABC.com, and Netflix.com. In addition, the PPTEn estimates over a range τ highlight many application characteristics, some of which are even too subtle for visual observation. We refer to the en-

tropy estimates as "entropy fingerprints" because of how closely related to the flow characteristics they are.

Following this, we designed a MAC protocol that leverages per-flow traffic forecasts to schedule data slots in a timely manner. Chapter 4 presents TRANSFORMA, a collision-free, scheduled-based medium access control protocol that employs a novel approach to medium access based on traffic forecasting. TRANSFORMAs traffic forecaster identifies patterns in application flows and uses this information to schedule access to the medium most effectively. By doing so, TRANSFORMA tries to anticipate the workload at each node and sets transmission schedules accordingly. This is in contrast to traditional scheduled access protocols (e.g., DYNAMMA [57], which set schedules reactively and thus incur considerably higher delays. We showed through simulations that TRANSFORMA is able to identify the traffic patterns of various kinds of flows and use that information to schedule them, assuring each flow a packet delay on the order of its packet inter-arrival time. Our results also showed that TRANSFORMA is able to schedule real-time flows alongside background traffic with less adverse effects on the real time flows delay than 802.11 and DYNAMMA. Moving forward, our future work plans include: performing live experiments on a testbed as a way to cross-validate our simulation results and developing an analytical model to derive performance bounds for TRANSFORMA and as a way to validate our simulations.

In Chapter 5 we describe how we implemented TRANSFORMA using the Starix STX1201 with custom firmware and a matching Linux network device driver. The end result was a network link that faithfully executes the TRANSFORMA wireless MAC protocol and can be used to study its behavior away from the confines of a simulated setting. The two experiments presented illustrate some of the performance capabilities and behaviors of the protocol in various settings. The experiment results validate those we have seen in our simulations: delay performance of TRANSFORMA is very good for a schedule-based MAC and, thanks to its forecaster, TRANSFORMA divides the available bandwidth among active flows in proportion to their forecast bandwidth requirements. Our implementation and the results from the experiments we ran on it demonstrate that TRANSFORMA is implementable on real

hardware.

6.1 Future Work

There are a few key areas in this work that can be explored further. The first of these is the traffic forecasting component of the protocol. There exist many other forecasters that may be better suited to the task than the one we have chosen. Furthermore, it is possible to use a set of forecasters and choose the most appropriate one for each flow, possibly using a classification algorithm based on the entropy fingerprints we discovered.

Currently TRANSFORMA's performance in multi-hop scenarios is inversely proportional to the number of hops because each hop constitutes a separate flow and the state for each is set up separately as packets make their way from one hop to the next. If TRANSFORMA is made routing aware, it can set up the flow states for each hop during the route discovery phase to reduce the delay incurred by this process.

And lastly, we believe that TRANSFORMA's concepts of traffic awareness can be brought to the IEEE 802.11 arena by leveraging the Enhanced Distributed Channel Access (EDCA) function's four traffic classes and using the forecaster to dynamically tune their parameters.

Bibliography

[1] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, no. 27, pp. 379–423, 1948.

[2] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves, "Dynamma: A dynamic multi-channel medium access framework for wireless ad hoc networks," *Mobile Adhoc and Sensor Systems*, Jan 2007.

[3] "High rate ultra wideband phy and mac standard," ECMA Internal Standard ECMA-368, 2007.

[4] N. Abramson, "The ALOHA System-Another Alternative for Computer Communications," *Fall Joint Comput. Conf., AFIPS Conf. Proc.*, vol. 37, pp. 281–285, 1970.

[5] L. Kleinrock and F. Tobagi, "Packet switching in radio channels: Part i–carrier sense multiple-access modes and their throughput-delay characteristics," *Communications, IEEE Transactions on [legacy, pre - 1988]*, vol. 23, no. 12, pp. 1400–1416, Dec 1975.

[6] P. Karn, "MACA-a new channel access method for packet radio," *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, vol. 140, 1990.

[7] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "Macaw: a media access protocol for wireless lan's," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 4, pp. 212–225, 1994.

[8] B. Crow, I. Widjaja, L. Kim, and P. Sakai, "Ieee 802.11 wireless local area networks," *Communications Magazine, IEEE*, vol. 35, no. 9, pp. 116–126, Sep 1997.

[9] Z. Tang and J. Garcia-Luna-Aceves, "A protocol for topology-dependent transmission scheduling in wireless networks," *Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE*, pp. 1333–1337 vol.3, 1999.

[10] L. Bao and J. J. Garcia-Luna-Aceves, "A new approach to channel access scheduling for ad hoc networks," in *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2001, pp. 210–221.

[11] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves, "Energy-efficient collision-free medium access control for wireless sensor networks," *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, Nov 2003.

[12] V. Rajendran, J. Garcia-Luna-Aceves, and K. Obraczka, "Energy-efficient, application-aware medium access for sensor networks," *IEEE Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, Jan 2005.

[13] F. Tobagi and L. Kleinrock, "Packet switching in radio channels: Part ii–the hidden terminal problem in carrier sense multiple-access and the busy-tone solution," *Communications, IEEE Transactions on [legacy, pre - 1988]*, vol. 23, no. 12, pp. 1417–1433, Dec 1975.

[14] V. Petkov and K. Obraczka, "The case for using traffic forecasting in schedule-based channel access," in *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, jan. 2011, pp. 208 –212.

[15] L. Bao and J. Garcia-Luna-Aceves, "Hybrid channel access scheduling in ad hoc networks," *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*, pp. 46–57, Nov. 2002.

[16] Y. Freund, D. Haussler, D. Helmbold, and R. Schapire, "How to use expert advice," *Journal of the ACM*, Jan 1997. [Online]. Available: http://mercurio.srv.dsi.unimi.it/~cesabian/Pubblicazioni/jacm-97a.pdf

[17] M. Herbster and M. Warmuth, "Tracking the Best Expert," *Machine Learning*, vol. 32, no. 2, pp. 151–178, 1998.

[18] A. Adas, "Using adaptive linear prediction to support real-time vbr video under rcbr network service model," *Networking, IEEE/ACM Transactions on*, vol. 6, no. 5, pp. 635–644, Oct 1998.

[19] Y. Afek, M. Cohen, E. Haalman, and Y. Mansour, "Dynamic bandwidth allocation policies," *INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, vol. 2, pp. 880–887 vol.2, Mar 1996.

[20] P. Koutsakis, M. Paterakis, and S. Psychis, "Call admission control and traffic policing mechanisms for the wireless transmission of layered videoconference traffic from mpeg-4 and h.263 video coders," vol. 5, sep. 2002, pp. 2155 – 2159 vol.5.

[21] S. Chatziperis, P. Koutsakis, and M. Paterakis, "On achieving accurate call admission control for videoconference traffic transmission over wireless cellular networks," mar. 2007, pp. 3221 –3226.

[22] H. Liu and Y. Zhao, "Adaptive edca algorithm using video prediction for multimedia ieee 802.11e wlan," jul. 2006, pp. 10 –10.

[23] S. Shah, K. Chen, and K. Nahrstedt, "Dynamic bandwidth management for single-hop ad hoc wireless networks," *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on*, pp. 195–203, March 2003.

[24] ECMA International, "High rate ultra wideband PHY and MAC standard," ECMA International Standard, www.ecma-international.org, Tech. Rep., Dec 2005.

[25] G. Combs, "Wireshark network protocol analyzer," http://www.wireshark.org/. [Online]. Available: http://www.wireshark.org/

[26] L. Roberts, "A radical new router," *IEEE Spectrum*, July 2009.

[27] J. Riihijarvi, M. Wellens, and P. Mahonen, "Measuring complexity and predictability in networks with multiscale entropy analysis," in *INFOCOM 2009, IEEE*, April 2009, pp. 1107 –1115.

[28] Y. Gao, I. Kontoyiannis, and E. Bienenstock, "From the entropy to the statistical structure of spike trains," in *IEEE International Symposium on Information Theory*, July 2006, pp. 645–649.

[29] J. S. Richman and J. R. Moorman, "Physiological time-series analysis using approximate entropy and sample entropy," *Am J Physiol Heart Circ Physiol*, vol. 278, no. 6, pp. H2039–2049, 2000. [Online]. Available: http://ajpheart.physiology.org/cgi/content/abstract/278/6/H2039

[30] Sandvine Incorporated, "Global internet phenomena report," [Online] Available at: http://www.sandvine.com/news/global_broadband_trends.asp, May 2011.

[31] W. contributors, "YouTube Wikipedia entry," [Online] Available at: http://en.wikipedia.org/wiki/Youtube, October 2010.

[32] Hulu, "Hulu media faq," [Online] Available at: http://www.hulu.com/about/media_faq. [Online]. Available: http://www.hulu.com/about/media_faq

[33] N. Hunt, "Netflix encoding for streaming," [Online] Available at: http://blog.netflix.com/2008/11/encoding-for-streaming.html, November 2008.

[34] D. Bonfiglio, M. Mellia, M. Meo, and D. Rossi, "Detailed analysis of skype traffic," *Multimedia, IEEE Transactions on*, vol. 11, no. 1, pp. 117 –127, jan. 2009.

[35] Apple, "iChat in OS X Leopard," [Online] Available at: http://www.apple.com/asia/macosx/leopard/features/ichat.html, October 2010.

[36] Google, "GoogleTalk developer info," [Online] Available at: http://code.google.com/apis/talk/open_communications.html, October 2010.

[37] Apple, "iChat Wikipedia entry," [Online] Available at: http://en.wikipedia.org/wiki/Ichat, October 2010.

[38] L. Berkeley, "National laboratory network research. tcpdump: the protocol packet capture and dumper program. http://www.tcpdump.org," in *the Protocol Packet Capture and Dumper Program," 2003*, 2001, p. 164.

[39] M. Perényi and S. Molnár, "Enhanced skype traffic identification," in *ValueTools '07: Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007, pp. 1–9.

[40] D. Rossi, S. Valenti, P. Veglia, D. Bonfiglio, M. Mellia, and M. Meo, "Pictures from the skype," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 2, pp. 83–86, 2008.

[41] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of ethernet traffic," in *SIGCOMM '93: Conference proceedings on Communications architectures, protocols and applications*. New York, NY, USA: ACM, 1993, pp. 183–193.

[42] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic evidence and possible causes," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 835–846, 1996.

[43] J. Beran, R. Sherman, M. Taqqu, and W. Willinger, "Long-range dependence in variable-bit-rate video traffic," *Communications, IEEE Transactions on*, vol. 43, no. 234, pp. 1566–1579, Feb/Mar/Apr 1995.

[44] V. Paxson and S. Floyd, "Wide area traffic: the failure of poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, Jun 1995.

[45] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, "Self-similarity through high-variability: statistical analysis of ethernet lan traffic at the source level," *IEEE/ACM Transactions on Networking*, vol. 5, no. 1, pp. 71–86, 1997.

[46] K. Park, G. Kim, and M. Crovella, "On the relationship between file sizes, transport protocols, and self-similar network traffic," in *IEEE International Conference On Network Protocols*, Oct-1 Nov 1996, pp. 171–180.

[47] T. Karagiannis, M. Faloutsos, and M. Molle, "A user-friendly self-similarity analysis tool," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 81–93, 2003.

[48] C. Walsworth, E. Aben, K. Claffy, and D. Andersen, "The caida anonymized 2009 internet traces - (jan 15)," http://www.caida.org/data/passive/passive_2009_dataset.xml, January 2009.

[49] T. M. Cover and J. A. Thomas, *Elements of information theory*. New York, NY, USA: Wiley-Interscience, 1991. [Online]. Available: http://portal.acm.org/citation.cfm?id=129837

[50] F. Willems, Y. Shtarkov, and T. Tjalkens, "The context-tree weighting method: basic properties," *Information Theory, IEEE Transactions on*, vol. 41, no. 3, pp. 653–664, May 1995.

[51] G. Basharin, "On a statistical estimate for the entropy of a sequence of independent random variables," *Theory of Probability and its Applications*, vol. 4, p. 333, 1959.

[52] V. Vu, B. Yu, and R. Kass, "Coverage-adjusted entropy estimation," *Statistics in medicine*, vol. 26, no. 21, pp. 4039–4060, 2007.

[53] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to ddos attack detection and response," in *DARPA Information Survivability Conference and Exposition - Volume I*, vol. 1, April 2003, pp. 303–314 vol.1.

[54] K. Xu, Z.-L. Zhang, and S. Bhattacharyya, "Profiling internet backbone traffic: behavior models and applications," in *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2005, pp. 169–180.

[55] A. Wagner and B. Plattner, "Entropy based worm and anomaly detection in fast ip networks," in *WETICE '05: Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 172–177.

[56] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2005, pp. 217–228.

[57] A. Lall, V. Sekar, M. Ogihara, J. Xu, and H. Zhang, "Data streaming algorithms for estimating entropy of network traffic," in *SIGMETRICS '06/Performance '06: Proceedings of the joint international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 2006, pp. 145–156.

[58] A. Sang and S. Li, "A predictability analysis of network traffic," in *IEEE INFOCOM*, vol. 1, 2000, pp. 342–351 vol.1.

[59] V. Petkov and K. Obraczka, "Collision-free medium access based on traffic forecasting," in *Proceedings of the 13th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, June 2012.

[60] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2002, pp. 1567 – 1576 vol.3.

[61] L. van Hoesel, T. Nieberg, H. Kip, and P. Havinga, "Advantages of a tdma based, energy-efficient, self-organizing mac protocol for wsns," in *Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th*, vol. 3, 17-19 2004, pp. 1598 – 1602 Vol.3.

[62] L. Shi and A. Fapojuwo, "Tdma scheduling with optimized energy efficiency and minimum delay in clustered wireless sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 9, no. 7, pp. 927 –940, july 2010.

[63] L. Bing, Z. Lin, and Z. Huimin, "An adaptive schedule medium access control for wireless sensor networks," in *Networking, 2007. ICN '07. Sixth International Conference on*, 22-28 2007, pp. 12 –12.

[64] M. Sekine, S. Takeuchi, and K. Sezaki, "An energy-efficient mac protocol with lightweight and adaptive scheduling for wireless sensor networks," in *Radio and Wireless Symposium, 2007 IEEE*, 9-11 2007, pp. 161 –164.

[65] Z. Karakehayov and N. Andersen, "Energy-efficient medium access for data intensive wireless sensor networks," in *Mobile Data Management, 2007 International Conference on*, 1-1 2007, pp. 361 –365.

[66] Y. Liu, I. Elhanany, and H. Qi, "An energy-efficient qos-aware media access control protocol for wireless sensor networks," in *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, 7-7 2005, pp. 3 pp. –191.

[67] J. Cai, K.-H. Liu, X. Shen, J. Mark, and T. Todd, "Power allocation and scheduling for mac layer design in uwb networks," in *Quality of Service in Heterogeneous Wired/Wireless Networks, 2005. Second International Conference on*, 24-24 2005, pp. 8 pp. –2.

[68] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves, "Energy-efficient collision-free medium access control for wireless sensor networks," *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, Nov 2003.

[69] V. Rajendran, J. Garcia-Luna-Aceves, and K. Obraczka, "Energy-efficient, application-aware medium access for sensor networks," *IEEE Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, Jan 2005.

[70] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves, "Dynamma: A dynamic multi-channel medium access framework for wireless ad hoc networks," *Mobile Adhoc and Sensor Systems*, Jan 2007.

[71] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband internet traffic," in *IMC '09: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. New York, NY, USA: ACM, 2009, pp. 90–102.

[72] R. Pries, F. Wamser, D. Staehle, K. Heck, and P. Tran-Gia, "On traffic characteristics of a broadband wireless internet access," in *Next Generation Internet Networks, 2009. NGI '09*, 1-3 2009, pp. 1 –7.

[73] A. Sharma and E. M. Belding, "A case for application aware channel access in wireless networks," in *HotMobile '09: Proceedings of the 10th workshop on Mobile Computing Systems and Applications*. New York, NY, USA: ACM, 2009, pp. 1–6.

[74] S. Singh, P. Acharya, U. Madhow, and E. Belding-Royer, "Sticky csma/ca: Implicit synchronization and real-time qos in mesh networks," *Ad Hoc Networks*, Jan 2007. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1570870507000121

[75] P. Djukic and P. Mohapatra, "Soft-tdmac: A software tdma-based mac over commodity 802.11 hardware," in *INFOCOM 2009, IEEE*, 19-25 2009, pp. 1836 –1844.

[76] A. Sharma, M. Tiwari, and H. Zheng, "Madmac: Building a reconfiguration radio testbed using commodity 802.11 hardware," in *Networking Technologies for Software Defined Radio Networks, 2006. SDR '06.1st IEEE Workshop on*, 25-25 2006, pp. 78 –83.

[77] A. Sharma and E. M. Belding, "Freemac: framework for multi-channel mac development on 802.11 hardware," in *PRESTO '08: Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*. New York, NY, USA: ACM, 2008, pp. 69–74.

[78] A. Rao and I. Stoica, "An overlay mac layer for 802.11 networks," in *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*. New York, NY, USA: ACM, 2005, pp. 135–148.

[79] M. Herbster and M. Warmuth, "Tracking the best expert," in *Machine Learning*. Morgan Kaufmann, 1995, pp. 286–294.

[80] N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," in *Foundations of Computer Science, 1987., 28th Annual Symposium on*, oct. 1987, pp. 68 –77.

[81] N. Littlestone and M. Warmuth, "The weighted majority algorithm," in *Foundations of Computer Science, 1989., 30th Annual Symposium on*, oct-1 nov 1989, pp. 256 –261.

[82] Y. FREUND, D. HAUSSLER, D. HELMBOLD, R. SCHAPIRE, and et al., "How to use expert advice," *Journal of the ACM*, Jan 1997. [Online]. Available: http://mercurio.srv.dsi.unimi.it/~cesabian/Pubblicazioni/jacm-97a.pdf

[83] D. Helmbold, D. Long, T. Sconyers, and B. Sherrod, "Adaptive disk spin-down for mobile computers," *Mobile Networks and Applications*, vol. 5, p. 2000, 1998.

[84] Tcpdump/libpcap. [Online]. Available: http://www.tcpdump.org/

[85] Scalable networks. [Online]. Available: http://www.scalable-networks.com

[86] B. Crow, I. Widjaja, L. Kim, and P. Sakai, "Ieee 802.11 wireless local area networks," *Communications Magazine, IEEE*, vol. 35, no. 9, pp. 116 –126, sep 1997.

[87] Starix technology. [Online]. Available: http://www.starixtech.com

[88] Usb org. [Online]. Available: http://www.usb.org

[89] Iperf. [Online]. Available: http://iperf.sourceforge.net/