2009-05-26

Towards a Framework For Resource Allocation in Networks

Maththondage Chamara Sisirawansha Ranasingha
University of Miami, m.ranasingha@umiami.edu

UNIVERSITY OF MIAMI

TOWARDS A FRAMEWORK FOR RESOURCE ALLOCATION IN NETWORKS

By

Maththondage Chamara S. Ranasingha

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Coral Gables, Florida

June 2009

UNIVERSITY OF MIAMI

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

TOWARDS A FRAMEWORK FOR RESOURCE ALLOCATION IN NETWORKS

Maththondage Chamara S. Ranasingha

Approved:

Kamal Premaratne, Ph.D. Professor of Electrical and Computer Engineering	Terri A. Scandura, Ph.D. Dean of the Graduate School
Manohar N. Murthi, Ph.D. Associate Professor of Electrical and Computer Engineering	James W. Modestino, Ph.D. Professor of Electrical and Computer Engineering
Xiaodong Cai, Ph.D. Assistant Professor of Electrical and Computer Engineering	Subramanian Ramakrishnan, Ph.D. Associate Professor of Mathematics

RANASINGHA, MATHTHONDAGE CHAMARA S.

(Ph.D., Electrical and Computer Engineering)

Towards a Framework For Resource Allocation in Networks

(June 2009)

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Professor Kamal Premaratne.
No. of pages in text. (157)

Network resources (such as bandwidth on a link) are not unlimited, and must be shared by all networked applications in some manner of fairness. This calls for the development and implementation of effective strategies that enable optimal utilization of these scarce network resources among the various applications that share the network. Although several rate controllers have been proposed in the literature to address the issue of optimal rate allocation, they do not appear to capture other factors that are of critical concern. For example, consider a battlefield data fusion application where a fusion center desires to allocate more bandwidth to incoming flows that are perceived to be more accurate and important. For these applications, network users should consider transmission rates of other users in the process of rate allocation. Hence, a rate controller should consider application specific rate coordination directives given by the underlying application.

The work reported herein addresses this issue of how a rate controller may establish and maintain the desired application specific rate coordination directives. We identify three major challenges in meeting this objective. First, the application specific performance measures must be formulated as rate coordination directives. Second, it is necessary to incorporate these rate coordination directives into a rate controller.

Of course, the resulting rate controller must co-exist with ordinary rate controllers, such as TCP Reno, in a shared network. Finally, a mechanism for identifying those flows that require the rate allocation directives must be put in place.

The first challenge is addressed by means of a utility function which allows the performance of the underlying application to be maximized. The second challenge is addressed by utilizing the Network Utility Maximization (NUM) framework. The standard utility function (i.e. utility function of the standard rate controller) is augmented by inserting the application specific utility function as an additive term. Then the rate allocation problem is formulated as a constrained optimization problem, where the objective is to maximize the aggregate utility of the network. The gradient projection algorithm is used to solve the optimization problem. The resulting solution is formulated and implemented as a window update function. To address the final challenge we resort to a machine learning algorithm. We demonstrate how data features estimated utilizing only a fraction of the flow can be used as evidential input to a series of Bayesian Networks (BNs). We account for the uncertainty introduced by partial flow data through the Dempster-Shafer (DS) evidential reasoning framework.

To my parents

Acknowledgements

I extend my sincere gratitude and appreciation to my dissertation advisor and chairman of the committee, Professor Kamal Premaratne, and my co-advisor Professor Manohar N. Murthi for their guidance, support, suggestions and words of encouragement throughout the period this research was being conducted. I am also thankful to Professors James W. Modestino, Xiaodong Cai of the Department of Electrical and Computer Engineering, and Professor Subramanian Ramakrishnan of the Department of Mathematics, for accepting the appointment to the dissertation committee and for their suggestions and support.

The financial assistance I received through U.S. National Science Foundation (NSF) Grants CNS-0519933 and CCF-0347229 is gratefully acknowledged. The financial assistance I received from the Department of Electrical and Computer Engineering is also acknowledged.

I would like to thank my many friends and colleagues at University of Miami with whom I have had the pleasure of working over the years. Finally, I would like to extend my utmost gratitude to my parents and my wife Gayani for their support, encouragement and love, which made this work possible.

Maththondage Chamara S. Ranasingha

University of Miami

June 2009

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

The ascendancy of packet networks has led to the development of many novel networked applications. For example, applications such as telesurgery and distributed data fusion of sensor data all rely upon the packet delivery capabilities of the network. Due mainly to the dependence on the packet delivery capabilities of the network, Quality of Service (QoS) of these networked applications are mainly dependent on the availability of network resources. Clearly, network resources (such as bandwidth on a link) are not unlimited, and must be shared by all networked applications in some manner of fairness [MW00]. Although an application in general would prefer to have a high bit-rate allocated to it, the fact that network links are being shared by multiple applications implies that this goal cannot be achieved. Indeed, since each link is shared among several different sets of users (depending on their particular routes), the sum of the bit-rates of each source application using a link must not exceed its capacity.

1.1 Motivation

Each user has to settle on a bit-rate while simultaneously satisfying the capacity constraints of the shared links. This rate at which each user settles on, or the

allocated bandwidth to a particular user, cannot be an arbitrary value. One must allocate the available bandwidth to the network users in an optimal way to maximize the service quality of the network users. To achieve this objective, the underlying applications of the network users need to be considered. Some applications may impose certain bandwidth allocation directives in order to maximize the service quality; other applications may impose some coordination among users in the process of bandwidth allocation.

Multi-sensor target tracking application is a clear example of such an application with bandwidth allocation requirements [RMPF09]. In this scenario, individual sensors must send the measurements over a packet network to a decision center where data fusion takes place in order to construct the track of the target. This packet network can be a network shared by several other applications including regular TCP flows. Individual sensors have different degrees of importance, usually determined by the distance between the target and the sensor and the sensor sensitivity. Therefore, at different times, the decision center will require more information from certain sensors than others.

The importance of the optimal bandwidth allocation in multi-sensor target tracking becomes even more apparent if the decision center desires to classify the target based on the sensor measurements as well. In this situation, both the tracking and classification performances need to be considered simultaneously in the process of bandwidth allocation. Another example is a battlefield data fusion scenario where a fusion center fuses information from possibly multiple sources to gather information about the battlefield. In this scenario, the fusion center desires to allocate its limited bandwidth to sources that are perceived to be more critical to the mission or are considered to provide more accurate information [HL01]. Such a scenario warrants a

mechanism that enables each 'sink' node to request a set of relevant nodes to transmit data according to certain bit-rate coordination requirements that are determined by the fusion task and other factors (e.g., delay and reliability of received information and its relevance to the mission objectives [ZPB02, RMP06]).

1.1.1 Rate Allocation

The 'traditional' packet network rate and congestion control algorithms, such as TCP Vegas and TCP Reno, do not take those directives which are given by the underlying application into account in the process of bandwidth allocation. These rate control algorithms take into account the utility of the network users by means of the *individual* bit-rates allocated to the network users; bit-rates of the other users are not considered in the process of rate allocation. In situations where the utility of the network users is not fully dependent on the individual bit-rates, a rate allocation algorithm that can consider application oriented directives in the process of rate allocation is warranted.

However, the rate allocation algorithm has to satisfy several requirements for it to be deployed in available shared networks. First, the rate allocation algorithm should be able to be implemented as a window flow control algorithm. This is vital in order to coexist with the available TCP variants. Second, the rate allocation algorithm should not assume that the routers can provide any additional feedback other than simple packet marking. Further, it should not assume special selective packet discarding capabilities. This requirement is important for the rate allocation algorithm to be deployed in available shared networks. The other perhaps the most important requirement is that the rate allocation algorithm should not penalize the regular TCP flows that utilize regular TCP Reno or TCP Vegas schemes. This

requirement is important to facilitate coexistence of the rate allocation algorithm with available TCP variants.

Several challenges have to be addressed in order to implement a rate allocation algorithm which can incorporate application specific rate allocation directives and satisfy these requirements. The first challenge is how to measure the QoS of applications in terms of transmission bit-rates. The next challenge is how to incorporate these QoS measures into a rate controller. Moreover, it is important to note that there may be several types of applications sharing the same network. Usually, these applications have different bandwidth allocation directives. For example, a target tracking application would have different rate allocation directives than a data fusion application. However, both the target racking and the data fusion applications need to share the same network. Then, the question of how to identify the flows corresponding to each application becomes critical because one must be able to identify the rate allocation directives associated with each flow. Once the flow is identified and the rate allocation directives are selected, then the appropriate rate allocation algorithm can be applied to satisfy the rate allocation directives. This of course calls for an effective strategy for flow classification.

1.1.2 Flow Classification

One obvious approach for flow classification is the inspection of the contents of the packets. However, flow classification via the inspection of the contents of the packets can be difficult to implement in real time. Moreover, this approach brings privacy issues and difficulties associated with encrypted data. Although port based classification is one alternative [Log, MKK+01], they are inefficient in the current

Internet setup mainly due to applications such as P2P [MHLB04, EMA06, NA06, PTK06, AMG07] which are known to use arbitrary port numbers.

The objective of this proposed work is to address these issues posed by the rate allocation requirements. Two main sections can be identified in this work. The first section is how to derive an efficient rate allocation algorithm in the form of a window update function that can utilize and coexist with the existing network infrastructure. The second section is the development of an accurate flow classification framework in order to facilitate the rate allocation directives.

1.2 Rate Allocation Approach

Network resource allocation methods (particularly rate/congestion control algorithms) are best explained by the Network Utility Maximization (NUM) approach pioneered by Kelly [KMT98, LL99, LPW02, Low03]. In the NUM view, each data emitting source node (which is sending data to a receiver over a packet network) is considered to possess a utility function which measures its QoS. This utility function is a concave function of the bit-rate of the source node only. For example, the function $U_s(f_s) = \alpha_s \log(f_s)$ in which f_s is the bit-rate of source s, and α_s is a constant, is a popular candidate that often used. Each source node wants to maximize its utility, which is equivalent to maximizing its bit-rate f_s. However, the sum of the individual bit-rates of all flows sharing a particular link cannot exceed the link capacity. Given a fixed routing infrastructure, the NUM problem is then stated as follows: Find a set of source bit-rates for all users in a network such that the sum of all the users' utility functions is maximized subject to the link capacity constraints. This is a convex optimization problem that has a unique solution. In networking, this problem is solved in

a decentralized, distributed manner using iterative methods that do not require communication amongst the data emitting sources [KMT98, LL99, LPW02, Low03]. The NUM framework has been used to design and analyze new networking rate/congestion control algorithms and active queue management (AQM) methods, and to explain the existing Internet standard TCP Reno [Low03]. Different rate control algorithms can be derived by selecting different utility functions [BP95, CWL04].

Most available approaches use utility functions that are functions of transmission bit-rates of individual sources only. However, it is impossible to incorporate application specific rate allocation directives if the utility function is solely a function of the transmission bit-rate of the source node, since coordination among different sources may be vital in application specific rate allocation. It is actually possible to define an utility function which is a function of transmission bit-rates of several sources in order to incorporate application specific rate allocation directives.

The work described in this dissertation demonstrates how the utility function in the NUM framework can be adopted to accommodate flow coordination directives dictated by the data sink node/task manager that is attempting to maximize an application specific QoS. In particular, we demonstrate how a standard flow control utility function can be augmented by an additive function that reflects the bandwidth coordination of flows. Through the proper choice of this additive term, the resulting rate resource allocation problem can be made to remain a convex optimization and it can be solved in a distributed manner using the gradient projection algorithm which is commonly used in network rate/congestion control. This approach leads to a bit-rate control algorithm in which all source nodes in the shared network update their bit-rates periodically. The ordinary source nodes in the shared network adjust their bit-rates as before, based on feedback from ACK packets. The data sources that

are participating in the proposed algorithm similarly update their bit-rates periodically. However, these sources adjust their rates using a combination of ordinary ACK packet feedback, and additional feedback from the data sink node which is directing the rate coordination. This approach of rate allocation can be extended to various applications. In our work, rate control algorithms for applications with rate *ratio* requirements and multi-sensor target tracking applications have been derived.

1.3 Flow Classification Approach

Machine learning techniques have been proposed to perform classification of flows by studying packet header information. Both *unsupervised* [MHLB04,RSSD04,KPF05, ZNA05, BTA$^+$06, EMA06, PTK06] and *supervised* [MZ05, LM06, NA06, AMG07] machine learning techniques have been proposed. These techniques include hidden Markov models (HMMs) [OSST04, WMM04], nearest neighbor methods [RSSD04], Bayesian methods [MZ05], Bayesian neural networks (BNNs) [AMG07] and BNs [Pea88, WZA06].

All these algorithms use full flow data in the process of traffic classification. However, rate allocation requires one to implement an on-line flow classification algorithm. The approach we propose in this work is based on a BN model of traffic flows. We employ a strategy where a window of packets, with the size of the window being gradually increased, enables online classification with a minimal number of packets. We propose to utilize a series of BNs, each corresponding to a different size of the window. This strategy of having a 'growing' window of increasing size necessitates the classification probabilities to be updated after each window update. This constitutes a major and vital difference between the proposed approach and existing

algorithms where classification probabilities are typically not continuously updated with the traffic flow progression.

We develop and utilize a DS belief theoretic method as the classification probability update mechanism. A DS theoretic update mechanism is employed to update the classification probabilities after each iteration of the window size increase. This scheme continues until the flow is classified. The advantage of DS theory is its ability to conveniently represent a wide variety of data imperfections [BP99]. DS theory provides an excellent framework for modeling imperfect data and as such, it has been extensively used in various applications in the past (e.g., see [BP99, DS04, HPS07]).

The rest of this dissertation is organized as follows. In Chapter 2, we review the available TCP rate controllers and the NUM framework. In Chapter 3, we review the available approaches for flow classification. A rate allocation algorithm to improve the QoS in multi-sensor target tracking is discussed in Chapter 4. A rate allocation algorithm to improve the performance of a target classification algorithm together with the target tracking is discussed in Chapter 5. Our flow classification algorithm is discussed in Chapter 6. A rate allocation algorithm to maintain rate ratio requirements is discussed in Chapter 7. Finally, Chapter 8 concludes the document.

CHAPTER 2

TCP Rate Controllers and the Network Utility Maximization (NUM) Framework

TCP rate controllers are used to control the transmission rates of sources in the current Internet. Several versions of TCP rate controllers have been proposed in the literature. The main objective of these rate controllers is to minimize the congestion of the network. TCP rate controllers decrease the transmission rate in an event of congestion in the network and increase the transmission rate if there is no congestion.

TCP rate controllers have window-based congestion control mechanisms [Bro00]. The transmission rate of the sender of the TCP flow is restricted by a quantity called the congestion *window size*. The window size is the maximum amount of data that the sender can send without receiving any ACK [RFC99]; it is also the maximum number of packets that may remain in the network loop formed between the sender and the receiver. Since it takes one round trip time (RTT) to receive the ACK after a packet is sent, the window size is also the maximum amount of data that the sender can send within one RTT. Hence, the congestion window size is primarily the determining factor of the transmission rate of the sender. The TCP congestion control mechanism uses two main phases to control the outstanding data to be injected into the network; the *slow start phase* and the *congestion avoidance phase*.

- **Slow Start Phase:** When a new connection is established, the TCP congestion control mechanism goes through the slow start phase. In this phase, the congestion window initially has one packet so that the sender can send only one packet before it gets any ACK. Thereafter, the sender increases its congestion window by one for each positive ACK it receives. So, in the initial state, the sender injects one packet into the connection and waits for the ACK. With the reception of the ACK, it increases the congestion window by one packet, allowing it to inject two packets. With the reception of ACKs from these two packets, it increases the congestion window to four, allowing the injection of four packets [RFC97]. The slow start phase continues until the congestion window reaches its threshold value (referred to as the *slow start threshold*) or until any packet loss is detected via a time out or duplicate ACKs. The TCP sender switches from the slow start phase to the congestion avoidance phase as the congestion window size passes the slow start threshold. The slow start phase resets with the reception of packet losses [Bro00, RFC99, RFC97].

- **Congestion Avoidance Phase:** After the sender's congestion window size exceeds the slow start threshold value, the TCP congestion control mechanism enters the congestion avoidance algorithm. The congestion avoidance phase does not aggressively increase the congestion window size or the transmission rate. Different TCP rate controllers use different algorithms to change the congestion window size while the flow is in the congestion avoidance phase. For a greater fraction of the connection time, TCP stays in the congestion avoidance phase. Hence, it is valid to assume that congestion avoidance is the dominant phase in the TCP congestion control mechanism [Bro00].

The basic challenge all these rate control algorithms must overcome is how to reliably identify network congestion when it truly occurs. Two main approaches are used for this purpose: (1) consider packet losses as an indication of congestion, (2) consider queuing delay, which is the delay caused by packets been queued at routers along the network path, as an indication of congestion. The rationale behind the first approach is that the intermediate nodes such as routers and switches would have to drop packets as the congestion of the network increases because they cannot dequeue packets at the rate that they can enqueue packets. The rationale behind the second approach is that network congestion would result in queues of the intermediate nodes to grow. Because of these two strategies of congestion identification, most available TCP rate controllers can be divided into two main categories: marking/loss based controllers and delay based controllers. However, controllers that combine both loss based congestion identification and delay based congestion identification have recently been proposed [CFM+09].

2.1 Marking/Loss Based TCP Controllers

TCP controllers that utilize marked or lost packets as network congestion indicators are very popular mainly due to the fact that the rate controller of the current Internet, TCP Reno, falls into this category. A brief review of the marking/loss based controllers including TCP Reno is given in this section.

2.1.1 TCP Reno

This is the most popular congestion control algorithm in the current Internet. The sender must reduce its congestion window size with the reception of a packet loss.

Similarly, a reception of an ACK means the system is experiencing reduced congestion and the sender needs to increase its congestion window size. The TCP sender detects packet losses using ACK timers or by the reception of duplicate ACKs. The congestion avoidance phase of TCP Reno increases the congestion window at most by one packet in one RTT. In the TCP Reno version, congestion control continues in the congestion avoidance phase after the reception of isolated losses (those which occur in between several positive ACKs) but reduces the congestion window size by one half of the original size (size at the time that the sender receives a packet loss). If several losses occur during a single RTT, TCP Reno will enter the slow start phase [Bro00]. Hence, the congestion window size changes, in the congestion avoidance phase, are called *additive increase and multiplicative decrease (AIMD)*. Note that the AIMD method of TCP Reno can be formulated as follows.

On ACK reception:

$$cwnd + a \rightarrow cwnd \tag{2.1}$$

On packet lost:

$$cwnd - b * cwnd \rightarrow cwnd \tag{2.2}$$

Here *cwnd* is the congestion window size and a and b are real parameters. Further note that, for TCP Reno, $a = 1$ and $b = 0.5$.

TCP Reno with Packet Marking

The main drawback associated with TCP Reno is its inability to avoid packet losses, since acknowledgment occurs after the packets are lost [FJ93, FB00, GK01, LM01,KBKL03] . To avoid packet losses, information regarding impending congestion in the router has to be transmitted to the sender, so that the sender can adjust its transmission rate beforehand to avoid the packet losses.

Explicit Congestion Notification (ECN) is the mechanism used to provide the feedback from the router to the sender about impending congestion [BCKC02]. To do this, the routers set (mark) the ECN bit of the packet header in order to notify the sender about impending congestion. In this mechanism, the router must be capable of observing its queue size and other available information and predicting impending congestion. Moreover, it should be capable of identifying those packets to be marked in order to propagate the severity of impending congestion. The *AQM* scheme is the tool used in the router for the congestion prediction and marking packet identification. First, it calculates the marking probability to reflect the severity of the impending congestion and then marks the packets with the determined probability. The sender is then able to adjust its transmission rate depending on the number of marked packets, in order to restrain possible congestion. This approach is significant because of its ability to mitigate the unnecessary packet losses [LM01, KBKL03]. The AQM scheme should be sophisticated enough to predict the congestion with reasonable accuracy. Several AQM schemes have been proposed to calculate the packet marking probability in the router. Among them, the *Random Early Detection (RED)* is the most popular. This method uses an average queue size as the parameter to calculate the packet marking probability [FJ93].

TCP Reno must now react to the reception of marked packets in order to reduce the transmission rate since the marked packets are an indication of impending congestion. This objective is achieved by treating marked packets similar to lost packets in TCP Reno. Hence, the same rate controller is used with the marked packets as well.

2.1.2 High-Speed TCP (HSTCP)

This is again a marking/loss based controller. This controller has been designed to be deployed in high speed situations where the congestion window needs to be a large number [Flo03]. TCP Reno has a disadvantage that it takes a long time to achieve large congestion windows and HSTCP tries to solve that problem. Let $cwnd_{low}$ and $cwnd_{high}$ be real parameters. Then the HSTCP algorithm sets a and b values for AIMD as follows. If $cwnd \leq cwnd_{low}$, then

$$a = 1, \text{ and } b = 0.5, \tag{2.3}$$

which is similar to TCP Reno. However, this selection is not appropriate for networks with high bandwidth delay product, since it can take extremely long time to achieve large congestion window sizes. So it is necessary to select a and b as functions of the current congestion window size $cwnd$ if $cwnd > cwnd_{low}$. The rationale behind the derivation of expressions for a and b is that the steady-state response of TCP Reno, which is $\frac{1.2}{\sqrt{p}}$ if p is the dropping probability, should be changed to $\frac{0.12}{p^{0.835}}$. To achieve this objective, following expressions are proposed:

$$b = (0.1 - 0.5) * \frac{\log cwnd - \log cwnd_{low}}{\log cwnd_{high} - \log cwnd_{low}} + 0.5, \tag{2.4}$$

and

$$a = 2 * cwnd^2 * \frac{0.078}{cwnd^{1.2}} * \frac{b}{2 - b}. \tag{2.5}$$

2.1.3 Binary Increase Congestion (BIC) TCP

This marking/loss based controller is specifically designed to address the issue of high delay high bandwidth networks. In such an environment, TCP Reno can take

an unacceptably long time to achieve its equilibrium. This controller consists of two main parts [XHR04].

1. **Binary Search Increase**

   This algorithm maintains two quantities; $cwnd_{max}$ and $cwnd_{min}$. Initially,

   $$cwnd \rightarrow cwnd_{min}, \tag{2.6}$$

   where $cwnd$ is the current window size. When a packet loss is detected $cwnd_{max}$, $cwnd_{min}$ and $cwnd$ are updated as follows.

   $$cwnd \rightarrow cwnd_{max}, \quad b \times cwnd \rightarrow cwnd, \quad cwnd \rightarrow cwnd_{min}. \tag{2.7}$$

   Here b is a constant with default value of 0.2. Then performs the binary search by jumping to the mid-point $\frac{cwnd_{min} + cwnd_{max}}{2}$. If packet losses are not detected then $cwnd_{min}$ is updated as

   $$cwnd \rightarrow cwnd_{min}. \tag{2.8}$$

   Then increase the window size to $\frac{cwnd_{min} + cwnd_{max}}{2}$.

2. **Additive Increase**

   If $\frac{cwnd_{min} + cwnd_{max}}{2} - cwnd > S_{max}$, where S_{max} is a parameter, then the algorithm switches to additive increase instead of changing directly to the target window.

2.1.4 Scalable TCP (STCP)

Here, the TCP Reno window update function is slightly modified in order to get an algorithm which is suitable for high-speed wide area networks. The scalable TCP

(SCTP) algorithm is as follows [Kel03].

On ACK reception:

$$cwnd + 0.01 \rightarrow cwnd \tag{2.9}$$

On packet lost:

$$0.875 * cwnd \rightarrow cwnd \tag{2.10}$$

Here *cwnd* is the size of the congestion window.

2.2 Delay Based TCP controllers

The total queuing delay experienced by the source is considered as the congestion notification in delay based TCP controllers. The strategy the source utilizes to measure the total queuing delay it experiences is an important factor that has a significant impact on the performance of a delay based TCP controller. The most common approach a source utilizes is to treat the difference between the current RTT and the minimum observed RTT as the total queuing delay. The underlying assumption in this approach is that the minimum observed RTT is the propagation delay.

2.2.1 TCP Vegas

In TCP Vegas it is proposed to react to the congestion before the packet losses are materializes. The congestion avoidance strategy of TCP Vegas can be summarized as follows [BP95].

- **Step1:** Estimate the quantity $BaseRTT$ by taking the minimum observed RTT.
- **Step2:** Calculate the expected throughput $Expected$ by

$$Expected = \frac{WindowSize}{BaseRTT}, \tag{2.11}$$

where $WindowSize$ is the current congestion window size.

- **Step3:** Calculate the actual throughput $Actual$ by

$$Actual = \frac{WindowSize}{CurrentRTT}. \tag{2.12}$$

- **Step4:** The new window size is calculated as follows.

$$WindowSize = \begin{cases} WindowSize + 1 & \text{if} \quad Expected - Actual < \alpha_s; \\ WindowSize - 1 & \text{if} \quad Expected - Actual > \beta_s; \\ WindowSize & \text{otherwise,} \end{cases} \tag{2.13}$$

where $\{\alpha_s, \beta_s\},\ 0 < \alpha_s \leq \beta_s$ are parameters. When the actual throughput gets closer to the expected throughput (i.e., $Expected - Actual < \alpha_s$), the connection is likely to under utilize the link. Hence it is necessary to increase the congestion window. On the other hand if the actual throughput getting farther away from the expected throughput (i.e., $Expected - Actual > \beta_s$) then the congestions are possible and the congestion window needs to be decreased. Furthermore, it is vital to keep the congestion window unchanged if $\alpha_s \leq Expected - Actual \leq \beta_s$ in order to reduce the fluctuations.

2.2.2 Enhanced TCP Vegas

Even though TCP Vegas achieves high link utilization compared to TCP Reno, it cannot prevent link under utilization in the presence of backward link congestion [CCC03]. Enhanced TCP Vegas attempts to address this very issue. A TCP time stamp mechanism is proposed to overcome the problems posed by backward congestion. With the time stamp mechanism, a few minor modifications have been proposed to the standard TCP Vegas algorithm. The actual throughput is estimated

as

$$Actual = \frac{WindowSize}{CurrentRTT - QD_b}, \tag{2.14}$$

where QD_b is the backward queuing delay. In the standard TCP Vegas, the propagation delay $BaseRTT$ is estimated by taking the minimum observed RTT. However, in the enhanced TCP Vegas, the propagation delay is estimated as the summation of the minimum observed delay in the forward direction and the minimum observed delay in the backward direction.

2.2.3 Fast TCP

Fast TCP is another delay based TCP controller. It is designed to facilitate faster convergence to the equilibrium than TCP Vegas. In Fast TCP, it is required to maintain a quantity $AvgRTT$ as follows:

$$AvgRTT = (1 - \beta) * AvgRTT + \beta * CurrentRTT, \tag{2.15}$$

where $\beta = \min\{3/WindowSize, 1/8\}$. Once the $AvgRTT$ is calculated, the $WindowSize$ can be calculated using [CWL04]

$$\begin{aligned} WindowSize = \min \Big\{ & 2\,WindowSize, \\ & (1-\gamma)\,WindowSize + \gamma \left(\frac{BaseRTT}{AvgRTT} * WindowSize + \alpha \right) \Big\}, \end{aligned} \tag{2.16}$$

where $\gamma \in (0, 1]$.

2.3 NUM Framework

The NUM framework was originally proposed and utilized to model the available TCP controllers [KMT98, LL99, LPW02, Low03]. However, because of the flexibility

of the NUM framework, many researchers have since used it to develop more sophisticated rate controllers. In the view of NUM framework, every source in the network measures its performance by means of an utility function. This utility function is a concave function of the transmission rate, and the objective of each source is to maximize its individual utility. The global objective of the network is to maximize the aggregate utility of all the sources in the network. However, summation of transmission rates for a link cannot exceed its link capacity. Hence, the global objective of the network can be formulated as a constrained optimization problem. Individual source rates can be calculated by solving the given optimization problem. Let us now discuss the detail of the NUM framework.

To proceed, let us define the following notations:

$\mathfrak{L}$	index set of links in the network
$\mathfrak{S}$	index set of sources in the network
L_s	index set of links utilized by the source s
$\mathfrak{C} = (c_\ell, \ell \in \mathfrak{L})$	capacities of the links in the network
R	routing matrix where entry $R_{\ell s} = 1$ if $\ell \in L_s$, 0 otherwise
$f_s(t)$	transmission rate of the source s at the time instance t
$p_\ell(t)$	congestion measure of the link ℓ at the time instance t
$q_s(t)$	aggregate congestion measure experienced by the source s
$y_\ell(t)$	aggregate transmission rate of the link ℓ

Now, we can model a network by $\mathfrak{L}$, $\mathfrak{S}$ and R. Note that $y_\ell(t) = \sum_s R_{\ell s} f_s(t)$ and $q_s(t) = \sum_\ell R_{\ell s} p_\ell(t)$. Moreover, denote in the vector form $\mathbf{f}(t) = (f_s(t), s \in \mathfrak{S})$, $\mathbf{q}(t) = (q_s(t), s \in \mathfrak{S})$, $\mathbf{y}(t) = (y_\ell(t), \ell \in \mathfrak{L})$ and $\mathbf{p}(t) = (p_\ell(t), \ell \in \mathfrak{L})$. Now we can write the above identities in the matrix form as

$$\mathbf{y}(t) = R\mathbf{x}(t) \quad \text{and} \quad \mathbf{q}(t) = R^T \mathbf{p}(t). \tag{2.17}$$

Now note that for every rate controller, the source updates the transmission rate

based on the current transmission rate $f_s(t)$ and the current congestion measure $q_s(t)$. So we can write

$$f_s(t+1) = F_s(f_s(t), q_s(t)), \tag{2.18}$$

for some function $F_s(\bullet)$. At equilibrium, since $f_s(t+1) = f_s(t) = f_S$, we have

$$f_s = F_s(f_s, q_s). \tag{2.19}$$

If F_s is continuously differentiable and $\dfrac{\partial F_s}{\partial q_s} \neq 0, \quad \forall f_s > 0 \quad \text{and} \quad q_s > 0$, there exists a unique continuously differentiable function u_s such that

$$q_s = u_s(f_s) > 0. \tag{2.20}$$

Then we can define the utility function $U_s(f_s)$ for the source s as

$$U_s(f_s) = \int u_s(f_s)\, df_s, \quad f_s \geq 0. \tag{2.21}$$

Furthermore, it can be shown that $U_s(f_s)$ is a concave function of f_s. Given the function F_s, the network objective can be formulated as the following constrained optimization:

$$\max_{\mathbf{f} \geq 0} \sum_s U_s(f_s) \quad \text{subject to} \quad \sum_s R_{\ell s} f_s \leq C_\ell, \ \forall \ell, \tag{2.22}$$

where the constraint states that the summation of the transmission rate of all the flows going through a particular link should not exceed the capacity of that particular link.

Now let us concentrate on how to derive a rate control algorithm if the utility functions U_s are given for all the sources. In order to find the optimal set of transmission rates, we need to solve the constrained optimization given in (2.22). It is necessary to have U_s as a concave function of f_s in order to have an unique solution.

First we define the Lagrangian $L(\mathbf{f}, \mathbf{p})$

$$\begin{aligned} L(\mathbf{f}, \mathbf{p}) &= \sum_s U_s(\mathbf{f}) + \sum_\ell p_\ell \left(C_\ell - \sum_s R_{\ell s} f_s \right) \\ &= \sum_s \left(U_s(\mathbf{f}) - f_s \sum_\ell R_{\ell s} p_\ell \right) + \sum_\ell C_\ell p_\ell. \end{aligned}$$

Let $(\mathbf{f}^*, \mathbf{p}^*)$ be the optimal equilibrium $(\mathbf{f}, \mathbf{p})$. Then we have

$$\mathbf{f}^* = \arg \max_{\mathbf{f} \geq 0} L(\mathbf{f}, \mathbf{p}^*). \tag{2.23}$$

Hence we can further state that

$$\frac{\partial L(\mathbf{f}^*, \mathbf{p}^*)}{\partial f_s} = 0 \quad \forall s \in \mathfrak{S}. \tag{2.24}$$

Hence at the equilibrium

$$U_s'(f_s^*) = q_s^* = \sum_\ell R_{\ell s} p_\ell^*. \tag{2.25}$$

Now the problem is how to achieve this equilibrium. As shown in [Ber99], the gradient projection algorithm enables one to achieve the optimum $(\mathbf{f}^*, \mathbf{p}^*)$ in an iterative manner by updating the transmission rates of individual sources in a distributed manner. In a networked setting, this allows for the distributed iterative solution of the optimization problem without explicit communication among the different sources.

To examine how the gradient projection algorithm could be used to achieve a practical rate-control equation, consider

$$\mathbf{f}(t+1) = \left[\mathbf{f}(t) + \mu(t) \left[\overline{\mathbf{f}}(t) - \mathbf{f}(t) \right] \right]^+, \tag{2.26}$$

where $[f]^+ = f$ if $f \geq 0$ and 0 otherwise, and

$$\overline{\mathbf{f}}(t) = \mathbf{f}(t) + s(t) \nabla L(\mathbf{f}(t), \mathbf{p}(t)). \tag{2.27}$$

The quantities $\mu(t)$ and $s(t)$ must be selected with due regard to system stability. A constant step size (i.e., $\mu(t) = 1$ and $s(t) = s$, where s is a constant) is suitable in

terms of stability. The modified algorithm then becomes

$$\mathbf{f}(t+1) = [\mathbf{f}(t) + s\,\Delta L(\mathbf{f}(t), \mathbf{p}(t))]^+ . \tag{2.28}$$

Based on (2.28), consider the rate update function for the transmission rate of source s:

$$f_s(t+1) = \left[f_s(t) + s\,\frac{\partial}{\partial f_s} L(\mathbf{f}, \mathbf{p}) \right]^+$$

$$f_s(t+1) = [f_s(t) + s\,(U_s'(f_s(t)) - q_s(t))]^+ .$$

2.4 NUM Interpretation of Available TCP Algorithms

The available TCP rate controllers can be interpreted using the above mentioned NUM framework. The main task here is the derivation of the corresponding utility function.

2.4.1 TCP Reno

In TCP Reno the window size is increased by 1 packet per every RTT if there is no congestion. On the other hand, the window size is decreased by half when a packet drop is detected. Hence, in each period t, the window is increased by $1/w_s(t)$ with probability $1-q_s(t)$ and decreased to $w_s(t)/2$ with probability $q_s(t)$. Here, $w_s(t)$ is the window size of the source s, and $q_s(t)$ is the marking or dropping probability experienced by the source s. Then we can calculate the average change in window size in period t as

$$\frac{1}{w_s(t)}(1 - q_s(t))f_s(t) - \frac{w_s(t)}{2} q_s(t) f_s(t),$$

and write the F_s function in (2.18) for TCP Reno as

$$F_s(f_s(t), q_s(t)) = f_s(t) + \frac{1}{w_s(t)}(1 - q_s(t))f_s(t) - \frac{w_s(t)}{2} q_s(t) f_s(t). \tag{2.29}$$

Since $w_s(t)/RTT_s$ we can write

$$q_s = \frac{2}{2 + f_s^2 RTT_s^2} = u_s(f_s). \tag{2.30}$$

Finally the utility function can be derived as,

$$U_s(f_s) = \int u_s(f_s) df_s \ = \frac{\sqrt{2}}{RTT_s} \arctan\left(\frac{f_s RTT_s}{\sqrt{2}}\right). \tag{2.31}$$

2.4.2 TCP Vegas

The window update function for TCP Vegas can be summarized as follows.

$$w_s(t+1) = \begin{cases} w_s(t) + 1 & \text{if} \quad \dfrac{w_s(t)}{d_s} - \dfrac{w_s(t)}{RTT_s} < \alpha_s; \\ w_s(t) - 1 & \text{if} \quad \dfrac{w_s(t)}{d_s} - \dfrac{w_s(t)}{RTT_s} > \beta_s; \\ w_s(t) & \text{otherwise,} \end{cases} \tag{2.32}$$

where d_s is the propagation delay of the source s. If q_s is the queuing delay of the source s, then $RTT_s = d_s + q_s$. For the purpose of this discussion let $\beta_s = \alpha_s$. Now, at the equilibrium,

$$\frac{w_s(t)}{d_s} - \frac{w_s(t)}{RTT_s} = \alpha_s. \tag{2.33}$$

The we can write

$$q_s = \frac{\alpha_s d_S}{f_s} = u_s(f_s). \tag{2.34}$$

Finally the utility function can be derived as,

$$U_s(f_s) = \int u_s(f_s) df_s \ = \alpha_s d_S \log f_s. \tag{2.35}$$

CHAPTER 3

Available Approaches for Flow Classification

Of all the approaches available for classification of network flows, perhaps the most accurate approach is the packet content inspection. Unfortunately, inspection of the content of the packets can be extremely computationally intensive and hence difficult to implement in real time. Moreover, this approach brings privacy issues to the fore. In addition, such methods must also overcome difficulties associated with encrypted data. Because of these difficulties, alternative approaches have been proposed and utilized in the research literature.

Port based methods are perhaps the most well established and well researched approaches of flow classification [Log, MKK+01]. This approach is based on the fact that different applications are associated with different IP port numbers. The classifier can simply look at the IP header and perform the classification task based on the port number. Although port based methods had been employed quite successfully in the past, it is considered to be unreliable in the current Internet setup due mainly to their inability to identify applications that utilize arbitrary port numbers, e.g., P2P [MHLB04, EMA06, NA06, PTK06, AMG07]. Furthermore, certain applications are known to intentionally mislead classification by using port numbers that are more commonly associated with other applications.

Traffic flow classification methods that are inspired by machine learning techniques have recently attracted the attention of researchers. Such machine learning approaches typically perform classification of Internet traffic by studying packet header information. These approaches can be classified as either *unsupervised* [MHLB04, RSSD04,KPF05,ZNA05,BTA$^+$06,EMA06,PTK06] or *supervised* [MZ05,LM06,NA06, AMG07]. Unsupervised approaches attempt to classify flows into classes possessing the same characteristics without any regard to the application associated with the flow. On the other hand, supervised approaches attempt to associate an application class for a given flow.

Flow classification algorithms can also be classified according to the duration of flow required to make a classification decision. This criterion categorizes flow classification algorithms as *full flow information based approaches* and *partial flow information based approaches*. Most of the available classification algorithms fall into the category of full flow information based approaches because they use full flow data in the process of traffic classification. These full flow feature values (e.g., flow duration, number of packets, number of packets in the forward/backward directions, etc.) can be computed or estimated only at the flow termination. Such a strategy can be very useful for management and planning purposes. Real time bandwidth allocation however require partial flow information based approaches so that flow classification in real time can be carried out. There are only a few approaches that have been proposed to conduct flow classification based only on a small fraction of the flow.

3.1 Full Flow Information Based Approaches

Various machine learning tools have been proposed for flow classification utilizing full flow information.

3.1.1 Hidden Markov Models (HMMs)

Oveissian, et al. [OSST04] and Wright, et al. [WMM04] use HMMs for Internet traffic classification. Oveissian, et al. [OSST04] demonstrate that a two state HMM can be used for this purpose. Further, it is observed that a two state HHM is sufficiently well able to discern among different flows even though it may not be possible to describe a flow accurately. Wright, et al. [WMM04] use the HMM approach to identify the underlying protocol of the flows. A Markov model with four states, INSERT, SERVER MATCH, CLIENT MATCH and DELETE, is used for this purpose. Both Oveissian, et al [OSST04] and Wright, et al [WMM04] use the well known EM algorithm [DLR77] for the estimation of model parameters.

3.1.2 QoS Mapping Approaches

There are few flow classification algorithms designed specifically for QoS mapping, e.g. [MHLB04]. The distinctive feature of the QoS mapping flow classification technique is that the classification process uses a smaller number of classes. The objective of this approach is to classify traffic into a smaller number of classes with similar properties without attempting to identify the underlying application. Roughan, et al. [RSSD04] also propose an approach for classification of flows into different classes with similar properties. This approach is designed in a way such that the classification is good enough for the purpose of QoS mapping. Two classification algorithms,

the nearest neighbor (NN) method and the linear discriminant analysis, are utilized to map different flows into different QoS classes. This approach works well with a smaller number of classes.

3.1.3 Bayesian Methods

Moore and Zuev [MZ05] examine Bayesian methods for Internet traffic classification. The basis for this approach is that the conditional probability of the class C_j given the feature y can be written as

$$P(C_j|y) = \frac{P(C_j)p(y|C_j)}{\sum_i P(C_i)p(y|C_i)}, \tag{3.1}$$

where $P(C_j)$ is the prior probability of the class j and $P(y|C_j)$ is the likelihood conditional probability of the feature y. The classification is done by selecting the class with the highest likelihood $P(C_j|y)$ value. The quantities $P(C_j)$ and $P(y|C_j)$ are estimated in the training process. There are two methods proposed to estimate $P(y|C_j)$. In the first method, $P(y|C_j)$ is approximated by a normal distribution and the parameters of the normal distribution (i.e., mean and variance) are estimated in the training process. The second method is the kernel estimation. In the kernel estimation , we use

$$P(y|C_j) = \frac{1}{n_{C_j}h} \sum_{x_i:C(x_i)=C_j} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(y-x_i)^2}{2h^2}\right), \tag{3.2}$$

where n_{C_j} is the number of instances of the class C_j and $C(x_i)$ is the class of x_i. The complexity of the kernel estimation is much higher even though higher accuracies can be achieved.

3.1.4 Bayesian Neural Networks (BNNs)

Auld, et al. [AMG07] propose a BNN for classification of Internet traffic flows. BNN is a neural network which has probabilistic outputs. It is shown that a reasonable accuracy can be achieved with a BNN with only one hidden layer. The input layer's number of nodes is equal to the number of features. The output layer's number of nodes is equal to the number of classes. A *soft max* filter is used in the output layer to convert the outputs into probabilities. Each node in the output layer generates the corresponding class probability. The classification is done by selecting the output node with the highest probability. All nodes in the network use hyperbolic tangent activation functions. The conjugate gradient algorithm is used for training. This approach requires the use of a substantially larger number of features.

3.2 Partial Flow Information Based Approaches

There appear to be mainly two approaches in the literature that address the challenges associated with real time flow classification. The first approach performs classification based on the first few packets of the flow. For example, Bernaille, et al. [BTA$^+$06, BTS06] propose a method where features extracted from only the first five packets of the flow are utilized. After feature extraction, an unsupervised machine learning algorithm is used for the classification. This approach seems to perform with lower accuracies with the unknown traffic flows even though classification accuracies are good for known applications. Erman, et al. [EMA$^+$07] utilized a semi-supervised method to perform traffic classification and demonstrated that their approach can be extended to online classification by selecting only the first few packets for the classification. The second approach that has been proposed for real time traffic classification

is the use of features from multiple sub-flows for classification purposes. For example, the method in Nguyen and Armitage [NA06] uses 25 packets at the beginning and 25 packets at the middle of each flow for classification purposes. The performance of this approach may depend on the locations of the sub-flows. Moreover, short flows may create difficulties.

CHAPTER 4

Rate Allocation in an Application with a Single Objective: Multi-Sensor Target Tracking

Multi-sensor target tracking is one of the most important applications that require the imposition of transmission rate allocation directives. Intuitively, one expects that, as the target moves, the sensors closer to the target with more relevant estimates should transmit their data to a decision center/sink node at a higher rate than those sensors with less relevant data. Accordingly, the bandwidth resources of the network must be allocated and distributed in a manner that ensures the sensors with the more reliable measurements receive a higher proportion of the available bandwidth.

Many aspects of the challenges posed by the multi-sensor target tracking applications operating over a shared network are discussed in the literature. For example, [OS07] discusses a distributed Kalman filter (KF) approach where central fusion is not used. The work in [SSF+04] derives the optimal KF algorithm to handle packet loss. The work in [SC06] demonstrates how distributed target tracking algorithms can be adapted to handle packet losses and delays by performing local estimates that are transmitted to a sink node (or decision center). The work in [ONV06] proposes a framework for target tracking using quantized data transmitted over noisy channel. The work in [RGR06] and [ZMVM06] present algorithms for bandwidth efficient

target tracking, based on KF and particle filtering frameworks, respectively. An efficient sensor scheduling algorithm for target tracking is proposed in [SM08]. A sensor selection approach for target tracking is in [ZNV08].

However, the problem of optimal allocation of the available bandwidth among different sensors in multi-sensor target tracking appears not to have been studied in the literature [RMPF08,RMPF09]. Our objective is to develop a new rate/congestion control algorithms that can take into account the QoS requirements of the multi-sensor target tracking application. Moreover, this particular rate control algorithm should be able to be deployed in the current Internet without changing the available infrastructure.

Our work focuses on KF based multi-sensor target tracking applications [AG92, Sah96,BP99,CYMBKC00,GH01,GGB+02,FF99]. In particular, we examine the case in which multiple sensors transmit their readings over a shared packet network to a decision center/sink node that fuses the data and calculates the target track [HD06]. First, we suitably define a target tracking utility function that measures the QoS of the target tracking application. Then the standard rate/congestion control utility function is augmented with the target tracking utility function, leading to a modified rate/congestion control convex optimization problem. The modified optimization problem is solved using the gradient projection algorithm.

4.1 KF Based Target Tracking

KF is one of the most common and studied approaches for target tracking [Sah96, BP99,CYMBKC00,GH01,GGB+02,FF99]. We start with a description of the basic model for the KF based target tracking by first considering a single sensor model.

Let us denote the target's position (measured using the sensors), velocity (estimated using the KF), and acceleration at the k-th time instant in the x and y directions via $(x(k), v_x(k), a_x(k))$ and $(y(k), v_y(k), a_y(k))$, respectively. Let $\Delta = t(k+1) - t(k)$, where $t(k)$ is the time in seconds of the occurrence of event k. Here, Δ is the period of the KF update and it is kept as a constant (i.e., independent of k).

4.1.1 Plant Equation

With the KF state vector taken as

$$X(k) = \begin{bmatrix} x(k) & v_x(k) & y(k) & v_y(k) \end{bmatrix}^T, \tag{4.1}$$

and the target acceleration $(a_x(k), a_y(k))$ treated as the plant noise [Wat98, KBS03, DL04], the equations of motion of the target can be described via

$$X(k+1) = \Phi\, X(k) + \Gamma\, W(k), \tag{4.2}$$

where

$$\Phi = \begin{bmatrix} 1 & \Delta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta \\ 0 & 0 & 0 & 1 \end{bmatrix};\ \Gamma(k) = \begin{bmatrix} \frac{\Delta^2}{2} & 0 \\ \Delta & 0 \\ 0 & \frac{\Delta^2}{2} \\ 0 & \Delta \end{bmatrix};\ W(k) = \begin{bmatrix} a_x(k) \\ a_y(k) \end{bmatrix}. \tag{4.3}$$

Here, the plant noise vector $W(k)$ at $t(k)$ is taken to be a white noise process with zero mean and covariance matrix $Q(k)$, i.e., $W(k) \sim (0, Q(k))$.

4.1.2 Measurement Equation

We take the measurement equation of the KF to be

$$Z(k) = H\, X(k) + V(k), \tag{4.4}$$

where

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}; \quad V(k) = \begin{bmatrix} d_x(k) \\ d_y(k) \end{bmatrix}. \tag{4.5}$$

Here, the measurement noise vector $V(k)$ at $t(k)$ is taken to be $V(k) \sim (0, R(k))$. Two components contribute to measurement noise: the first component is the error introduced by the proximity of the sensor (sensor proximity error) to the target; the second component is the error introduced by finite bit-rate approximation of the measurements which cannot be transmitted with an infinite bit-rate.

4.1.3 KF Update Equations

The KF equations that yield the target position and velocity estimates are the following [Wat98, KBS03, DL04]:

$$\hat{X}(k+1|k+1) = \Phi\, \hat{X}(k|k) + K(k+1) \left[Z(k+1) - H\, \Phi\, \hat{X}(k|k) \right], \tag{4.6}$$

where

$$K(k+1) = P(k+1|k)\, H^T \left[H\, P(k+1|k)\, H^T + R(k+1) \right]^{-1}; \tag{4.7}$$

$$P(k+1|k+1) = [I - K(k+1)\, H]\, P(k+1|k); \tag{4.8}$$

$$P(k+1|k) = \Phi\, P(k|k)\, \Phi^T + \Gamma\, Q(k)\, \Gamma^T. \tag{4.9}$$

4.1.4 Extension to Multi-Sensor Target Tracking Environments

As mentioned earlier, KF principles can be used to construct a target track using data from more than one sensor. Two approaches, *measurement fusion (MF)* and *state-vector fusion (SF),* are often used to perform the multi-sensor target tracking [GH01]. Let us consider a multi-sensor target tracking scenario with N sensors.

MF Method

MF uses a single KF to fuse all the sensor measurements and thereby obtain the fused track of the target. Within MF, two main methods MF1 and MF2 are utilized [GH01].

MF1 Method In this method, measurements from all the sensors are stacked before being fed into the KF. This is achieved by augmenting the measurement vector $Z(k)$ as

$$Z(k) = \begin{bmatrix} Z_1(k) & \cdots & Z_N(k) \end{bmatrix}^T, \tag{4.10}$$

where $Z_i(k)$ is the measurement vector for the i-th sensor. Accordingly, a $2N \times 4$ sized measurement model matrix is obtained as

$$H = \begin{bmatrix} H_1 & \cdots & H_N \end{bmatrix}^T, \tag{4.11}$$

where H_i is the measurement model matrix for the i-th sensor. The corresponding $2N \times 2N$ sized measurement error covariance matrix is

$$R(k) = diag\left\{R_1(k), \cdots, R_N(k)\right\}, \tag{4.12}$$

where $R_i(k)$ is the measurement error covariance matrix for the i-th sensor.

MF2 Method In this method, measurements are initially fused before being fed into a single KF, i.e.,

$$Z(k) = \left[\sum_{i=1}^{N} R_i(k)^{-1}\right]^{-1} \sum_{i=1}^{N} R_i(k)^{-1} Z_k(k). \tag{4.13}$$

The measurement error covariance matrix $R(k)$ is given as

$$R(k) = \left[\sum_{i=1}^{N} R_i(k)^{-1}\right]^{-1}. \tag{4.14}$$

In our target tracking application, the measurement model matrices for all the N sensors are identical, i.e., $H_i \equiv H,\ \forall i = 1, \ldots, N$. When this is the case, MF1 and MF2 are functionally identical [GH01]. Therefore, from now on, we will concentrate on MF2 method only.

SF Method

SF uses an individual KF for each sensor and then fuses the estimates from this bank of KFs to get the fused estimate of the track. The basic convex combination of estimates can be used to generate the fused estimate of the target track [CYMBKC00]. Let the state estimate of the i-th sensor be $\hat{X}_i(k|k)$ and the error covariance matrix of the i-th sensor be $P_i(k|k)$. Then the fused estimate $\hat{X}(k|k)$ is given as

$$\hat{X}(k|k) = P(k|k) \sum_{i=1}^{N} P_i(k|k)^{-1} \hat{X}_i(k|k), \tag{4.15}$$

where $P(k|k)$ is the fused error covariance matrix given by

$$P(k|k) = \left[\sum_{i=1}^{N} P_i(k|k)^{-1} \right]^{-1}. \tag{4.16}$$

4.2 Utility of Target Tracking

The objective of this work is to find an optimal set of sensor transmission rates in order to maximize the target tracking performance in a shared network. In the process of finding the optimal set of transmission rates, the first step should be to find a quantity by which the target tracking performance can be measured. This quantity to be maximized is the *utility* of the target tracking application. In KF based estimation, one typically minimizes the trace of the error covariance matrix in order to maximize estimation accuracy. Hence, it is reasonable to measure the

performance of the target tracking application by the negative of the trace of the filtered error covariance matrix at time instance $k+1$, i.e., $-Tr[P(k+1|k+1)]$. To simplify the analysis, and to emphasize the location estimates, we define the utility as the trace of the components of $P(k+1|k+1)$ that correspond to location estimates. Recalling that the first and third entries of $X(k)$ in (4.1) correspond to location, we define the utility $S(k+1)$ as

$$S(k+1) \equiv -P(k+1|k+1)_{1,1} - P(k+1|k+1)_{3,3}. \tag{4.17}$$

Here, $[\bullet]_{i,i}$ denotes the i-th diagonal entry of the matrix $[\bullet]$.

We may maximize the QoS of the multi-sensor target tracking application by maximizing the utility $S(k+1)$. To proceed, we need to derive a closed form expression for $S(k+1)$ as a function of the measurement errors of the multiple sensors. It is clear that different methods of multi-sensor fusion may have different expressions for $S(k+1)$. Hence, the two fusion methods, MF and SF are considered separately. In what follows, we make the following assumptions:

- $Q(k) \equiv Q_i(k) = q\, I_2$, $\forall i = 1, \ldots, N$, $\forall k$, where $q > 0$ is a positive scalar and I_2 is the 2×2 identity matrix. So, the plant noise covariances are identical and diagonal (indicating independent noise components) in all the sensors.

- $R_i(k) = r_i(k)\, I_2$, where $r_i(k) > 0$ is a positive scalar. With MF2 method of fusion, the measurement error covariance matrix in (4.14) then becomes

$$R(k) = r(k)\, I_2, \text{ where } \frac{1}{r(k)} = \sum_{i=1}^{N} \frac{1}{r_i(k)}. \tag{4.18}$$

We will also need the following

Definition 1 *A 4×4 matrix P that takes the form $P = \begin{bmatrix} \Phi & \emptyset \\ \emptyset & \Phi \end{bmatrix}$, where Φ is a 2×2 symmetric matrix, is said to have a* Φ-block diagonal form. ■

4.2.1 MF Method

Claim 1 *Suppose the initial error covariance matrix $P(0|0)$ is $\Phi(0)$-block diagonal with arbitrary $\Phi(0)$. Then, with MF, the following are true:*

(i) The error covariance matrix $P(k|k)$ is $\Phi^{MF}(k)$-block diagonal with

$$\Phi^{MF}(k) = \begin{bmatrix} p_1^{MF}(k) & p_2^{MF}(k) \\ p_2^{MF}(k) & p_4^{MF}(k) \end{bmatrix}.$$

(ii) The utility $S(k+1)$ can be expressed as

$$S^{MF}(k+1) = \frac{-2\,a^{MF}(k)}{a^{MF}(k)/r(k+1)+1}.$$

The entries of $\Phi^{MF}(k)$ and $S^{MF}(k+1)$ above appear in the proof.

Proof: By induction. The claim is trivially true for $k = 0$ since $P(0|0)$ is $\Phi(0)$-block diagonal. Suppose the claim is true for k.

(i) Given that $P(k|k)$ is $\Phi^{MF}(k)$-block diagonal with entries given by item (i) of Claim 1, substitute $P(k|k)$ into (4.9) to show that $P(k+1|k)$ is $B^{MF}(k)$-block diagonal with

$$B^{MF}(k) = \begin{bmatrix} a^{MF}(k) & b^{MF}(k) \\ b^{MF}(k) & c^{MF}(k) \end{bmatrix},$$

where

$$\begin{aligned} a^{MF}(k) &= p_1^{MF}(k) + 2\,p_2^{MF}(k)\,\Delta + p_4^{MF}(k)\,\Delta^2 + \frac{q}{4}\,\Delta^4; \\ b^{MF}(k) &= p_2^{MF}(k) + p_4^{MF}(k)\,\Delta + \frac{q}{2}\,\Delta^3; \\ c^{MF}(k) &= p_4^{MF}(k) + q\,\Delta^2. \end{aligned} \tag{4.19}$$

From (4.14), we note that $R(k+1) = diag\{r(k+1), r(k+1)\}$. Substitute $P(k+1|k)$ and $R(k+1)$ into (4.8) to show that $P(k+1|k+1)$ is $\Phi^{MF}(k+1)$-block diagonal with

$$\Phi^{MF}(k+1) = \begin{bmatrix} p_1^{MF}(k+1) & p_2^{MF}(k+1) \\ p_2^{MF}(k+1) & p_4^{MF}(k+1) \end{bmatrix}, \tag{4.20}$$

where

$$p_1^{MF}(k+1) = \frac{a^{MF}(k)}{a^{MF}(k)/r(k+1)+1};$$
$$p_2^{MF}(k+1) = \frac{b^{MF}(k)}{a^{MF}(k)/r(k+1)+1};$$
$$p_4^{MF}(k+1) = \frac{c^{MF}(k)[a^{MF}(k)/r(k+1)+1] - b^{MF}(k)^2/r(k+1)}{a^{MF}(k)/r(k+1)+1}. \tag{4.21}$$

(ii) Use (4.17) to prove this. ■

4.2.2 SF Method

Claim 2 *Suppose the initial error covariance matrix $P_i(0|0)$ for the i-th sensor is $\Phi_i(0)$-block diagonal with arbitrary $\Phi_i(0)$. Then, with SF, the following are true:*

(i) The error covariance matrix $P_i(k|k)$ of the KF associated with the i-th sensor is $\Phi_i^{SF}(k)$-block diagonal with

$$\Phi_i^{SF}(k) = \begin{bmatrix} p_{1i}^{SF}(k) & p_{2i}^{SF}(k) \\ p_{2i}^{SF}(k) & p_{4i}^{SF}(k) \end{bmatrix}.$$

(ii) The fused error covariance matrix $P(k|k)$ is $\Phi^{SF}(k)$-block diagonal with

$$\Phi^{SF}(k) = \begin{bmatrix} p_1^{SF}(k) & p_2^{SF}(k) \\ p_2^{SF}(k) & p_4^{SF}(k) \end{bmatrix}.$$

(iii) The utility $S(k+1)$ can be expressed as

$$S^{SF}(k+1) = \frac{-2\,\rho_1(k)}{\rho_1(k)/r(k+1) + \rho_1(k)\,\rho_4(k) - \rho_2(k)^2}.$$

The entries of $\Phi_i^{SF}(k)$, $\Phi^{SF}(k)$ and $S^{SF}(k+1)$ above appear in the proof.

Proof: By induction. The claim is trivially true for $k = 0$ since $P_i(0|0)$ is $\Phi_i(0)$-block diagonal. Suppose the claim is true for k.

(i) Given that $P_i(k|k)$ is $\Phi_i^{SF}(k)$-block diagonal with entries given by item (i) of Claim 2, substitute $P_i(k|k)$ into (4.9) to show that $P_i(k+1|k)$ is $B_i^{SF}(k)$-block diagonal with

$$B_i^{SF}(k) = \begin{bmatrix} a_i^{SF}(k) & b_i^{SF}(k) \\ b_i^{SF}(k) & c_i^{SF}(k) \end{bmatrix},$$

where

$$\begin{aligned} a_i^{SF}(k) &= p_{1i}^{SF}(k) + 2\,p_{2i}^{SF}(k)\,\Delta + p_{4i}^{SF}(k)\,\Delta_i^2 + \frac{q}{4}\,\Delta^4; \\ b_i^{SF}(k) &= p_{2i}^{SF}(k) + p_{4i}^{SF}(k)\,\Delta + \frac{q}{2}\,\Delta^3; \\ c_i^{SF}(k) &= p_{4i}^{SF}(k) + q\,\Delta^2. \end{aligned} \tag{4.22}$$

We note that $R_i(k+1) = diag\{r_i(k+1), r_i(k+1)\}$. Substitute $P_i(k+1|k)$ and $R_i(k+1)$ into (4.8) to show that $P_i(k+1|k+1)$ is $\Phi_i^{SF}(k+1)$-block diagonal with

$$\Phi_i^{SF}(k+1) = \begin{bmatrix} p_{1i}^{SF}(k+1) & p_{2i}^{SF}(k+1) \\ p_{2i}^{SF}(k+1) & p_{4i}^{SF}(k+1) \end{bmatrix}, \tag{4.23}$$

where

$$\begin{aligned} p_{1i}^{SF}(k+1) &= \frac{a_i^{SF}(k)}{a_i^{SF}(k)/r_i(k+1)+1}; \\ p_{2i}^{SF}(k+1) &= \frac{b_i^{SF}(k)}{a_i^{SF}(k)/r_i(k+1)+1}; \\ p_{4i}^{SF}(k+1) &= \frac{c_i^{SF}(k)[a_i^{SF}(k)/r_i(k+1)+1] - b_i^{SF}(k)^2/r_i(k+1)}{a_i^{SF}(k)/r_i(k+1)+1}. \end{aligned} \tag{4.24}$$

(ii) First, use (4.23) to show that $P_i^{-1}(k+1|k+1)$ is $\overline{\Phi}_i^{SF}$-block diagonal with

$$\overline{\Phi}_i^{SF} = \begin{bmatrix} \frac{c_i^{SF}(k)}{a_i^{SF}(k)c_i^{SF}(k)-b_i^{SF}{}^2} + \frac{1}{r_i(k+1)} & \frac{-b_i^{SF}}{a_i^{SF}(k)c_i^{SF}(k)-b_i^{SF}{}^2} \\ \frac{-b_i^{SF}}{a_i^{SF}(k)c_i^{SF}(k)-b_i^{SF}{}^2} & \frac{a_i^{SF}}{a_i^{SF}(k)c_i^{SF}(k)-b_i^{SF}{}^2} \end{bmatrix}.$$

Substitute in (4.16) to show that $P(k+1|k+1)$ is $\Phi^{SF}(k+1)$-block diagonal with

$$\Phi^{SF}(k+1) = \begin{bmatrix} p_1^{SF}(k+1) & p_2^{SF}(k+1) \\ p_2^{SF}(k+1) & p_4^{SF}(k+1) \end{bmatrix}, \tag{4.25}$$

where

$$p_1^{SF}(k+1) = \frac{\rho_1(k)}{\rho_1(k)/r(k+1) + [\rho_1(k)\,\rho_4(k) - \rho_2(k)^2]};$$
$$p_2^{SF}(k+1) = \frac{\rho_2(k)}{\rho_1(k)/r(k+1) + [\rho_1(k)\,\rho_4(k) - \rho_2(k)^2]};$$
$$p_4^{SF}(k+1) = \frac{1/r(k+1) + \rho_4(k)}{\rho_1(k)/r(k+1) + [\rho_1(k)\,\rho_4(k) - \rho_2(k)^2]},$$

and

$$\rho_1(k) = \sum_{i=1}^{N} \frac{a_i^{SF}(k)}{\delta_i(k)}; \quad \rho_2(k) = \sum_{i=1}^{N} \frac{b_i^{SF}(k)}{\delta_i(k)};$$
$$\rho_4(k) = \sum_{i=1}^{N} \frac{c_i^{SF}(k)}{\delta_i(k)}; \quad \delta_i(k) = a_i^{SF}(k)\,c_i^{SF}(k) - b_i^{SF}(k)^2.$$

(iii) Use (4.17) to prove this. ■

Claims 1 and 2 provide closed form expressions for the utility functions $S^{MF}(k+1)$ and $S^{SF}(k+1)$, respectively. In fact, these expressions for the utilities can be expressed in a unified manner as follows:

$$S(k+1) = \frac{-2}{1/r(k+1) + A(k)}, \tag{4.26}$$

where

$$A(k) = \begin{cases} 1/a^{MF}(k), & \text{for MF;} \\ [\rho_1(k)\,\rho_4(k) - \rho_2(k)^2]/\rho_1(k), & \text{for SF.} \end{cases} \tag{4.27}$$

Next, we have

Claim 3 *For all $k > 0$, the following are true:*

(i) For MF, if $P(0|0) = I$, then $a^{MF}(k)\,c^{MF}(k) - b^{MF}(k)^2 > 0$ and $p_\ell \geq 0$, $\ell = 1, 2, 4$.

(ii) For SF, if $P_i(0|0) = I$, $\forall i$, then $\rho_1(k)\,\rho_4(k) - \rho_2(k)^2 > 0$ and $p_{\ell i}^{SF}(k) \geq 0$, $\ell = 1, 2, 4$.

Proof: See Appendix A. ■

So, for both MF and SF, Claim 3 identifies conditions that guarantee positivity of $A(k)$, $\forall k > 0$.

4.2.3 Properties of Utility

To find the set of optimal transmission rates that maximize the utility $S(k+1)$ in a shared network, one must capitalize upon the relationship between $S(k+1)$ and the sensor transmission rates. This relationship should be formulated through the sensor measurement error covariance $r_i(k+1)$ values since $S(k+1)$ is a function of the $r_i(k+1)$ values which in turn depend on the sensor transmission rates. Consequently, the next step is to formulate this relationship by modeling the $r_i(k+1)$ values in terms of the sensor transmission rates.

As mentioned earlier, the measurement error covariance term $r_i(k+1)$ consists of two main components: the first component is due to the sensor proximity error; and the second component is due to the finite bit-rate error. Let us assume that f_i and $\gamma_i(k+1)$ correspond to the transmission bit-rate and the sensor proximity error covariance of the i-th sensor, respectively. The model of $r_i(k+1)$ should be motivated by several issues:

- As $f_i \to \infty$, we need $r_i(k+1) \to \gamma_i(k+1)$: This is important because, if the sensor has infinite transmission rate, then the measurement error covariance consists of only the proximity error covariance.

- As $f_i \to 0$, we need $r_i(k+1) \to \infty$: If the sensor transmission rate is zero, then nothing is transmitted. Hence the measurement error covariance should become infinite.

• $r_i(k+1)$ should be a decreasing function of f_i: This is because the accuracy of the reading should increase as more data is transmitted.

• $S(k+1)$ should be a concave function of $\mathbf{f}$ (the vector of all the sensor bit-rates): This requirement is necessary to ensure a global maximum for the utility.

Based on these requirements, we use the following model for the measurement error covariance for sensor i:

$$r_i(k+1) = \gamma_i(k+1) + \frac{d}{f_i}, \ \forall i = 1, \ldots, N, \tag{4.28}$$

where $d > 0$ is a constant. The second term d/f_i in (4.28) represents the component of the measurement error covariance consisting of the finite bit-rate approximation of the sensor measurement. One may of course choose alternate functions corresponding to this component (e.g., the rate distortion function of a Gaussian random variable); however, as Lemma 1 below establishes, (4.28) guarantees the concavity of $S(k+1)$.

Lemma 1 *Given the measurement error covariance in (4.28), the utility $S(k+1)$ (for both SF and MF) is concave with respect to (w.r.t.) $\mathbf{f}$.*

Proof: See Appendix B. ■

Having the utility $S(k+1)$ as a concave function of the sensor transmission rates $\mathbf{f}$ enables us to determine the set of optimal sensor transmission rates $\mathbf{f}^*$ that maximizes $S(k+1)$:

$$\mathbf{f}^* = \arg \max_{\mathbf{f} \geq 0} S(k+1). \tag{4.29}$$

Suppose, at time k, the target tracking fusion center must select the bit-rates to maximize the target tracking utility for the next iteration, i.e., $S(k+1)$. Note that $S(k+1)$ is a function of $r_i(k+1)$ which we have modeled via (4.28) as a function of $\gamma_i(k+1)$ and f_i, for all i. Since $\gamma_i(k+1)$ is not known at time k, we assume

$\gamma_i(k+1) \approx \gamma_i(k)$ to obtain the following estimates for $\hat{r}_i(k+1)$ and $\hat{S}(k+1)$:

$$\frac{1}{\hat{r}(k+1)} = \sum_{i=1}^{N} \frac{1}{\hat{r}_i(k+1)}, \text{ with } \hat{r}_i(k+1) = \gamma_i(k) + \frac{d}{f_i};$$
$$\hat{S}(k+1) = \frac{-2}{1/\hat{r}(k+1) + A(k)}, \tag{4.30}$$

where $A(k)$ is as in (4.27).

To see how $\hat{S}(k+1)$ affects the sensor bit-rate allocations, consider a situation where N sensors transmit their packetized data over a single dedicated link. The aggregate transmission rate being constrained by the link capacity C, the sensor bit-rates must then satisfy $\sum_{i=1}^{N} f_i \leq C$. Let us examine where $\hat{S}(k+1)$ achieves its maximum:

- *When* $\gamma_i(k) = \gamma(k),\ \forall i = 1, \ldots, N$*:* When all $\gamma_i(k)$ values are identical, all the sensors have identical proximity error covariances and hence they are equally important. Hence, the sensor rates f_i are identical.

- *When* $\gamma_i(k) \in (0, \infty),\ \forall i = 1, \ldots, N$*:* A sensor with a lower $\gamma_i(k)$ possesses a higher accuracy, and consequently the sensor will receive a higher f_i value.

- *When* $\gamma_i(k) = 0$ *for some sensors:* A $\gamma_i(k) = 0$ indicates perfect measurements, and non-zero f_i values will be allocated to only these sensors.

- *When* $\gamma_i(k)$ *is infinite for some sensors:* An infinite $\gamma_i(k)$ indicates completely corrupted and, hence, useless measurement, and corresponding sensors will be allocated zero f_i values.

4.2.4 Incorporation of Target Tracking Requirements

Given the NUM framework for the analysis and design of rate controllers, how does one include target tracking QoS requirements into this framework? One standard

approach is to modify the utility function of the sources to accommodate those requirements. However, as we mentioned earlier, care must be taken to ensure that the re-formulated problem and its solution *does not entail the re-design of networks.* That is, we assume no minimum bit-rate guarantees, and no special or additional active queue management feedback from routers, and also assume that the rate/congestion control is end-to-end. Therefore, to accommodate the target tracking QoS utility, we work within Kelly's original framework and propose to augment the standard bit-rate utility function with an additive term reflecting target tracking QoS.

First, let us note that there are basically two types of sources/users on the network: (1) *target tracking users* who are running the target tracking application; and (2) *ordinary users* who are running ordinary data transfer applications. Now consider the following new utility function that all network users use:

$$U_j(\mathbf{f}) = V_j(f_j) + K_j \hat{S}(k+1), \ \forall j, \tag{4.31}$$

where $K_j \geq 0$ is a real parameter. Observe the following regarding (4.31). The first term $V_j(f_j)$ is a concave function of the source data rate f_j. This term addresses rate maximization, and is from the standard (original) utility function. The second term addresses the target tracking QoS requirements since $\hat{S}(k+1)$ is the quantity we should maximize to maximize the target tracking QoS (see Section 4.2). The parameter K_j determines the emphasis placed upon target tracking QoS over the standard utility regarding rate maximization; $K_j = 0$ whenever source j is an ordinary source (i.e., not a target tracking sensor source). Therefore, for ordinary sources the utility function reduces to the original rate maximization utility function.

Therefore, for the target tracking users, the utility function in (4.31) reflects the twin goals of raw data rate maximization, and the attainment of the target tracking QoS requirements. Moreover, for ordinary sources, (4.31) reduces to the standard

data transfer utility function. Now recall that Lemma 1 establishes the concavity of $\hat{S}(k+1)$ w.r.t. $\mathbf{f}$. Moreover, we assume that $V_j(f_j)$ is a typical rate maximization utility and consequently is concave. Therefore, since $K_j \geq 0$, the new utility (4.31) is also guaranteed to be concave, is a requirement in the NUM framework. We may now express the network flow utility maximization problem as

$$\max_{\mathbf{f}\geq 0} \sum_j U_j(\mathbf{f}) \quad \text{subject to} \quad \sum_j R_{j\ell} f_j \leq C_\ell, \ \forall \ell \in \mathfrak{L}, \tag{4.32}$$

where $\mathfrak{L}$ is a set that indexes all the links in the network. Since $U_j(\mathbf{f})$ is a concave function of $\mathbf{f}$, the sum $\sum_j U_j(\mathbf{f})$ is also a concave function, leading to a convex optimization problem. To solve this optimization problem, we take the primal-dual approach [Ber99, LPW02, Low03], and consider the Lagrangian

$$\begin{aligned} L(\mathbf{f}, \mathbf{p}) &= \sum_j U_j(\mathbf{f}) + \sum_\ell p_\ell \left(C_\ell - \sum_j R_{j\ell} f_j \right) \\ &= \sum_j \left(U_j(\mathbf{f}) - f_j \sum_\ell R_{j\ell} p_\ell \right) + \sum_\ell C_\ell p_\ell. \end{aligned} \tag{4.33}$$

Here, $\mathbf{p}$ is a column vector of Lagrange multipliers. In the NUM framework, $\mathbf{p}$ is construed as containing the link 'prices' $p_\ell, \ \ell \in \mathfrak{L}$, i.e., p_ℓ is the price per unit bandwidth that link ℓ charges to any source using link ℓ [KMT98, Ber99, LL99, LPW02]. Recall that $R_{j\ell} = 1$ if source j uses link ℓ and it is 0 otherwise. The sum $\sum_\ell R_{j\ell} p_\ell$ is the sum of all the prices that source j incurs by using its particular links in the network (recall that the routing is assumed to be fixed). Then $\sum_\ell R_{j\ell} p_\ell$ is the total price charged by the network to source j for transmitting with rate f_j.

So, the original constrained optimization problem has been converted into a Lagrangian dual problem with a duality gap of zero [KMT98, Ber99, LL99, LPW02]. The solution to this Lagrangian dual problem is achieved by maximizing the Lagrangian:

$$\mathbf{f}^* = \arg \max_{\mathbf{f}\geq 0} L(\mathbf{f}, \mathbf{p}^*), \tag{4.34}$$

where $\mathbf{p}^*$ is the column vector of optimal Lagrange multipliers. To achieve this, we compute the partial derivatives of the Lagrangian $L(\mathbf{f}, \mathbf{p})$ w.r.t. f_m:

$$\frac{\partial}{\partial f_m} L(\mathbf{f}, \mathbf{p}) = \frac{\partial}{\partial f_m} \left(\sum_j U_j(\mathbf{f}) \right) - \sum_\ell R_{m\ell} p_\ell. \tag{4.35}$$

4.2.5 TCP Reno Compatible Rate Control

Noting that TCP Reno is perhaps the most widely used TCP protocol in the current internet [KR04], we first develop the proposed algorithm in a way that it is compatible with flows that implement TCP Reno. With this in mind, we start by assuming that $V(f_j)$ is the TCP Reno utility function. Note that we can use the utility functions of any other rate control TCP variant, as long as it is a concave function. Now $V(f_j)$ for TCP Reno can be written as [Low03]

$$V(f_j) = \frac{\sqrt{2}}{RTT_j} \arctan\left(\frac{f_j RTT_j}{\sqrt{2}} \right), \tag{4.36}$$

where RTT_j is the RTT of source j. To proceed, let us take $K_j \equiv K$ for all target tracking sources and recall that $K_j = 0$ for all ordinary sources. Then, (7.5) reduces to

$$\frac{\partial}{\partial f_m} L(\mathbf{f}, \mathbf{p}) = \frac{2}{2 + f_m^2 RTT_m^2} - q_m + \xi_m(\mathbf{f}). \tag{4.37}$$

Here, $I_m = 1$ if source m is a target tracking source and 0 otherwise, and

$$\begin{aligned} q_m &= \sum_{\ell \in \mathfrak{L}} R_{m\ell} p_\ell; \\ \xi_m(\mathbf{f}) &= I_m N K \frac{\partial \hat{S}(k+1)}{\partial f_m} \\ &= -2 I_m N K \frac{\partial \hat{r}_m(k+1)/\partial f_m}{\left[\left(1/\hat{r}(k+1) + A_m(k)\right) \hat{r}_m(k+1) \right]^2}. \end{aligned} \tag{4.38}$$

Here,

$$\hat{r}_m(k+1) = \gamma_m(k) + \frac{d}{f_m}; \;\; \frac{\partial}{\partial f_m} \hat{r}_m(k+1) = -\frac{d}{f_m^2}, \tag{4.39}$$

where q_m is the total price incurred by source m for using the network. For TCP Reno, q_m is the marking/loss probability observed by source m.

Transmission Rate Update

The convex optimization problem in (4.32) can be solved and the optimum (7.4) can be achieved via the gradient projection approach [Ber99]. As shown in [Ber99], the gradient projection algorithm enables one to achieve the optimum $(\mathbf{f}^*, \mathbf{p}^*)$ in an iterative manner by updating the transmission rates of individual sources in a distributed manner. This iterative update has a period (i.e., the RTT) which is not equal to the period of the KF update. The transmission rate update iterations are indexed by t. In a networked setting, this allows for the distributed iterative solution of the optimization problem without explicit communication among the different sources.

The gradient projection can be used to get the following rate update function.

$$f_m(t+1) = \Big[f_m(t) + s\Big(\frac{2}{2 + f_m(t)^2 RTT_m(t)^2} - q_m(t) - \xi_m(\mathbf{f}(t))\Big)\Big]^+. \qquad (4.40)$$

Note that, to calculate $\xi_m(\mathbf{f})$, the f_i and $\gamma_i(k)$ values for all the sensors are required. In the multi-sensor target tracking application, only the sink node, where the KF based algorithms are used to calculate the target track, would have access to this information. Hence, $\xi_m(\mathbf{f})$ is the scalar feedback information sent by the sink node to the sensor source m. Note that, $\xi_m(\mathbf{f}) = 0$ whenever $K = 0$, i.e., additional feedback is unnecessary for the ordinary sources. Furthermore, in TCP Reno, the total price q_m is the overall marking or dropping probability observed by source m. This q_m value can be calculated by counting the number of received and marked packet over a fixed period of time. Therefore, a target tracking source updates its rate to $f_m(t+1)$ using its current rate $f_m(t)$, the current marking probability q_m, and the rate coordination feedback $\xi_m(\mathbf{f}(\mathbf{t}))$ from the sink node as in (4.40). None of the sources need to know

the rates of any other source. The sink node needs to keep track of the individual rates of the sources in the sensor group, and provide per-flow feedback to each source.

Window Flow Controller Update

To express the rate update in (4.40) as a window update function, note that $f_m(t) = w_m(t)/RTT_m(t)$, where $f_m(t)$ and $w_j(t)$ are the transmission rate and window size of source m, respectively (one can easily convert this window size from bits to packets as in standard window flow control). Then, (4.40) yields the ideal window update function as

$$w_m(t+1) = RTT_m(t+1)\left[\frac{w_m(t)}{RTT_m(t)} + s\left(\frac{2}{2+w_m(t)^2} - q_m(t) - \xi_m(\mathbf{f}(t))\right)\right]^+. \quad (4.41)$$

However, since $RTT_m(t+1)$ is not available at the time of the window update, for implementation purposes, we use the approximation $RTT_m(t+1) \approx RTT_m(t)$ which yields

$$w_m(t+1) = \Big[w_m(t) + s\,RTT_m(t)\Big(\frac{2}{2+w_m(t)^2} - q_m(t) - \xi_m(\mathbf{f}(t))\Big)\Big]^+. \quad (4.42)$$

4.2.6 TCP Vegas Compatible Rate Control

The above rate control algorithm can be easily modified to work with flows that implement TCP Vegas. For TCP Vegas, the utility function is $V(f_j) = \alpha \log(f_j)$ [LPW02]. When the gradient projection algorithm is used to solve the corresponding problem, we obtain the rate update function as

$$f_m(t+1) = \Big[f_m(t) + s\Big(\frac{\alpha}{f_m(t)} - q_m(t) - \xi_m(\mathbf{f}(t))\Big)\Big]^+. \quad (4.43)$$

The corresponding window flow controller update is

$$w_m(t+1) = \Big[w_m(t) + s\,RTT_m(t)\Big(\frac{\alpha\,RTT_m(t)}{w_m(t)} - q_m(t) - \xi_m(\mathbf{f}(t))\Big)\Big]^+. \quad (4.44)$$

The total price $q_m(t)$ in this window update function is the total queueing delay experienced by packets sent by source m. This quantity is estimated at the source as the difference between the current RTT and the minimum observed RTT [LPW02, BP95]. The scalar feedback of this version is same as the feedback in the TCP Reno version.

Table 4.1 summarizes the above TCP Reno and TCP Vegas compatible target tracking rate control algorithms and the functions of the network and various nodes and flows w.r.t. the target tracking flows; the ordinary flows use TCP Reno or Vegas (as the case may be).

Table 4.1: Target Tracking Rate Control Algorithms: Summary

	TCP Reno Version	TCP Vegas Version
Network	Update marking probability	Update queuing delay
Sink Node	Periodically (KF update is higher) calculate $a_i^{SF}(k)$, $b_i^{SF}(k)$ and $c_i^{SF}(k)$ for SF, and $a^{MF}(k)$, $b^{MF}(k)$ and $c^{MF}(k)$ for MF.	Periodically (KF update is higher) calculate $a_i^{SF}(k)$, $b_i^{SF}(k)$ and $c_i^{SF}(k)$ for SF, and $a^{MF}(k)$, $b^{MF}(k)$ and $c^{MF}(k)$ for MF.
	Calculate $\xi_m(\mathbf{f})$ as each packet is received and send it with the ACK packet.	Calculate $\xi_m(\mathbf{f})$ as each packet is received and send it with the ACK packet.
Source	Update rate using (4.40) or window using (4.42) as each ACK is received.	Update rate using (4.43) or window using (4.44) for every RTT.

4.3 Simulations

In this section, we illustrate the application and performance of the rate coordination protocol in a multi-sensor target tracking application.

4.3.1 Environment

The simulated test area of size 4000×4000 ft^2 has 16 equally spaced sensors sending readings to a single sink node that acts as the decision center. The experiment we

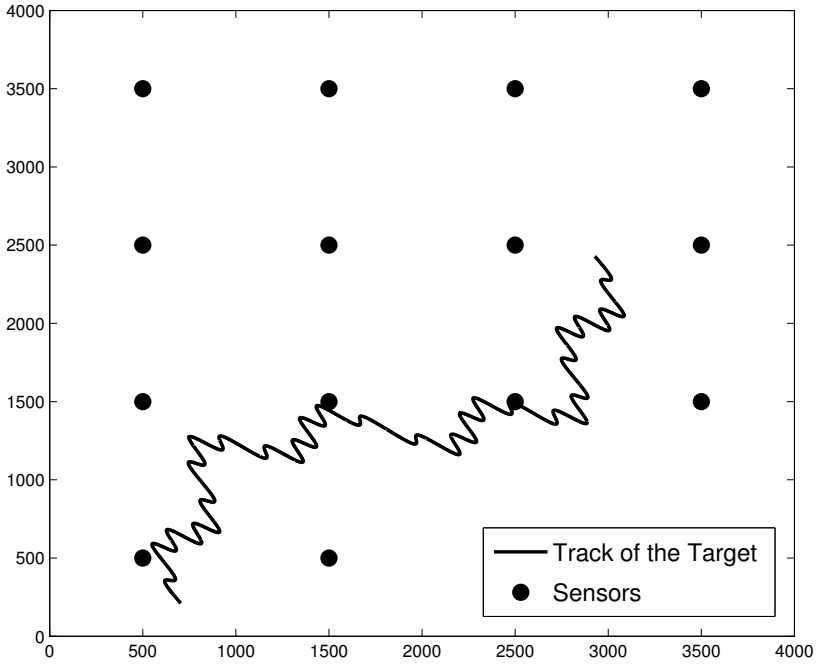

Figure 4.1: Test area: sensor arrangement and tracks.

conducted involved a moving target with pre-determined path in the test area. This path is shown in Fig. 4.1. Some comments regarding the sensor layout and the target path that we utilize for this set of experiments are in order:

- An equally spaced sensor layout ensures that the simulated area receives uniform coverage from the sensors. Although a more realistic scenario may have sensor nodes that are more unevenly spaced, this uniform spacing that we utilize provides valuable insight into the performance of our proposed algorithm. As the target moves in the pre-determined path, we can guarantee that the significance of sensors changes over time because of the equally spaced sensor

layout. These changes in the sensor significance necessitate an effective rate allocation algorithm.

- The pre-determined target path is composed of a summation of several sinusoids thus incorporating both slow and rapid changes of the direction of travel. Such slow and rapid changes are more reflective of a target movement. The need for effective rate allocation becomes more apparent when the target moves more rapidly.

4.3.2 Sensors

The Sensors are assumed to (a) measure the x and y positions of the target; (b) produce readings of a constant size in bytes; (c) possess the capability to transmit readings at a variable rate depending on the allocated bandwidth (if 2 readings per second can be transmitted with the allocated bandwidth of f, then 4 readings per second can be transmitted with the allocated bandwidth of $2f$; and (d) feature readings whose accuracy, and hence $\gamma_i(k)$, are proportional to the square of the distance between the target and the particular sensor. The layout of the sensors in the test area is shown in Fig. 4.1.

4.3.3 Network

The sensor readings are transmitted to the sink node via a network with the topology shown in Fig. 4.2. This topology is selected such that it is possible to examine the effect of the rate allocation algorithm on ordinary data transfer flows. This topology has ordinary data transfer flows with varying number of hops in order to simulate the effect of target tracking flows on ordinary data transfer flows and

vice versa. Target tracking flows are selected such that the number of hops are not identical in order to simulate a more realistic network setting. In a multi-sensor target tracking application, it is usual to have a link shared by all target tracking flows. We have simulated this scenario by selecting the topology in a way that all the 16 flows share a common link.

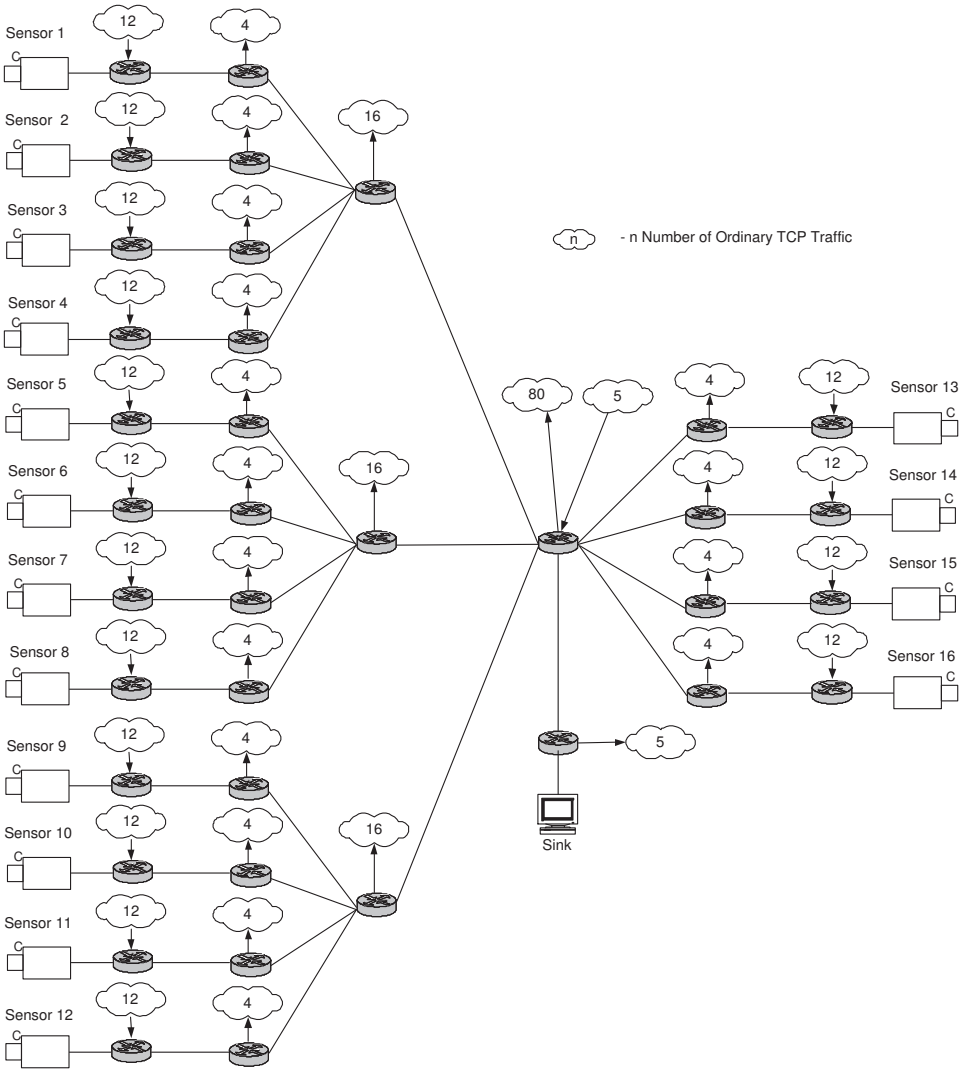

Figure 4.2: Network topology for target tracking application.

The flow from the sensor #n to the sink is identified as flow #n. Flows #1-12 go via 5 intermediate routers while flows #13-16 go via 4 intermediate routers. Each link is congested with ordinary data transfer traffic (either TCP Reno or TCP

Vegas depending on the simulation case). In Fig. 4.2, the 'cloud' next to each router identifies the exact number of incoming and outgoing ordinary data transfer traffic flows at that router. Ordinary data transfer flows start and stop transmission at random times.

4.3.4 Sink Node/Decision Center

The sink employs a KF-based algorithm to estimate the track of the target, relying upon either MF or SF to combine the readings from different sensors. The KF-based algorithms utilize an update period of $\Delta = 0.01$s. In the absence of any new sensor readings, the KF-based methods utilize the common Zero Order Hold assumption. In addition, the sink node sends the per-flow feedback $\xi_j(\mathbf{f})$ to source j with every ACK packet sent to source j.

4.3.5 Rate Allocation

The sensors send fixed size readings at a higher (or lower) rate if they have a higher (or lower) bandwidth allocated. In fact, the reading transmission frequency is directly related to the allocated bit-rate since the reading size is a constant. In the proposed new algorithm, this bit-rate allocation is determined by the window update functions given in (4.42) and (4.44).

4.3.6 Simulation Setups

Four simulation setups were considered in the simulation. All simulations were carried out in the ns-2 network simulation environment [ns2].

MFR: MF with TCP Reno

In this simulation setup, the sink node/decision center uses the MF method for combining the sensor measurements. The ordinary data transfer flows always use the Internet standard TCP Reno. All of the routers in the network employ RED AQM. The target tracking sensors transmit their measurements using either ordinary TCP Reno, or the new method given by (4.42). Two scenarios were considered:

- Using TCP Reno (MFR Reno): All flows (both sensor and ordinary flows) run the standard Internet protocol TCP Reno. This provides a target tracking performance baseline for MF that a standard network protocol can provide.

- Using the proposed protocol (MFR New): The target tracking flows run the new protocol with $K = 400$ and $s = 1$; the ordinary data transfer flows still employ TCP Reno. The sensors update their window sizes (and hence their transmission rates) every RTT (roughly every 200ms). The time-varying $a^{MF}(k)$ quantity in the utility function is updated at a slower rate at the sink node/decision center in its calculation of the per-flow scalar feedback $\xi_m(\mathbf{f})$ provided to each sensor. At the sensors, the marking probability is reconstructed by calculating the number of marked packets over a one second period of time.

SFR: SF with TCP Reno

These simulations parallel the MFR Reno, and MFR New cases in 4.3.6, except that now the sink node/decision center utilizes the SF method for computing the target track.

- SFR Reno: A baseline for SF in which all flows utilize the standard TCP Reno.

- SFR New: The target tracking flows run the new protocol except $K = 800$ is used. The utility terms $a_i^{SF}(k)\, b_i^{SF}(k)\, c_i^{SF}(k)\, \forall i$ are updated at a slower rate than the window update.

MFV: MF with TCP Vegas

This simulation is similar to the MFR case in that MF is used by the sink node/decision center to combine the sensor data and compute the target track. However, the ordinary data transfer flows utilize TCP Vegas instead of TCP Reno. Two scenarios were considered:

- Using TCP Vegas (MFV Vegas): All flows (both sensor and ordinary data transfer) run the data transfer protocol TCP Vegas. This gives a baseline performance for MF utilizing TCP Vegas.

- Using the proposed protocol (MFV New): The target tracking flows run the new protocol given by (4.44) with $K = 400$ and $s = 25$; the ordinary data transfer flows still employ TCP Vegas.

SFV: SF with TCP Vegas

This simulation parallels the ones done in 4.3.6, except that SF is used by the decision center/sink node to combine the sensor data. Two scenarios are considered.

- Using TCP Vegas (SFV Vegas): All flows (both sensor and ordinary data transfer) run TCP Vegas. This gives a baseline performance for SF utilizing TCP Vegas.

- Using the new protocol (SFV New):Now the target tracking flows run the new protocol given by (4.44) with $K = 800$ and $s = 25$; the ordinary data transfer flows still employ TCP Vegas.

4.3.7 Results

For the MFR case, the reconstructed tracks for both the baseline MFR Reno, and the new protocol MFR New, together with the actual target track are given in Fig. 4.3. This figure only shows a (500×400) ft^2 section of the test area for the target. These results represent the performance within other parts of the test area adequately well. In this figure, one can see that when the MF-based target tracking application relies upon the standard TCP Reno for dictating the sensor transmission rates (the MFR Reno case), the target tracking performance suffers. In contrast, when the sensors use the new protocol (MFR New), the target tracking application is able to achieve a much more accurate track. This is due to the fact that the new protocol augments the standard TCP Reno utility with a target tracking QoS term that allows for the more relevant sensors to get additional bit-rate. The corresponding simulation results for the *SFR*, *MFV* and *SFV* cases are given in Figs. 4.4, 4.5 and 4.6, respectively.

In the figures, one can see that the target tracking performance provided by the new protocols (the *MFR New*, *SFR New*, *MFV New* and *SFV New* cases) is superior to the target tracking performance provided by standard "off-the-shelf" TCP protocols (the *MFR Reno*, *SFR Reno*, *MFV Vegas* and *SFV Vegas* cases). When we consider reconstructed tracks provided by ordinary data transfer protocols, we note that the sink node/decision center fails to estimate sharp changes in the path since it does not have enough information. On the other hand, the proposed protocols are

capable of providing bit-rates that allow the sink node/decision center to estimate these sharp changes in a much better manner.

In order to do a numerical evaluation, the Mean Square Error (MSE) values for the estimated tracks are calculated. These calculated MSE values for all four simulation setups are given in Table 4.2. Note that the MSE values are given for different K values and it is clear from the results that the MSE decreases as the K value increases. This is appropriate since increasing K value means putting more emphasis on the target tracking QoS maximization, and hence the accuracy of the estimated track increases.

Table 4.2: MSE Values for Estimated Tracks

MFR New		SFR New		MFV New		SFV New	
K	MSE	K	MSE	K	MSE	K	MSE
25	65.60	50	77.89	25	60.52	50	70.10
50	60.72	100	67.88	50	57.79	100	68.23
100	49.63	200	61.36	100	55.28	200	67.41
200	44.14	400	51.30	200	49.57	400	62.24
400	38.67	800	44.86	400	43.66	800	56.14
MFR Reno		SFR Reno		MFV Vegas		SFV Vegas	
K	MSE	K	MSE	K	MSE	K	MSE
0	72.28	0	78.86	0	88.76	0	88.19

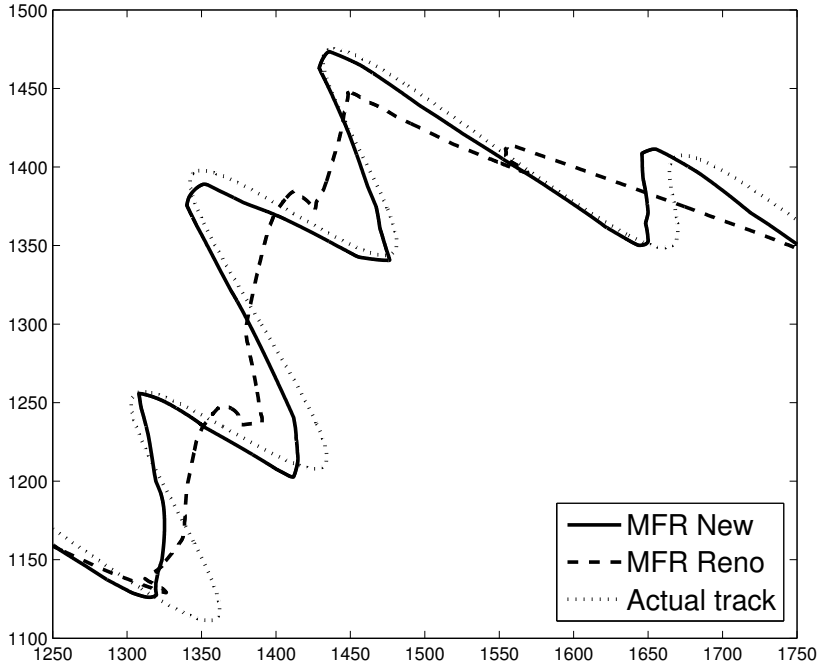

Figure 4.3: Reconstructed tracks for MFR.

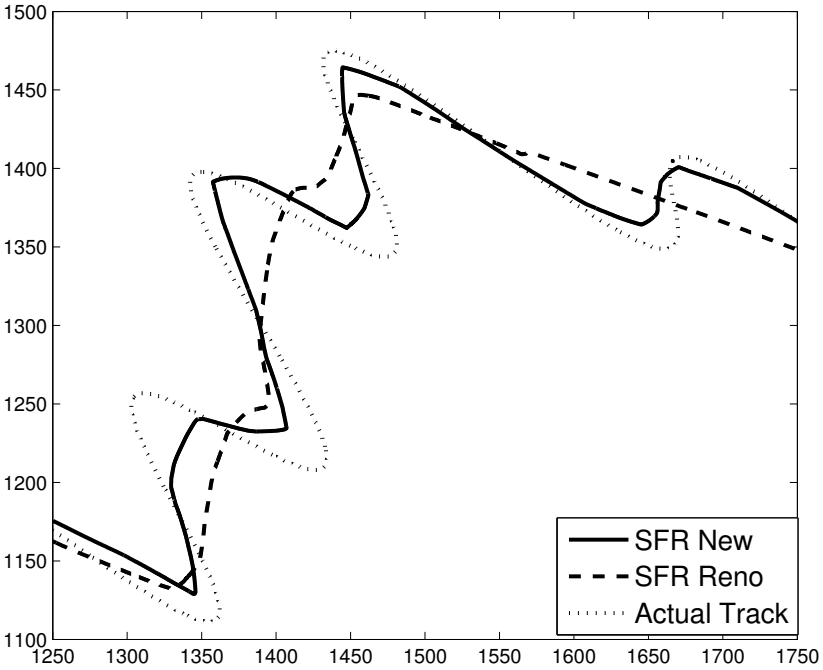

Figure 4.4: Reconstructed tracks for SFR.

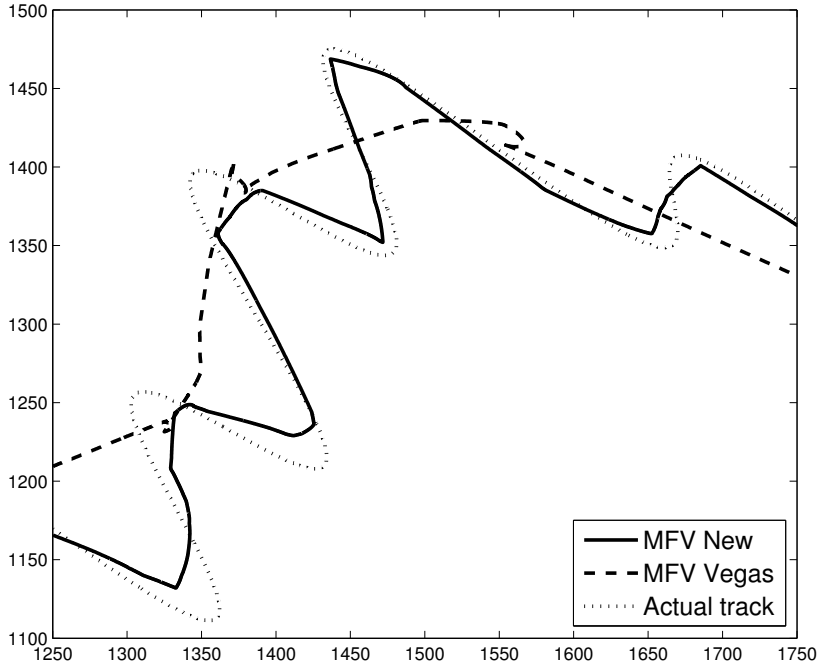

Figure 4.5: Reconstructed tracks for MFV.

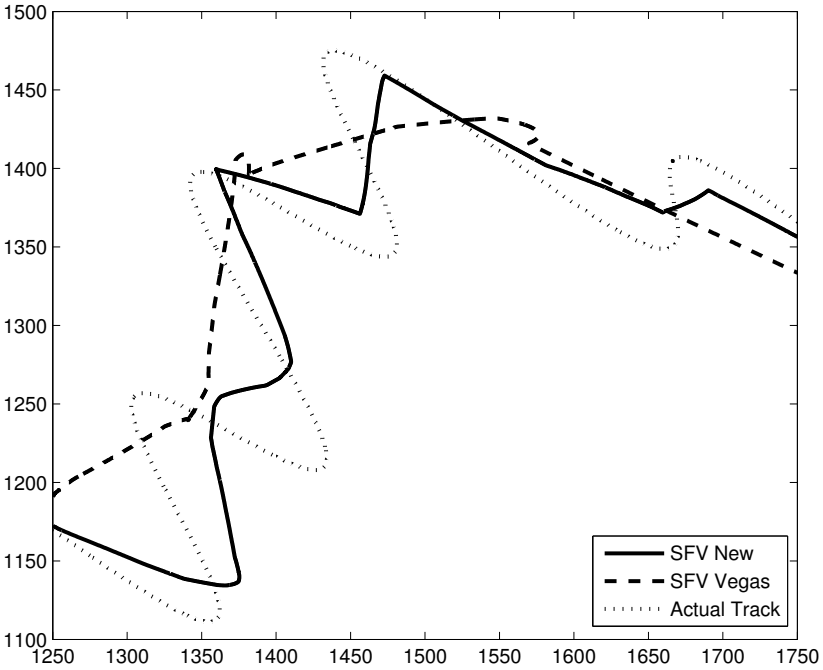

Figure 4.6: Reconstructed tracks for SFV.

4.3.8 Impact on Ordinary Data Transfer Flows

How does the proposed protocol impact the ordinary data transfer flows that are not members of the target tracking application? To study this, we compared the baseline bit-rates that sources utilizing standard TCP mechanisms get in MFR Reno, SFR Reno, MFV Vegas, and SFV Vegas cases, with the new bit-rates they get in MFR New, SFR New, MFV New, and SFV New cases, respectively. Fig. 4.7 provides the average percentage *gain* that the ordinary data transfer sources achieve over their baseline rates when the new protocol is used by the target tracking sources.

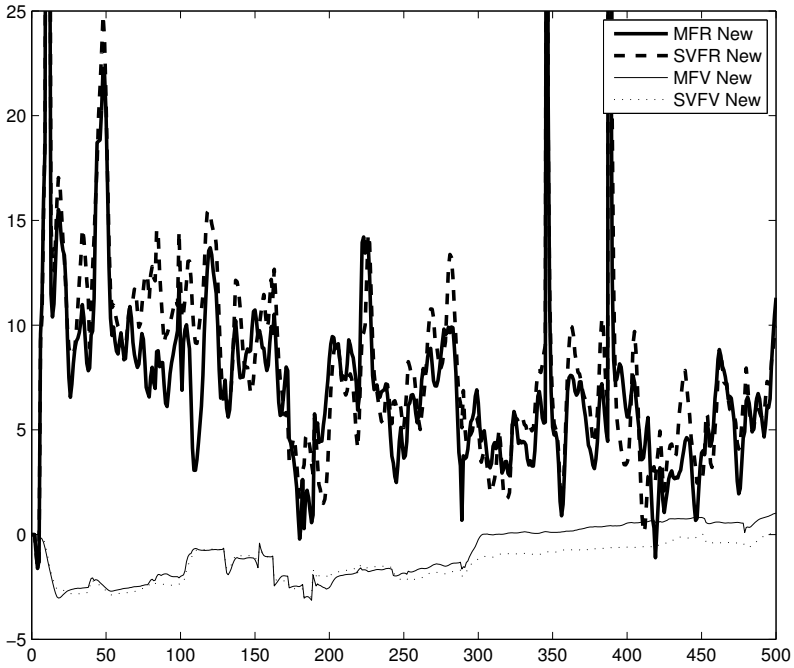

Figure 4.7: Average percentage of gain of ordinary data transfer flows.

These results show that, for the MFR New and SFR New cases, the ordinary TCP Reno sources actually gain in terms of bit-rate, on average, over their nominal baseline rates. For the Vegas case, the average gain is slightly negative. The average gains for all four simulation setups with different K values are given in Table 4.3. In general,

the new protocol, on average, does not adversely affect the standard TCP Reno flows, while the TCP Vegas flows are only slightly adversely affected. By balancing the rate maximization utility with the target tracking utility, the new protocol is able to gain better target tracking performance without causing the rest of the network to suffer from congestion collapse.

Table 4.3: Average Percentage Gain of Ordinary Data Transfer Flows

MFR New		SFR New		MFV New		SFV New	
K	Gain	K	Gain	K	Gain	K	Gain
25	12.3	50	12.9	25	-0.6	50	-0.7
50	11.9	100	12.8	50	-0.7	100	-0.3
100	10.8	200	11.6	100	-0.8	200	-1.1
200	8.6	400	10.8	200	-0.8	400	-1.2
400	7.4	800	8.4	400	-0.9	800	-1.4

4.3.9 Target Catching Experiment

We conducted a target catching experiment where the sink periodically sends the estimates of the target's position and velocity to a mobile agent. The objective of the mobile agent is to catch the target in a minimum possible time. We assume that the mobile agent has a constant speed and is capable of sudden changes in its direction of movement. When the mobile agent gets a new estimate of the position and velocity of the target, based on its own position and velocity, it calculates its new direction in order to catch the target. When the mobile agent is within 5 ft of the target, then the target is considered to be 'caught'.

The agent starts its pursuit at time 300 s. We have conducted fifteen experiments for each simulation setup by randomly selecting the initial position and the velocity of the mobile agent. The initial position is selected in a way that it is always inside the

4000×4000 ft^2 simulated area. Moreover, the velocity of the mobile agent is selected in a way that it is always higher than the average speed of the target (20 ft/s) in order to make sure that the mobile agent can catch the target. In order to compare the performance of the proposed approach with the standard rate controllers, we have calculated the percentage improvement in the catching times with the introduction of the new protocol. The average percentage improvement of the catching times for the experiments for all four cases are given in Table 4.4. It is clear from the results that the average catching time is reduced with the introduction of the new protocol for all four simulation setups. Results for the MFR case with the initial position of $(2000, 1600)$ and the velocity of $35m/s$ appear in Fig. 4.8. It is clear form the figure that the agent catches the target much earlier with MFR New than with MFR Reno.

Table 4.4: Average Percentage Improvement of the Catching Times

MFR	SFR	MFV	SFV
39.58	63.63	49.25	60.57

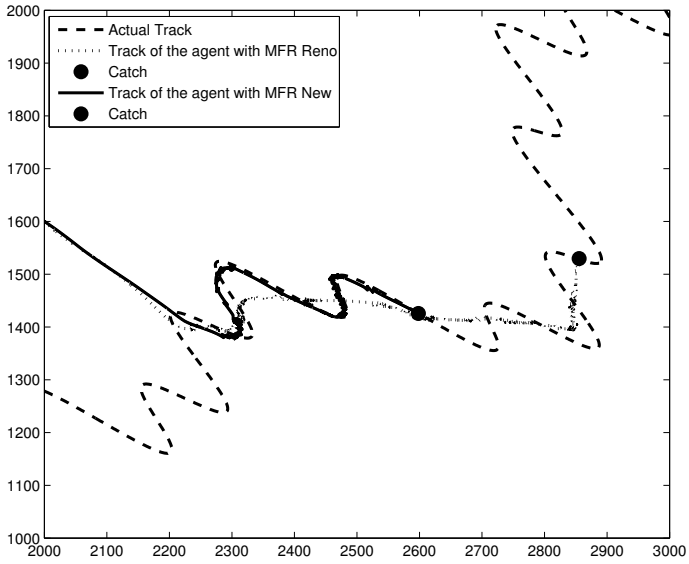

Figure 4.8: Tracks of the mobile agent.

4.3.10 User Datagram Protocol (UDP) Rate Controller

Another set of experiments were conducted using a UDP based rate controller for the proposed protocol. This rate controller is not a window based rate controller as in the above experiments. In this experiment sensor transmission rates were directly calculated using (4.40). TCP Reno was used for regular data transfer flows. A same network topology as explained in the above experiment was used in this experiment. Two experiment setups, one with MF and another one with SF were used. For MF and SF, same parameters as explained in the above experiment were used. For comparison, simulations were conducted by selecting TCP Friendly Rate Controller (TFRC) as the rate controller for sensors. Here, TFRC does not utilize feedback from the sink node. Simulation results in terms for MSE are given in Table 4.5. It is clear from the results that performance of both MF and SF have increased with the new rate controller over TFRC.

Table 4.5: UDP Experiment: MSE Values for Estimated Tracks

	MF	SF
New	44.62	47.75
TFRC	75.20	79.15

4.3.11 Effects of available bandwidth on the tracking performance

We have conducted an experiment to evaluate the effects of available bandwidth on the tracking performance. In this experiment, we changed the bandwidths of congested links and calculated MSE corresponding to different bandwidths. The strategy we adopted to change the bandwidth is that we multiplied bandwidths of all congested links by a constant multiplication factor. Corresponding MSE values for

different multiplication factors for MF are given in Fig. 4.9. Same results for SF are given in Fig. 4.10.

It is clear from the results that the tracking performance for both MF and SF increase as the bandwidth increases up to some point and stays steady after that. The reason for this behavior is that the estimation error due to the transmission delay becomes prominent after some point and the performance cannot be increased by just increasing the bandwidth.

4.3.12 Effects of quantized measurements

In the above experiments we have assumed that readings are constant in byte size. Moreover, the frequency of transmission is changed based on the available bandwidth. Another obvious possibility is that the readings are transmitted at a constant rate with varying sizes in bytes. In this approach it is necessary to quantize the measurements based on the available bandwidth before transmission.

In this experiment, sensors are taken to be capable of producing readings of varying quantized levels at a fixed frequency. The quantized level is based on the available bandwidth. We used a linear quantizer with 12 bits. This gives 24 bits (3 bytes) to represent a single location (12 bits for the x coordinate and 12 bits for the y coordinate). With a 12 bit quantizer, possible number of levels are $2^{12} = 4096$. In this experiment, the same network topology that was used in the original experiment was used. Two simulation setups, MFR and SFR, were used for the experiment. The performance of the simulation was measured by means of MSE values. Calculated MSE vales for the two simulation setups were given in Table 4.6. It is clear from the results that the performance of both MF and SF improved with the introduction of the new protocol.

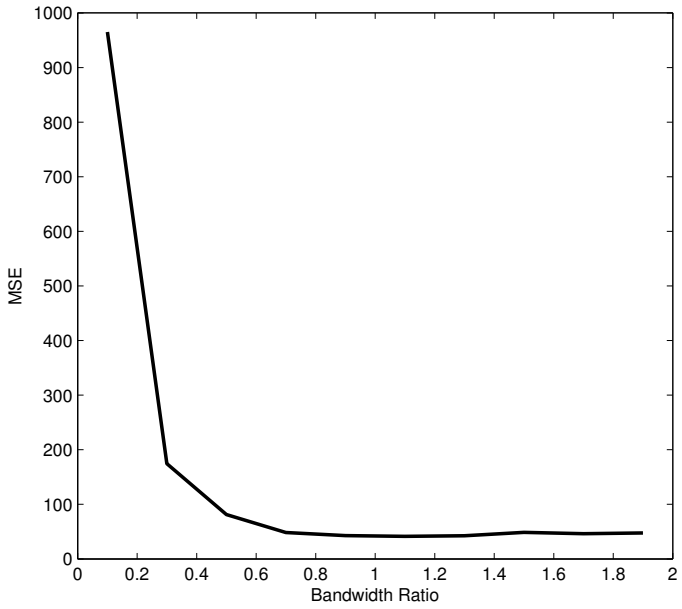

Figure 4.9: Tracking performance of MF for different bandwidths.

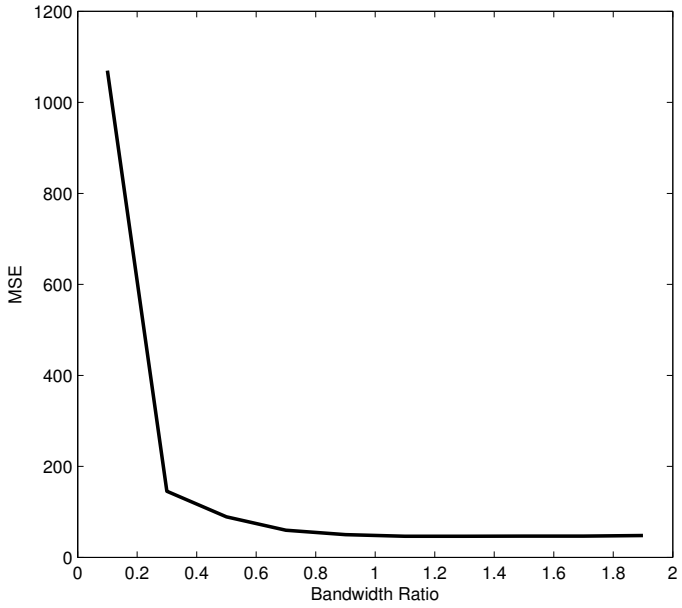

Figure 4.10: Tracking performance of SF for different bandwidths.

Table 4.6: Quantized Experiment: MSE Values for Estimated Tracks

	MFR	SFR
New	78.30	69.22
Reno	125.46	113.03

4.3.13 Effects of measurement approximation

In the above experiments, we used a zero-order hold in the case when new measurements are not arrived for Kalman update. In this approach, for each sensor, the previously arrived measurement is used for Kalman updates until a new measurement is arrived. Once a new measurement is arrived, the previous measurement is replaced with the newly arrived measurement.

However, one natural extension to the above approach is to use a measurement approximation strategy to approximate the measurement by utilizing the previously arrived measurements in the case when new measurements are not arrived for Kalman update. In this experiment we utilized the linear prediction to approximate the measurements. Here, two most recent measurements are used for the linear prediction. For this experiment, MSE values of the estimated tracks for both the proposed protocol and TCP Reno are given in Table 4.7. It is clear form the results that the proposed protocol has better performance than TCP Reno. However, it is noticeable that the performances are lowered from the zero-order hold approximation for both the new protocol and TCP Reno. Reason for this performance degradation is that the measurements are not exactly accurate since proximity errors are there in the measurements. Hence, the linearly approximated measurements utilizing the most

recent two measurements can be even worst than the zero-order hold approximated measurements.

Table 4.7: Measurement Approximation Experiment: MSE Values for Estimated Tracks

	MFR	SFR
New	67.40	77.96
Reno	85.61	90.01

CHAPTER 5

Rate Allocation in an Application with Multiple Objectives: Multi-Sensor Target Tracking and Classification

Can the resource allocation framework developed in the previous chapter be utilized in scenarios that require one to simultaneously satisfy multiple objectives? In this chapter we explore this issue and conclude that it is indeed possible to extend the same resource allocation framework for this purpose. We justify this claim for one specific application: *simultaneous tracking and classification of targets.* We believe that this constitutes a sufficiently generic application so that the principles and the results that we develop in this chapter can form the basis on which other application scenarios can be addressed.

While the principle objective of target tracking is to accurately estimate the path of a target utilizing the measurements from multiple sensors, the main objective of target classification is to accurately determine the class to which each target belongs to. This task requires the selection and use of appropriate utility functions that capture the QoS measures of both the objectives of target tracking and classification. Of course, the utility function that we employed in the previous chapter can still be used for the target tracking objective; the target classification objective however calls for the selection of a new utility function.

5.1 Target Classification Utility Function

We now develop a new utility function for the target classification objective so that target classification performance can be maximized by maximizing the utility function.

To proceed, let us first explain the classification environment. We assume that classification is based on a real-valued attribute vector $X = \{\chi_1, \chi_2, \ldots, \chi_N\}$ of length N. Furthermore, Naïve Bayes approach as explained in Section 3.1.3 is used for classification. The N attributes are measured by N sensors and, as before, the readings are transmitted to the sink node through a shared network. Note that measurement inaccuracies will be unavoidable due to the unavailability of an infinite bandwidth along each link. The parameters of various conditional probability distributions and prior probabilities are assumed to be stored at the sink node; the conditional probabilities are calculated based on the received attribute measurements.

Let us assume that the classifier is to classify the targets into Θ number of classes $\{C_1, C_2, \ldots, C_\Theta\}$. To proceed, we introduce the following quantities:

$$Pr(C_i) = \text{Prior probability of the class } i;$$

$$Pr(\chi_k|C_i) = \text{Conditional probability of the } k\text{-th attribute } \chi_k \text{ given the class } C_i.$$

The unavailability of an infinite bandwidth along each link introduces an error into the conditional probability $Pr(\chi_k|C_i)$. Let us assume that this error is ϵ_{ik}. Later in Section 5.1.2, we will develop a model for this error.

We proceed by introducing the quantity Υ as

$$\Upsilon = \frac{\sum_{j=1}^{\Theta} Q'(j)}{\sum_{j=1}^{\Theta} Q(j)} \sum_{i=1}^{\Theta} \left| \frac{Q(i)}{\sum_{j=1}^{\Theta} Q(j)} - \frac{Q'(i)}{\sum_{j=1}^{\Theta} Q'(j)} \right|, \tag{5.1}$$

where

$$Q(i) = Pr(C_i) \prod_{k=1}^{N} Pr(\chi_k|C_i) \text{ and } Q'(i) = Pr(C_i) \prod_{k=1}^{N} [Pr(\chi_k|C_i) + \epsilon_{ik}]. \tag{5.2}$$

Note the following:

- Υ is a measure of the difference between the estimated probability distributions and their corresponding true distributions.
- $\Upsilon \rightarrow 0$ as $\epsilon_{ik} \rightarrow 0$, $\forall i, k$.

Therefore, it is clear that classification performance can be maximized by minimizing the quantity Υ. In Section 5.1.1, we in fact establish the relationship between this quantity Υ and the Kullback-Leibler (K-L) divergence between the estimated and the actual probability distributions.

Next, we make the following assumption, the consequences of which we will discuss later.

Assumption 1 *The errors in the conditional probability estimates are small compared to their actual conditional probability values. In particular, we assume that* $|\epsilon_{ik}/Pr(\chi_k|C_i)| < \epsilon$, $\forall i, k$, *for some small real number* $\epsilon << 1$.

With Assumption 1 in place, by ignoring second-order terms related to ϵ_{ik}, we can write

$$\frac{Q'(i)}{\sum_{j=1}^{\Theta} Q'(j)} - \frac{Q(i)}{\sum_{j=1}^{\Theta} Q(j)} \approx \Upsilon'(i), \tag{5.3}$$

where

$$\Upsilon'(i) = \frac{Pr(C_i)}{\sum_{j=1}^{\Theta} Q(j) \sum_{j=1}^{\Theta} Q'(j)}$$
$$\sum_{j=1}^{\Theta} \left[Pr(C_j) \left[\prod_{k=1}^{N} Pr(\chi_k|C_i) \left(\prod_{k=1}^{N} Pr(\chi_k|C_j) + \sum_{l=1}^{N} \frac{\prod_{m=1}^{N} Pr(\chi_m|C_j)}{Pr(\chi_l|C_j)} \epsilon_{jl} \right) \right. \right.$$
$$\left. \left. - \prod_{k=1}^{N} Pr(\chi_k|C_j) \left(\prod_{k=1}^{N} Pr(\chi_k|C_i) + \sum_{l=1}^{N} \frac{\prod_{m=1}^{N} Pr(\chi_m|C_i)}{Pr(\chi_l|C_i)} \epsilon_{il} \right) \right] \right]. \quad (5.4)$$

So we may write

$$\Upsilon \approx \Upsilon' \equiv \frac{\sum_{j=1}^{\Theta} Q'(j)}{\sum_{j=1}^{\Theta} Q(j)} \sum_{i=1}^{\Theta} |\Upsilon'(i)|. \quad (5.5)$$

Henceforth, we will take Assumption 1 to hold true. Therefore, instead of Υ, we will continue to use Υ'. Note that Υ' can be expressed as

$$\Upsilon'(i) = \frac{1}{\sum_{j=1}^{\Theta} Q(j) \sum_{j=1}^{\Theta} Q'(j)}$$
$$\left[\sum_{l=1}^{N} Pr(C_i) \prod_{k=1}^{N} Pr(\chi_k|C_i) \sum_{j=1}^{\Theta} \frac{Pr(C_j) \prod_{k=1}^{N} Pr(\chi_k|C_j)}{Pr(\chi_l|C_j)} \epsilon_{jl} \right.$$
$$\left. - \sum_{j=1}^{\Theta} Pr(C_j) \prod_{k=1}^{N} Pr(\chi_k|C_j) \sum_{l=1}^{N} \frac{Pr(C_i) \prod_{k=1}^{N} Pr(\chi_k|C_i)}{Pr(\chi_l|C_i)} \epsilon_{il} \right]. \quad (5.6)$$

By considering that conditional and prior probability values are necessarily positive while ϵ_{ik} can take either positive or negative values, we may now establish the follow-

ing upper bound on the absolute value of $\Upsilon'(i)$:

$$|\Upsilon'(i)| \leq \frac{1}{\sum_{j=1}^{\Theta} Q(j) \sum_{j=1}^{\Theta} Q'(j)} \left[\sum_{l=1}^{N} Pr(C_i) \prod_{k=1}^{N} Pr(\chi_k|C_i) \sum_{j=1}^{\Theta} \frac{Pr(C_j) \prod_{k=1}^{N} Pr(\chi_k|C_j)}{Pr(\chi_l|C_j)} |\epsilon_{jl}| \right.$$
$$\left. + \sum_{j=1}^{\Theta} Pr(C_j) \prod_{k=1}^{N} Pr(\chi_k|C_j) \sum_{l=1}^{N} \frac{Pr(C_i) \prod_{k=1}^{N} Pr(\chi_k|C_i)}{Pr(\chi_l|C_i)} |\epsilon_{il}| \right]. \quad (5.7)$$

Simple, yet somewhat lengthy, manipulations (see Appendix C) then yield

$$\sum_{i=1}^{\Theta} |\Upsilon'(i)| \leq \frac{\sum_{l=1}^{N} \sum_{j=1}^{\Theta} B_{jl} |\epsilon_{jl}|}{\sum_{j=1}^{\Theta} Q'(j)}, \text{ where } B_{jl} = 2 \frac{Pr(C_j) \prod_{k=1}^{N} Pr(\chi_k|C_j)}{Pr(\chi_l|C_j)}. \quad (5.8)$$

Now substitute (5.8) into (5.5) to get

$$\Upsilon' \leq \frac{\sum_{l=1}^{N} \sum_{j=1}^{\Theta} B_{jl} |\epsilon_{jl}|}{\sum_{j=1}^{\Theta} Q(j)}. \quad (5.9)$$

It is the negative of this upper bound of Υ' that we propose to use as our target classification utility function. Accordingly, we consider the following classification utility function:

$$S_C = -\frac{\sum_{l=1}^{N} \sum_{j=1}^{\Theta} B_{jl} |\epsilon_{jl}|}{\sum_{j=1}^{\Theta} Q(j)}. \quad (5.10)$$

5.1.1 Relationship Between Υ and Kullback-Leibler (K-L) Divergence

A justification for selecting the quantity Υ in (5.1) for developing a utility function for the target classification objective is that Υ can be thought of as a measure of the estimated and actual probability distributions. Indeed, in this section, we demonstrate this fact by establishing the relationship between Υ and the K-L divergence of the estimated and actual probability distributions. The K-L divergence is a widely used measure of the divergence between two probability distributions.

To proceed, note that, we may define the K-L divergence as

$$KLD = \sum_{i=1}^{\Theta} \frac{Q(i)}{\sum_{j=1}^{\Theta} Q(j)} \log \left[\frac{Q(i) \sum_{j=1}^{\Theta} Q'(j)}{Q'(i) \sum_{j=1}^{\Theta} Q(j)} \right]. \tag{5.11}$$

By replacing the log[•] term of (5.11) with the first two terms of the Taylor series expansion of the log[•] function about unity, we get the following approximation of KLD:

$$KLD' = \sum_{i=1}^{\Theta} \frac{Q(i)}{\sum_{j=1}^{\Theta} Q(j)} \left(\frac{Q(i) \sum_{j=1}^{\Theta} Q'(j)}{Q'(i) \sum_{j=1}^{\Theta} Q(j)} - 1 \right) \tag{5.12}$$

$$= \sum_{i=1}^{\Theta} \frac{Q(i)}{Q'(i)} \frac{\sum_{j=1}^{\Theta} Q'(j)}{\sum_{j=1}^{\Theta} Q(j)} \left(\frac{Q(i)}{\sum_{j=1}^{\Theta} Q(j)} - \frac{Q'(i)}{\sum_{j=1}^{\Theta} Q'(j)} \right). \tag{5.13}$$

Therefore, we may bound the absolute value of KLD' as

$$|KLD'| \leq \sum_{i=1}^{\Theta} \frac{Q(i)}{Q'(i)} \frac{\sum_{j=1}^{\Theta} Q'(j)}{\sum_{j=1}^{\Theta} Q(j)} \left| \frac{Q(i)}{\sum_{j=1}^{\Theta} Q(j)} - \frac{Q'(i)}{\sum_{j=1}^{\Theta} Q'(j)} \right|. \tag{5.14}$$

With Assumption 1 in place, we then have

$$\frac{Q(i)}{Q'(i)} = \frac{\prod_{k=1}^{N} Pr(\chi_k|C_i)}{\prod_{k=1}^{N}[Pr(\chi_k|C_i)+\epsilon_{ik}]} = \prod_{k=1}^{N} \frac{Pr(\chi_k|C_i)}{[Pr(\chi_k|C_i)+\epsilon_{ik}]}$$
$$\leq \prod_{k=1}^{N} \frac{1}{1-\epsilon} = \left(\frac{1}{1-\epsilon}\right)^N. \tag{5.15}$$

Substitute (5.15) into (5.14) to get an upper bound for KLD':

$$|KLD'| \leq \left(\frac{1}{1-\epsilon}\right)^N \frac{\sum_{j=1}^{\Theta} Q'(j)}{\sum_{j=1}^{\Theta} Q(j)} \sum_{i=1}^{\Theta} \left| \frac{Q(i)}{\sum_{j=1}^{\Theta} Q(j)} - \frac{Q'(i)}{\sum_{j=1}^{\Theta} Q'(j)} \right|$$
$$= \left(\frac{1}{1-\epsilon}\right)^N \Upsilon. \tag{5.16}$$

5.1.2 Conditional Probability Error Model

Recall that an N number of sensors are being used to measure an N number of attributes $\{\chi_1, \chi_2, \ldots, \chi_N\}$. The attribute measurements are then sent to a common sink node via a shared network. The common sink node uses stored conditional probability distributions to compute the conditional probability values corresponding to the received attribute measurements. Since the sensors do not have the ability to use an infinite bandwidth to send the information, the measurements being received at the sink node will not be ideal and will be associated with an error.

To proceed, consider the sensor k and the corresponding attribute measurement χ_k. Let f_k be the transmission bit-rate of the sensor k. Let us assume that the error introduced into the measurement χ_k due to the non-availability of infinite bandwidth is t_k. In other words, the sink node is forced to utilize the attribute value $\chi_k + t_k$ to

calculate the conditional probability values and perform the classification task. Now, observe the following regarding the measurement error t_k:

- As $f_k \to \infty$, we need $|t_k| \to 0$: That is, if an infinite transmission rate is available to the sensor, then the measurement error should be zero.
- As $f_k \to 0$, we need $|t_k| \to \infty$: This is the opposite to the above, i.e., if the sensor transmission rate is zero, then no information is being transmitted and accordingly, the measurement error should become infinite.
- $|t_k|$ should be a decreasing function of f_k: This is because the accuracy of the reading should increase as more data are transmitted.
- S_C should be a concave function of $\mathbf{f}$ (the vector of all the sensor bit-rates): This requirement is imposed to ensure a global maximum for the utility.

Based on these observations, we propose and use the following model for the measurement error t_k:

$$|t_k| = \frac{d_c}{f_k}, \ \forall k = 1, \ldots, N, \tag{5.17}$$

where $d_c > 0$ is a constant.

Next, we note the following: the error ϵ_{ik} of the conditional probability $Pr(\chi_k|C_i)$ is directly due to the measurement error t_k. Assuming that t_k is the only source of the error ϵ_{ik}, we use the following approximation of ϵ_{ik}:

$$|\epsilon_{ik}| \approx |t_k| \, \Delta_{ik} = \frac{d_c}{f_k} \, \Delta_{ik}, \tag{5.18}$$

where $\Delta_{ik} = \left| \dfrac{d}{d\chi_k} \, Pr(\chi_k|C_i) \right|$. If the conditional probability $Pr(\chi_k|C_i)$ is normally distributed, we would have

$$\Delta_{ik} = \left| \frac{(\chi_k - \mu_{ik})}{\sigma_{ik}^3 \sqrt{2\pi}} \, \exp\left(-\frac{(\chi_k - \mu_{ik})^2}{2\sigma_{ik}^2} \right), \right| \tag{5.19}$$

where μ_{ik} and σ_{ik} are the mean and the standard deviation of the distribution.

As mentioned earlier, it is important that the utility function be a concave function of $\mathbf{f}$ (the vector of all the sensor bit-rates). It turns out that the above model for the error renders a concave utility function S_C

Lemma 2 *Given that the absolute value of the conditional probability error $|\epsilon_{ik}|$ can be described as in (5.18), the classification utility S_C is concave w.r.t. $\mathbf{f}$.* □

Proof: Note that $|\epsilon_{ik}|$ as given in (5.18) is convex w.r.t. $\mathbf{f}$. Hence, $-|\epsilon_{ik}|$ is a Concave function of $\mathbf{f}$. Further, note that S_C is a linear combination of the $-|\epsilon_{ik}|$ values. Hence, S_C is concave w.r.t. $\mathbf{f}$ because $B_{jl} \geq 0$, $\forall j, l$, and $Q(j) \geq 0$, $\forall j$. ■

5.1.3 Consequences of Assumption 1

Now we are in a position to further study the consequences of Assumption 1. We first express Assumption 1 as

$$\frac{(d_c/f_k)\,\Delta_{ik}}{Pr(\chi_k|C_i)} < \epsilon \Longrightarrow f_k > \frac{d_c\,\Delta_{ik}}{\epsilon\,Pr(\chi_k|C_i)}. \tag{5.20}$$

For this to be true for all the classes we need

$$f_k > \max_i \frac{d_c\,\Delta_{ik}}{\epsilon\,Pr(\chi_k|C_i)}. \tag{5.21}$$

So, it is clear from (5.21) that a minimum rate requirement should be imposed so that Assumption 1 holds true.

5.2 Multiple Objective Utility and Iterative Rate Update Function

To simultaneously satisfy both the target tracking and classification objectives. we now modify the target tracking utility function in (4.31) by adding the target classification utility as an additive term. The modified utility function then becomes

$$U_j(\mathbf{f}) = V_j(f_j) + K\hat{S}(k+1) + K_c S_C,\ \forall j, \tag{5.22}$$

where K and K_c are two real parameters. The parameter K determines the emphasis placed on the target tracking performance improvement; the parameter K_c determines the emphasis placed on the target classification performance improvement.

Following much the same approach as was used earlier, we can now obtain the following:

5.2.1 TCP Reno Compatible Controller

- Rate update function:

$$f_m(t+1) = \left[f_m(t) + s \left(\frac{2}{2 + f_m(t)^2 RTT_m(t)^2} - q_m(t) - \zeta_m(\mathbf{f}(t)) \right) \right]^+, \quad (5.23)$$

where

$$\zeta_m(\mathbf{f}(t)) = \xi_m(\mathbf{f}(t)) + K_c I_m \frac{d_c}{f_m(t)^2 \sum_{j=1}^{\Theta} Q(j)} \sum_{j=1}^{\Theta} B_{jm} \Delta_{jm}. \quad (5.24)$$

- Window update function:

$$w_m(t+1) = \left[w_m(t) + s\, RTT_m(t) \left(\frac{2}{2 + w_m(t)^2} - q_m(t) - \zeta_m(\mathbf{f}(t)) \right) \right]^+. \quad (5.25)$$

5.2.2 TCP Vegas Compatible Controller

- Rate update function:

$$f_m(t+1) = \left[f_m(t) + s \left(\frac{\alpha}{f_m(t)} - q_m(t) - \zeta_m(\mathbf{f}(t)) \right) \right]^+. \quad (5.26)$$

- Window update function:

$$w_m(t+1) = \left[w_m(t) + s\, RTT_m(t) \left(\frac{\alpha\, RTT_m(t)}{w_m(t)} - q_m(t) - \zeta_m(\mathbf{f}(t)) \right) \right]^+. \quad (5.27)$$

5.3 Experiments

The same simulation environment and the network used in Section 4.3 is employed in this experiment. The target moves along the same path. Now however each sensor measures and sends an attribute to be used for the classification task together with the position information. The i-th sensor measures and sends the attribute χ_i. These attribute values change over the time. As explained in Section 4.3, the frequency at which measurements are transmitted depends on the allocated transmission rate. The sink node uses either MF or SF for target tracking. Moreover, the sink node uses Naïve Bayes method to classify the target into one out of ten equi-probable classes. The same four simulation setups MFR, SFR, MFV and SFV with the same parameter values except for K and K_c, as explained in Section 4.3.6 are employed in this experiment. Values used for parameters K and K_c are given in Table 5.1.

Table 5.1: Experiment Parameters: Multi-Sensor Target Tracking and Classification

	MFR	SFR	MFV	SFV
K	100	200	100	200
K_c	20	20	20	20

For the MFR case, the reconstructed tracks for both the baseline MFR Reno, and the new protocol MFR New, together with the actual target track, are given in Fig. 5.1. Dark dots in the figure indicate locations where the target is wrongly classified. This figure only shows a (600×350) ft^2 section of the test area for the target. These results are representative of the performance within other parts of the test area. In Fig. 5.1, one can see that when the sensors use the new protocol, the target tracking application is able to achieve a much more accurate track. Moreover, one can see that the wrongly classified locations are lowered for the new protocol.

The corresponding simulation results for the *SFR*, *MFV* and *SFV* cases are given in Figs. 5.2, 5.3 and 5.4, respectively.

As in Section 4.3 we have numerically evaluated the target tracking performance by calculating the MSE values for the estimated tracks. The calculated MSE values are given in Table 5.2. It is clear from the results that for all four simulation setups, the MSE values have decreased with the new protocol.

Table 5.2: Target Tracking MSE values: Multi-Sensor Target Tracking and Classification

	MFR	SFR	MFV	SFV
New	45.02	52.03	45.23	49.15
Reno/Vegas	70.39	75.78	82.58	85.48

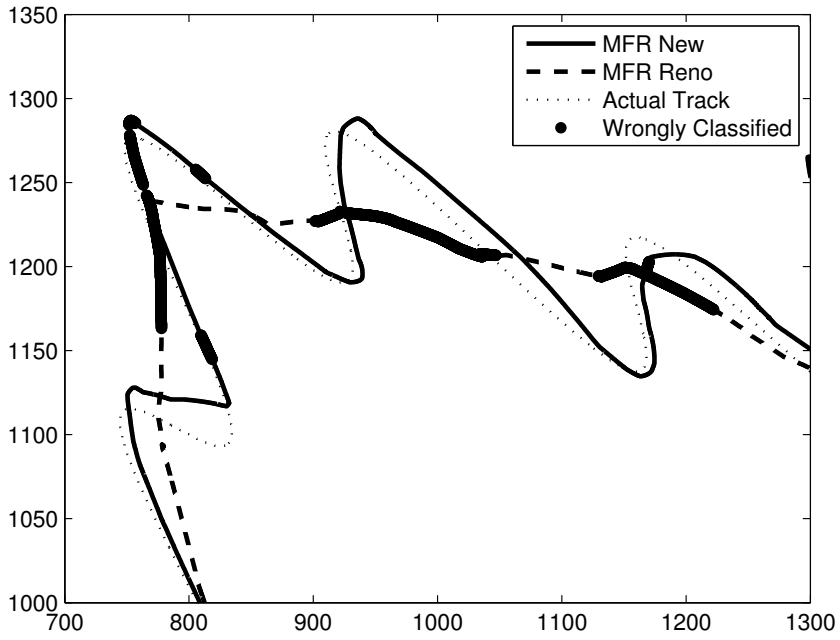

Figure 5.1: Reconstructed tracks for MFR: Multi-Sensor Target Tracking and Classification

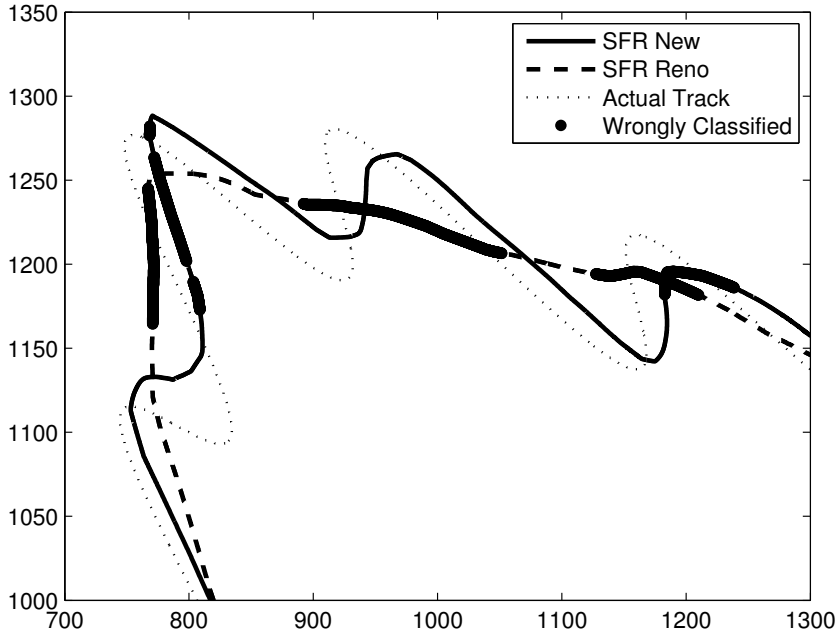

Figure 5.2: Reconstructed tracks for SFR: Multi-Sensor Target Tracking and Classification

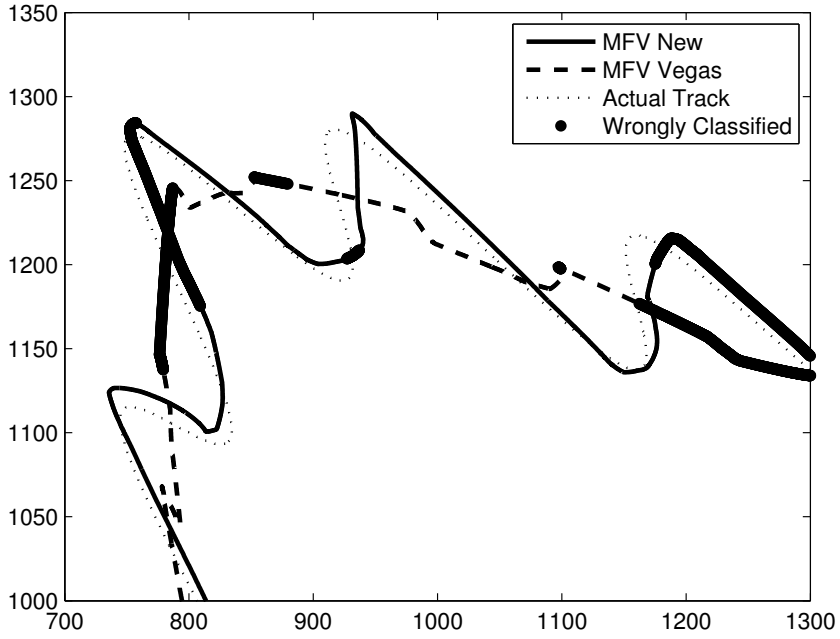

Figure 5.3: Reconstructed tracks for MFV: Multi-Sensor Target Tracking and Classification

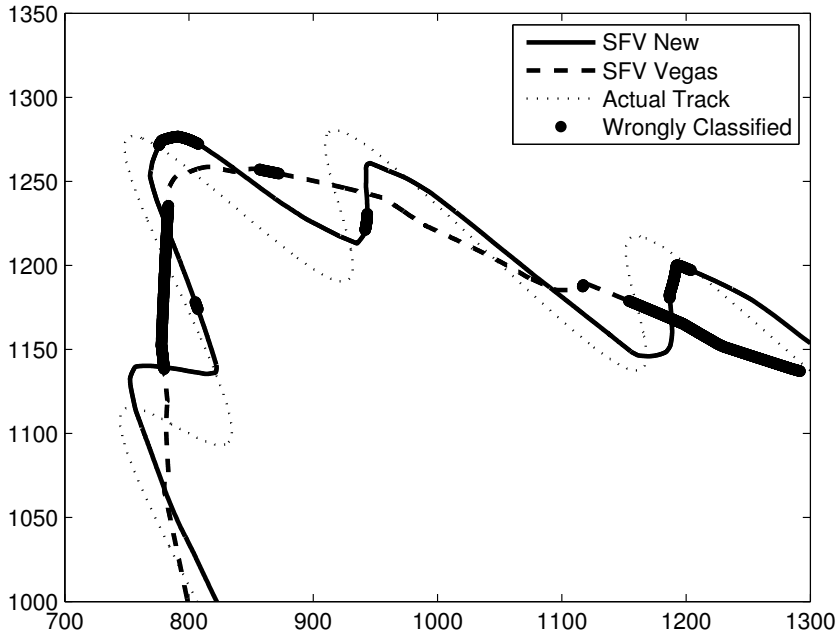

Figure 5.4: Reconstructed tracks for SFV: Multi-Sensor Target Tracking and Classification

We utilized two measures to numerically evaluate the target classification performance.

1. K-L Divergence between the original and the estimated probability distributions: The original probability distribution is the class probability distribution calculated using the actual attribute values. The estimated probability distribution is the class probability distribution calculated using the attribute values available at the sink node.

2. The percentage of locations that the original and the estimated classification decisions agree with other: The original classification decision is derived using the actual attribute values. The estimated classification decision is derived using the attribute values available at the sink node.

The calculated average K-L Divergence values are given in Table 5.3. It is clear from the results that the average K-L Divergence values have decreased with the new protocol. The decisions agreeing percentages are given in Table 5.4. It is clear from the results that, with the new protocol, percentages are increased.

Table 5.3: Target Classification K-L Divergence values: Multi-Sensor Target Tracking and Classification

	MFR	SFR	MFV	SFV
New	0.21	0.22	0.28	0.21
Reno/Vegas	0.62	0.62	0.51	0.51

Table 5.4: Target Classification Accuracies: Multi-Sensor Target Tracking and Classification

	MFR	SFR	MFV	SFV
New	90.86	90.27	89.76	93.12
Reno/Vegas	82.10	82.10	87.21	87.21

CHAPTER 6

Belief Theoretic Approach for Flow Classification

In this chapter, we propose a BN model based traffic classification algorithm. Classification is carried out using a window of increasing size of packets to enable online classification. In order to represent different sizes of the window, we propose to utilize a series of BNs. Partial flow information cannot provide the full information that one requires for classification. As the size of the window of packets is increased, or in other words, as more information about the flow becomes available, uncertainties associated with information gathered from partial flow will diminish. However, one must accurately capture these uncertainties in order to perform a more accurate classification with partial flow information.

DS theory is capable of representing uncertainties in a more intuitive way. For example, when a single value cannot be discerned for a particular variable, DS theory can be utilized to consider a composite instead of a single value. Hence, one can use DS theory to model sources of evidence, particularly uncertain sources of evidence in a more realistic and informative way than probabilistic models. So, with DS theoretic techniques, one can arrive at a decision with the full understanding of the associated underlying uncertainties. DS belief theoretic methods are more robust to modeling errors [Sme99]. DS theory provides techniques to combine several sources of evidence

in a more productive manner. Furthermore, probabilistic models can be easily transformed into DS theory models. Similarly, transition from DS theory to probability can also be done easily. Moreover, when there is no any uncertainties, DS models converge to a Bayesian model. So, one can consider DS theoretic models as extensions of Bayesian models. Machine learning algorithms can be improved by augmenting with DS notions since DS theory facilitates one to extract useful knowledge from imperfect data. Moreover, DS theory facilitates one to utilize techniques such as *evidence filtering (EF)* for detection of faint patterns in 'clutter' [DBP06a, DBP06b, DBP07]. A DS belief theory based approach is ideal for our proposed approach because of its ability to represent a wide variety of data imperfections while requiring little more information than voting and set intersection techniques [BP99]. Hence, we propose to utilize DS theory in our flow classification algorithm.

6.1 DS Theory: A Primer

To better understand the proposed algorithm, one must familiar with the preliminaries of DS theory. Hence, we introduce preliminaries of DS theory in this section. Suppose the flows are to be classified into a finite set of mutually exclusive and exhaustive classes $\{1, 2, \ldots, num\_cls\}$. In DS theory, we capture these classes as propositions belonging to the *frame of discernment (FoD)* $\Theta = \{1, 2, \ldots, num\_cls\}$. It signifies the 'scope of expertise' about the problem domain. Note that $|\Theta|$, the cardinality of Θ, is the number of distinct flow classes num_cls. A proposition $n \in \Theta$, referred to as a *singleton,* represents the lowest level of discernible information. Elements in 2^{Θ}, the power set of Θ, form all propositions of interest in a DS theoretic model. A proposition that is not a singleton is referred to as a *composite.* A composite propo-

sition denotes lack of sufficient evidence to resolve among its constituent singletons, e.g., the inability to resolve between the two singletons 1 and 2 are captured via the composite $(1, 2)$.

The set $B \setminus C$ denotes all singletons in $B \subseteq \Theta$ that are not included in $C \subseteq \Theta$, viz., $B \setminus C = \{k \in \Theta : k \in B,\ k \notin C\}$; $\overline{C}$ denotes $\Theta \setminus C$. The null proposition is denoted via $\emptyset$.

Definition 2 *The mapping* $m : 2^{\Theta} \mapsto [0, 1]$ *is a* basic mass assignment *for the FoD* Θ *if (i)* $m(\emptyset) = 0$*; and (ii)* $\sum_{C \subseteq \Theta} m(C) = 1$. ■

The mass of a proposition is free to move into its individual singletons. This is how DS theory models *ignorance.* Complete ignorance, or complete lack of evidence to discern among any of the classes, can be conveniently captured via the *vacuous mass structure:*

$$m(C) = \begin{cases} 1, & \text{if } C = \Theta; \\ 0, & \text{otherwise.} \end{cases} \tag{6.1}$$

A proposition that possesses a nonzero mass is referred to as a *focal element;* the set of focal elements is the *core* F. The triple $\{\Theta, F, m(\bullet)\}$ is referred to as the *body of evidence (BoE).*

Definition 3 *Given a BoE* $\{\Theta, F, m(\bullet)\}$ *and* $C \subseteq \Theta$*, (i)* $Bel : 2^{\Theta} \mapsto [0, 1]$ *where* $Bel(C) = \sum_{B \subseteq C} m(B)$ *is the* belief of C*; and (ii)* $Pl : 2^{\Theta} \mapsto [0, 1]$ *where* $Pl(C) = 1 - Bel(\overline{C})$ *is the* plausibility of C. ■

So, while $m(C)$ measures the support assigned to proposition C *only,* $\mathrm{Bel}(C)$ takes into account the supports for all proper subsets of C as well. Thus $\mathrm{Bel}(C)$ represents the total support that can move into C without any ambiguity. $\mathrm{Pl}(C)$ represents the

extent to which one finds C plausible. When the core has singletons only, all of these notions reduce to probability.

A probability distribution $\Pr(\bullet)$ is said to be *compatible* with the underlying mass structure $m(\bullet)$ if $\mathrm{Bel}(C) \leq \Pr(C) \leq \mathrm{Pl}(C)$, $\forall C \subseteq \Theta$. An example of such a probability distribution is the *pignistic probability distribution BetP*$(\bullet)$ defined for each singleton $\theta^{(i)} \in \Theta$ as [Sme99]

$$\mathrm{BetP}(\theta^{(i)}) = \sum_{\theta^{(i)} \in A \subseteq \Theta} \frac{m(A)}{|A|}. \tag{6.2}$$

The *Dempster's Combination Rule (DCR)* enables one to 'pool' the evidence of two 'independent' BoEs to form a single BoE.

Definition 4 *Consider two BoEs* $\{\Theta, \mathcal{F}_i, m_i\}$, $i = 1, 2$, *that span the same FoD* Θ. *Then, provided that the* conflict *defined as* $K \equiv \sum_{\substack{C \in \mathcal{F}_1,\, D \in \mathcal{F}_2 \\ C \cap D = \emptyset}} m_1(C)\, m_2(D) \neq 1$, *the DCR yields the mass assignment* $m(\bullet) : 2^{\Theta} \mapsto [0, 1]$ *where*

$$m(A) = \frac{\sum_{\substack{C \in \mathcal{F}_1,\, D \in \mathcal{F}_2 \\ C \cap D = A}} m_1(C)\, m_2(D)}{1 - K}, \ \forall A \subseteq \Theta.$$

This combination operation is denoted as $m(\bullet) = (m_1 \oplus m_2)(\bullet)$. ■

The operation $\oplus$ is both associative and commutative thus enabling the combination of multiple BoEs with ease [Sha76].

Intrusion detection which is a special case of the flow classification problem where only two classes—normal traffic and attacks—are used, is an application domain where DS theory has been effectively utilized [CV05, HLG06, HL08]. Intrusion detection systems are instrumental in network security since they can help one to detect abnormal network behavior.

Chen and Venkataramanan [CV05] focus on intrusion detection in mobile ad hoc networks. The authors have pointed out that the detection accuracy of any particular node is not perfect. However, the accuracy can be improved by combining sources of evidence from several nodes. The combination of evidence is carried out using DS theoretic methods. Hu, et al. [HLG06] use the DS theory of evidence combination to identify intrusions in IP networks. The idea is to combine evidence from a fewer number of sensors to improve the accuracy. He and Leung [HL08] exploit changes in the pattern of normal traffic when an intrusion occurs to detect the intrusion. They use constant false alarm rate (CFAR) to detect anomalies. Again, DS theory is used to combine evidence generated from different detectors. So, these previous works essentially exploit the ease with which evidence can be combined with DS theoretic techniques. Our proposed use of DS theory is different because we utilize DS theory to model uncertainties inherent in partial flow information and to combine inferences derived with partial flows.

6.2 Proposed Approach

Details of our proposed flow classification algorithm is explained in this section. The proposed flow classification algorithm is based on a sequence of BNs. We use $(PK_{max}-1)$ number of BNs identified as $BN_2, \ldots, BN_{PK_{max}}$, in our implementation; here PK_{max} is an integer parameter. The model $BN_n, \quad \forall n = 2, \ldots, PK_{max}$, is trained using the first n packets of each flow.

In the classification stage, classification probabilities are calculated using BN_n if the number of packets received is n. The strategy we employ for classification is summarized in the flowchart in Fig. 6.1. The process in this flowchart is continued

until the end of the flow or PK_{max} number of packets are received. If either of these conditions—end of the flow or PK_{max}-th packet is detected—then the flow is classified with the class with the highest belief. Consequently, the approach never uses more than PK_{max} number of packets in classifying a flow.

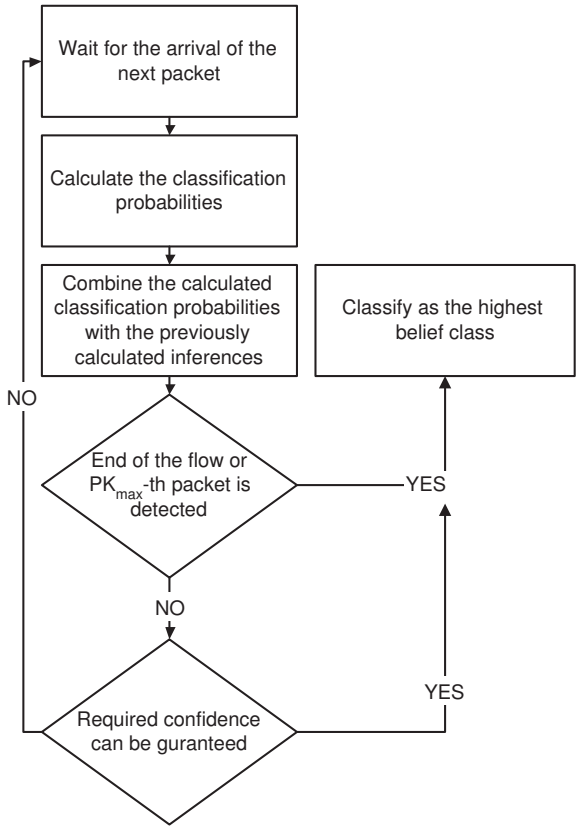

Figure 6.1: Flow Chart of the Proposed Approach.

6.2.1 Feature Selection

Feature selection is one of the most important steps in the classification process. Tow major requirements dictate the feature selection:

- One should be able to either directly measure or estimate each feature adequately well prior to flow termination.

- One must also be able to update the selected features as the flow progresses because of the availability of more information.

The first requirement excludes features such as flow duration, total number of packets per flow, total number of packets in the forward/backward direction, etc. With these requirements in mind, we have selected several appropriate features from the features proposed in [AMG07,BTA+06,MZ05,WZA06]. The set of features that were identified as appropriate for BN_n appears in Table 6.1.

6.2.2 Bayesian Network (BN)

As mentioned earlier, we used a sequence of BNs in the proposed algorithm. A BN is a graphical representation of probabilistic dependencies of a set of variables. This model consists of a set of nodes and links connecting those nodes. Nodes of the model represent variables and the links represent conditional probabilities defining the relationships between variables [Pea88]. One prominent advantage of the BN over the conventional naïve Bayes method is its ability to take dependencies among features into account.

Table 6.1: Set of Features Selected

	Feature	Symbol
Packet size:	– size of packet n	pkt_size_n
	– size of first packet (only if $n = 2$)	pkt_size_1
	– size of forward packets $1, \ldots, n$	$pkt\_size\_f\_1, \ldots, n$
	– size of backward packets $1, \ldots, n$	$pkt\_size\_b\_1, \ldots, n$
Average Packet size:	– average	$mean\_pkt\_size$
	– average in forward direction	$mean\_pkt\_size\_f$
	– average in backward direction	$mean\_pkt\_size\_b$
Minimum packet size:	– minimum	min_pkt_size
	– in forward direction	$min\_pkt\_size\_f$
	– in backward direction	$min\_pkt\_size\_b$
Maximum packet size:	– maximum	max_pkt_size
	– in forward direction	$max\_pkt\_size\_f$
	– in backward direction	$max\_pkt\_size\_b$
Inter-arrival Time:	– inter-arrival time $n - 1$	$int\_arr\_n - 1$
	– forward inter-arrival time $1, \ldots, n - 1$	$int\_arr\_f\_1, \ldots, n - 1$
	– backward inter-arrival time $1, \ldots, n - 1$	$int\_arr\_b\_1, \ldots, n - 1$
Average Inter-arrival	– average	$mean\_int\_arr$
Time:	– average in forward direction	$mean\_int\_arr\_f$
	– average in backward direction	$mean\_int\_arr\_b$
Minimum Inter-arrival	– minimum	min_int_arr
Time:	– in forward direction	$min\_int\_arr\_f$
	– in backward direction	$min\_int\_arr\_b$
Maximum Inter-arrival	– maximum	max_int_arr
Time:	– in forward direction	$max\_int\_arr\_f$
	– in backward direction	$max\_int\_arr\_b$
Magnitude of Fourier	– components $1, \ldots, n$	$pkt\_size\_ft\_1, \ldots, n$
Transform of	– forward direction components $1, \ldots, n$	$pkt\_size\_ft\_f\_1, \ldots, n$
Packet Sizes:	– backward direction components $1, \ldots, n$	$pkt\_size\_ft\_b\_1, \ldots, n$
Magnitude of Fourier	– components $1, \ldots, n - 1$	$int\_arr\_ft\_1, \ldots, n - 1$
Transform of	– forward direction components $1, \ldots, n - 1$	$int\_arr\_ft\_f\_1, \ldots, n$
Inter-arrival Times:	– backward direction components $1, \ldots, n - 1$	$int\_arr\_ft\_b\_1, \ldots, n - 1$
Standard Deviations:	– packet size	pkt_size_sd
	– forward size	$pkt\_size\_sd\_f$
	– backward size	$pkt\_size\_sd\_b$
	– inter-arrival time	int_arr_sd
	– forward inter-arrival time	$int\_arr\_sd\_f$
	– backward inter-arrival time	$int\_arr\_sd\_b$
Last Packet:	– is this the last packet	$last\_pkt$

Nodes of a BN used for Internet traffic classification represent features; links of the BN represent relationships between features. The conditional probabilities that relate two features are given via conditional probability tables (CPTs). So, each feature has a CPT that defines the relationship between the feature and its parent features. The topology of the BN is determined basically using domain knowledge. The topology of the BN we used in this work is given in Fig. 6.2. We have considered the following relationships in deriving this topology.

- Relationship of the overall mean packet size with mean packet size in the forward direction, mean packet size in the backward direction, standard deviation of packet sizes and individual packet sizes.

- Relationship of the overall mean inter-arrival time with the mean inter-arrival time in the forward direction, mean inter-arrival time in the backward direction, standard deviation of inter-arrival times and individual inter-arrival times.

- Relationship of the mean packet size in the forward direction with individual packet sizes in the forward direction and standard deviation of packet sizes in the forward direction.

- Relationship of the mean packet size in the backward direction with individual packet sizes in the backward direction and standard deviation of packet sizes in the backward direction.

- Relationship of the mean inter-arrival time in the forward direction with individual inter-arrival times in the forward direction and standard deviation of inter-arrival times in the forward direction.

- Relationship of the mean inter-arrival time in the backward direction with individual inter-arrival times in the backward direction and standard deviation of inter-arrival times in the backward direction.

- Relationship of overall maximum packet size with maximum packet size in the forward direction and maximum packet size in the backward direction.

- Relationship of overall minimum packet size with minimum packet size in the forward direction and minimum packet size in the backward direction.

- Relationship of overall maximum inter-arrival time with maximum inter-arrival time in the forward direction and maximum inter-arrival time in the backward direction.

- Relationship of overall minimum inter-arrival time with minimum inter-arrival time in the forward direction and minimum inter-arrival time in the backward direction.

In the topology given in Fig. 6.2, *class* is the root node. The variable *class* can take the values $1, \ldots, N$, where N is the number of classes in the classification. The variable $class = m$ means the flow belongs to class m. In the above topology, we made the reasonable assumption that all features are dependent on *class*.

The training process of BNs involves the estimation of the CPTs resident at each BN node and the estimation of the prior probabilities of the root node *class*. The maximum likelihood estimation (MLE) method is used for this purpose [Bun96]. Flows with at least n number of packets are used for the training of BN_n. Note that, the root node *class* is inherently discrete valued; if the number of traffic flow classes is *num_cls*, *class* may assume *num_cls* number of possible values. On the other

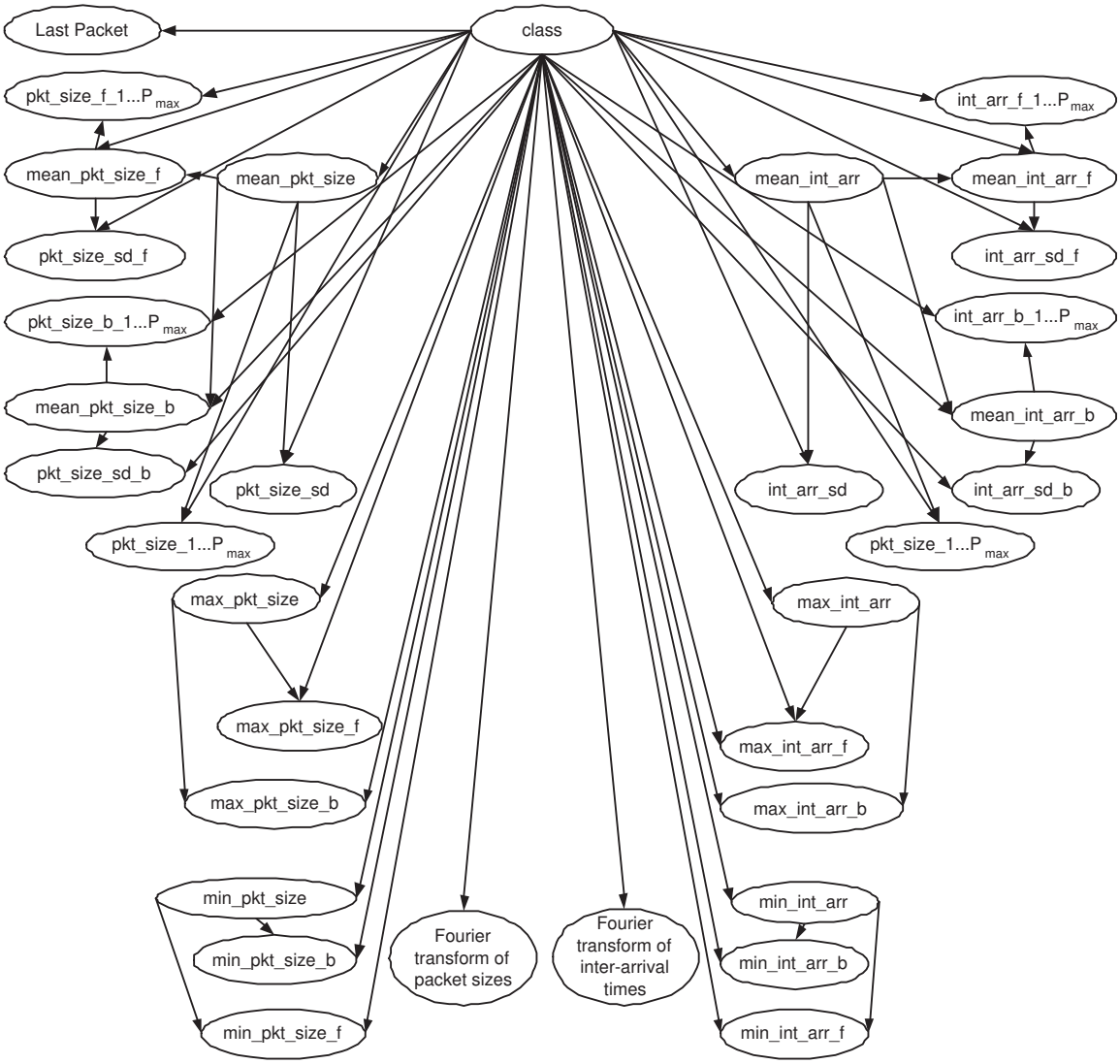

Figure 6.2: Topology of the BN.

hand, all other remaining features except *last_pkt*, which has two possible values, are continuous valued. Hence, it is necessary to discretize those feature values in order to utilize MLE. All of these features are discretized into *dis_num* number of discrete levels.

Uniform discretization is not appropriate in the current application since the feature values are unlikely to be uniformly distributed. Hence, we adopt a frequency based discretization where the range of values of each feature is discretized into

dis_num number of levels such that each level has approximately the same number of instances [WF99, ZSP+04].

6.2.3 Flow Classification

To explain the proposed flow classification algorithm, we use the integer valued subscript n to identify the window size. For example, the feature value vector immediately after the reception of the n-th packet is W_n. Suppose that the flow is not yet classified with the given confidence level at the end of $n-1$ packets. Then, with the arrival of the next packet, the operation of the proposed flow classification algorithm can be summarized via the following steps.

Step 1: Calculate W_n

In this step, the feature value vector W_n for the first n packets of a flow is calculated.

Step 2: BN Evidence Propagation

In this step, we use the calculated W_n and the usual BN evidence propagation mechanism [Pea88] to obtain the following posterior classification probabilities using BN_n:

$$P_n^{(m)} \equiv P(class = m|W_n),\ m = 1, \ldots, num\_cls. \tag{6.3}$$

Let us consider a simple example in order to explain the methodology used in the calculation of $P_n^{(m)}, \quad \forall m = 1, \ldots, num\_cls$. Let us assume that $num\_cls = 2$, $dis\_num = 2$ and consider only *mean_pkt_size* and *mean_pkt_size_f* in the process of classification. Let us consider a flow with *mean_pkt_size* $= 1$ and *mean_pkt_size_f* $=$

2. Now the classification probability $P_n^{(m)}$ for the class m can be calculated as

$$P_n^{(m)} = \frac{JP_n^{(m)}}{JP_n^{(1)} + JP_n^{(2)}}, \tag{6.4}$$

where

$$JP_n^{(m)} = P(class = m) \times P(mean\_pkt\_size = 1 | class = m) \times$$
$$P(mean\_pkt\_size\_f = 2 | mean\_pkt\_size = 1, class = m).$$

Corresponding conditional probabilities are stored in the CPTs of BN_n.

Step 3: Evidence Update

In this step, we update the previously calculated classification inferences with the newly calculated classification probabilities. To formulate the update mechanism within the DS theoretic framework, consider the FoD $\Theta = \{1, 2, \ldots, num\_cls\}$, where each singleton denotes a flow class. Recognizing that some portion of the calculated classification probability should be reduced and used to represent uncertainties, we employ the following mass structure on Θ for $C \subset \Theta$:

$$M_n(C) = \begin{cases} \gamma_n \, P_n^{(m)}, & \text{for } C = \{m\} \in \Theta; \\ \sum_{k=1}^{num\_cls} (1 - \gamma_n) \, P_n^{(k)}, & \text{for } C = \Theta; \\ 0, & \text{otherwise.} \end{cases} \tag{6.5}$$

Here, $\gamma_n \in [0, 1]$ is a pre-defined constant that enables one to conveniently capture the uncertainty associated with the computed classification probability values for the first n packets. We have shown in the simulation that the γ_n values should be selected appropriately in order to maximize the classification performances. Furthermore, we have implemented a scheme to select the appropriate γ_n values (see Section 6.2.4).

Notice that $M_n(\Theta)$ denotes complete ambiguity and is a recognition of the lack of sufficient evidence to decide on *any* class label for the flow. We also note that the BoE associated with the above mass assignment is $\{\Theta, F_n, M_n(\bullet)\}$, where the core F_n is allowed to have elements from $\{1, 2, \ldots, num\_cls, \Theta\}$, only. Hence, this is a Dirichlet mass assignment [JE07].

The BoE $\{\Theta, F_n, M_n(\bullet)\}$ captures the evidence one gathers from n packets. This evidence must now be combined with the BoE $\{\Theta, \mathfrak{F}_{n-1}, \mathfrak{M}_{n-1}(\bullet)\}$ that has already been constructed from previous packets. We use DCR for this purpose and recursively generate the mass structure

$$\mathfrak{M}_n(\bullet) = (\mathfrak{M}_{n-1} \oplus M_n)(\bullet), \quad \text{with } \mathfrak{M}_1(C) = \begin{cases} 1, & \text{for } C = \Theta; \\ 0, & \text{otherwise.} \end{cases} \tag{6.6}$$

Note that the recursion is initiated with the vacuous mass structure $\mathfrak{M}_1(\bullet)$.

Since the core may consist of only the singletons and complete ambiguity, the focal elements that the DCR generates also possess the same feature. Hence, we will use $\mathfrak{F} \equiv F_n = \mathfrak{F}_n, \ \forall n$.

Step 4: Classification Decision

Once the classification inferences are updated, we can make the classification decision. The classification decision takes the following form:

$$class = \begin{cases} \arg\max_m Bel_n(m) & \text{if } \max_m Bel_n(m) \geq \psi_n; \\ & \text{or } n = PK_{max}; \\ \text{Repeat from Step 1,} & \text{otherwise.} \end{cases} \tag{6.7}$$

The parameter ψ_n is a pre-defined threshold value that can be interpreted as the confidence one may place on the decision when n number of packets are used. If the

classification cannot be concluded with the required confidence, then the classifier goes to Step 1 and continues all the steps until classification is done with the given confidence, or until $n = PK_{max}$.

6.2.4 Parameter Estimation

As mentioned in Section 6.2.3, $\gamma_n, \quad \forall n = 2, \ldots, PK_{max}$, in (6.5) and $\psi_n, \quad \forall n = 2, \ldots, PK_{max}$, in (6.7) are parameters of the proposed flow classification algorithm. Let us define the vectors $\Gamma = [\gamma_2, \ldots, \gamma_{PK_{max}}]$ and $\Psi = [\psi_2, \ldots, \psi_{PK_{max}}]$. Moreover, define the vector $\mathbf{f} = [\Gamma \quad \Psi]$. It is noticeable from the simulation results that the performance of the classification algorithm directly depends on the selection of these parameters in $\mathbf{f}$. One can identify the classification accuracy and the required number of packets to perform the classification as two main performance measures. Hence, it is vital to select $\mathbf{f}$ in a way such that the classification accuracy is maximized while the required number of packets is minimized. However, it is obvious that these two objectives cannot be achieved simultaneously. In other words, classification accuracy can be maximized by increasing the required number of packets and the required number of packets can be decreased by sacrificing classification accuracy. To proceed, let us define the accuracy A as follows:

$$A = \frac{\#\text{ of correctly classified flows}}{\#\text{ of total flows}} \times 100\%. \tag{6.8}$$

The quantity A is used in the Internet traffic classification literature [AMG07,BTA+06, MZ05,WZA06] to measure the performance of classification algorithms. Let us define N as the average number of packets required for the classification.

First we have to define an objective function L which defines the overall performance that must be maximized. Obviously, the value of L should depend on

$\mathbf{f}$. Moreover, the objective function should be calculated based on A and N. One possibility for the objective function can be written as

$$L(\mathbf{f}) = A - wN, \tag{6.9}$$

where w is a positive real number. The value of w determines the amount of emphasis placed upon the minimization of the required number of packets for the classification.

As shown in [Ber99], the gradient projection algorithm enables one to achieve the optimum $\mathbf{f}^*$ in an iterative manner. In order to have a unique optimal, $L(\mathbf{f})$ should be a concave function of the parameter vector $\mathbf{f}$. However, in this application, concavity is not guaranteed and consequently gradient projection will not necessary lead to the globally optimal $\mathbf{f}^*$.

The off-line parameter update iterations are indexed by t. To see how the gradient projection algorithm is used to update the set of parameters, consider the parameter update

$$\begin{aligned} \mathbf{f}(t+1) &= \left[\mathbf{f}(t) + \mu(t)\left[\overline{\mathbf{f}}(t) - \mathbf{f}(t)\right]\right]^+, \text{ with} \\ \overline{\mathbf{f}}(t) &= \mathbf{f}(t) + s(t)\,\nabla L(\mathbf{f}(t)), \end{aligned} \tag{6.10}$$

where $[f]^+ = f$ if $f \geq 0$ and 0 otherwise. The quantities $\mu(t)$ and $s(t)$ must be selected appropriately. A constant step size (i.e., $\mu(t) = 1$ and $s(t) = s$, where s is a constant) is a suitable selection. The modified algorithm then becomes

$$\mathbf{f}(t+1) = \left[\mathbf{f}(t) + s\,\nabla L(\mathbf{f}(t))\right]^+. \tag{6.11}$$

As noted above, the partial derivatives $\nabla L(\mathbf{f}(t)$ need to be calculated in order to implement the gradient projection algorithm. These partial derivatives are not readily available and need to be estimated. At each off-line parameter estimation iteration, the partial derivatives are calculated by randomly selecting samples from the training data set.

6.3 Experiments

We conducted experiments to ascertain the feasibility of the proposed approach to online traffic classification. These experiments are discussed in this section.

6.3.1 Data Sets

Trace files from *publicly available* NLANR traces [NLA] were used for the experiments. Two data sets, Auckland-VI and Leipzig-II, were used for algorithm verification purposes. All flows in the data set were divided into 11 classes so that $num\_cls = 11$. It was impossible to determine the true application of the flows since these publicly available trace files do not provide application layer data. Hence, port-based classification is utilized to establish the "ground truth" true type of the flow. Although this process may lead to a few incorrect labels, we believe that it can be considered sufficiently accurate for verification of the proposed algorithm [EMA06, WZA06]. The flow class distribution for each data set appears in Table 6.2.

Training and Testing Environments

Subsets from the two data sets were selected as follows for training and testing.

- **Auckland-VI:** Twelve hours of data (from 2001-06-10 12:00 to 2001-06-11 00:00) were selected for training. Twelve hours in the next day (from 2001-06-11 12:00 to 2001-06-12 00:00) were selected for testing.

- **Leipzig-II:** Two and half hours of data (from 2003-02-21 15:00 to 2003-02-21 17:30) were selected for training. Two and half hours in the next day (from 2003-02-22 15:00 to 2003-02-22 17:30) were selected for testing.

The parameter $dis\_num = 20$ is used for the experiments.

Table 6.2: Flow Class Distributions of the Data Sets

Class	Flow Type	% of Flows		Avg Length	
		Auckland-VI	Leipzig-II	Auckland-VI	Leipzig-II
1	HTTP	75.75	45.70	25.11	33.37
2	SMTP	1.84	0.46	63.94	24.61
3	DNS	5.23	3.26	14.33	6.86
4	HALF-LIFE	2.46	1.65	7.09	189.46
5	POP3	0.33	0.67	30.00	31.63
6	FTP	0.12	0.18	180.48	101.93
7	IRC	0.56	0.08	3.29	2.94
8	NAPSTER	0.76	0.03	47.11	14012.89
9	AOL	0.16	0.11	13.00	45.91
10	All other TCP	12.38	28.97	69.01	47.98
11	All other UDP	0.42	18.89	287.89	23.17
	Overall	100.00	100.00	29.58	42.62

6.3.2 Selection of a Value for the Parameter PK_{max}

The integer value PK_{max} is a parameter used in the proposed approach. It is the maximum number of packets that can be used for the classification. First, it is required to select an appropriate value for this parameter. We conducted our first set of experiments to find an appropriate value for this parameter. For this set of experiments, we set $\gamma_n = 0.5, \quad \forall n$ and $\psi_n = 1, \quad \forall n$. Here, $\gamma_n = 0.5$ means we assign half of the calculated classification probabilities to represent the uncertainty. Moreover, $\psi_n = 1$ means we continue until the end of the flow or the PK_{max}-th packet is received since the belief cannot exceed one. Then we conducted experiments for different values of PK_{max} and calculated the overall classification accuracies. Results of these experiments are given in Fig. 6.3. It is clear from the results that the accuracy increases with the value of PK_{max}. Furthermore, note that further improvement in accuracy cannot be achieved after $PK_{max} = 8$ for the Auckland-VI data

set and $PK_{max} = 6$ for the Leipzig-II data set. In order to make sure that further improvement cannot be achieved for both the data sets, we set the value of PK_{max} to 10.

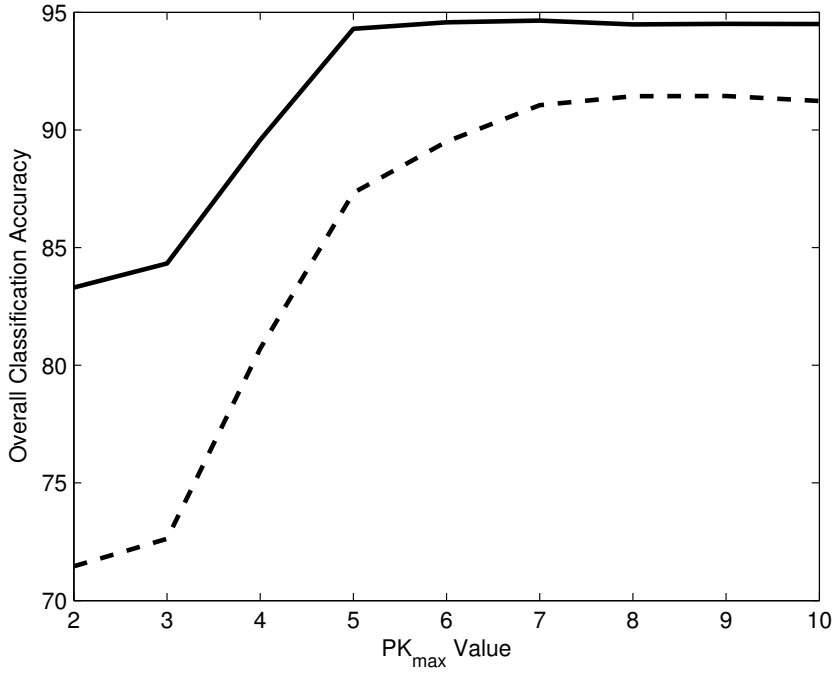

Figure 6.3: Classification Accuracies for Different PK_{max} Values.

6.3.3 Results With Random Parameters

In the next set of experiments, the parameters of the flow classification algorithm ($\gamma_n, \quad \forall n = 2, \ldots, 10$, and $\psi_n, \quad \forall n = 2, \ldots, 10$) are selected randomly. The classification results for ten sets of *randomly* selected parameters for the Auckland-VI data set are given in Table 6.3. Similar results for the Leipzig-II data set are given in Table 6.4. It is clear from the results from tests on both data sets that both classification accuracy and the required number of packets to make a classification decision depend

on the selection of the parameters. Hence, it is clear from the results of this set of experiments that a trained set of parameters need to be used in order to achieve the best classification performances.

Table 6.3: Classification Performance for *RANDOMLY* Selected Parameters - Auckland-VI

RandomSet	1	2	3	4	5	6	7	8	9	10
γ_2	0.10	0.50	0.50	0.20	0.60	0.40	0.50	0.20	0.60	0.80
γ_3	0.10	0.70	0.20	0.90	0.50	0.60	0.00	0.60	0.10	0.50
γ_4	0.20	0.90	0.00	0.30	0.60	0.40	0.10	0.70	0.90	0.20
γ_5	0.20	0.50	0.20	0.50	0.50	0.80	0.40	0.10	0.80	0.70
γ_6	0.90	0.00	0.70	0.70	0.70	0.70	0.70	0.30	0.20	0.10
γ_7	0.20	0.30	0.90	0.10	0.40	0.20	0.30	0.30	0.20	0.20
γ_8	0.10	0.10	0.80	0.20	0.30	0.10	0.10	0.10	0.90	0.70
γ_9	0.50	0.50	0.10	0.60	0.20	0.80	0.10	0.80	0.90	0.70
γ_{10}	0.80	0.70	0.50	0.90	0.00	0.80	0.60	0.30	0.90	0.80
ψ_2	0.40	0.50	0.00	0.70	0.80	0.30	0.90	0.30	0.40	0.40
ψ_3	0.50	0.10	0.40	0.00	0.40	0.60	0.60	0.90	0.60	0.80
ψ_4	0.80	0.80	0.30	0.70	0.30	0.50	0.60	0.00	0.50	0.10
ψ_5	0.20	0.90	0.10	0.40	0.50	0.20	0.10	0.80	0.70	0.10
ψ_6	0.20	0.70	0.10	0.90	0.60	0.40	0.40	0.90	0.90	0.80
ψ_7	0.60	0.40	0.60	0.50	0.00	0.20	0.10	0.60	0.80	0.80
ψ_8	0.80	0.60	0.60	0.30	0.20	0.10	0.20	0.20	0.90	0.90
ψ_9	0.40	0.60	0.70	0.30	0.30	0.80	0.20	0.30	0.20	0.90
ψ_{10}	0.10	0.90	0.60	0.00	0.10	0.50	0.60	0.80	0.50	0.20
Accuracy	88.49	70.53	71.46	69.09	80.13	82.03	84.37	82.59	78.61	72.91
# of Pkts	4.85	2.95	2.00	2.95	3.12	2.59	4.82	3.89	2.34	2.14

Table 6.4: Classification Performance for *RANDOMLY* Selected Parameters - Leipzig-II

RandomSet	1	2	3	4	5	6	7	8	9	10
γ_2	0.60	0.20	0.10	0.60	0.10	0.00	0.00	0.50	0.10	0.00
γ_3	0.50	0.50	0.60	0.90	0.70	0.80	0.30	0.50	0.40	0.40
γ_4	0.30	0.50	0.60	0.00	0.60	0.50	0.30	0.20	0.70	0.80
γ_5	0.30	0.80	0.70	0.60	0.90	0.80	0.70	0.90	0.60	0.80
γ_6	0.30	0.20	0.00	0.60	0.40	0.50	0.60	0.80	0.20	0.20
γ_7	0.90	0.50	0.60	0.50	0.30	0.10	0.90	0.10	0.80	0.90
γ_8	0.30	0.20	0.70	0.90	0.00	0.30	0.40	0.30	0.20	0.40
γ_9	0.00	0.20	0.50	0.00	0.10	0.80	0.50	0.20	0.50	0.80
γ_{10}	0.00	0.80	0.30	0.60	0.90	0.70	0.90	0.50	0.20	0.40
ψ_2	0.40	0.20	0.20	0.70	0.70	0.30	0.70	0.80	0.90	0.10
ψ_3	0.10	0.50	0.20	0.10	0.90	0.10	0.90	0.90	0.20	0.40
ψ_4	0.70	0.20	0.10	0.50	0.10	0.50	0.40	0.20	0.40	0.00
ψ_5	0.60	0.90	0.70	0.00	0.40	0.00	0.60	0.30	0.90	0.70
ψ_6	0.20	0.70	0.00	0.50	0.30	0.90	0.80	0.50	0.20	0.90
ψ_7	0.20	0.60	0.50	0.40	0.30	0.30	0.60	0.30	0.50	0.40
ψ_8	0.90	0.70	0.30	0.90	0.00	0.30	0.90	0.80	0.00	0.00
ψ_9	0.80	0.30	0.20	0.00	0.70	0.40	0.10	0.50	0.10	0.10
ψ_{10}	0.80	0.70	0.60	0.80	0.80	0.50	0.40	0.60	0.00	0.70
Accuracy	83.62	90.59	80.84	82.36	88.57	57.59	71.59	86.76	81.37	70.71
# of Pkts	2.10	2.98	2.75	2.75	3.49	2.75	3.70	3.49	2.76	3.49

6.3.4 Results With Trained Parameters

In this set of experiments, the parameter estimation approach given in Section 6.2.4 is utilized to find the trained set of parameters. Note that the possible values for classification accuracy (0-100) and the required number of packets (1-10) are of the same order. So, the objective function given in (6.9) is used with $w = 1$. In the parameter estimation, 50 iterations are used. After the parameter estimation iterations, the resulting trained parameters are used in the classification algorithm. Classification results with the trained set of parameters for the Auckland-VI data set in terms of classification accuracy are given in Fig. 6.4 and in terms of required number of packets are given in Fig. 6.5. The corresponding results for the Leipzig-II data set are given in Fig. 6.6 and Fig. 6.7. It is clear from the results that for the Auckland-VI data set, it is possible to achieve over 91% classification accuracy by utilizing, on average, less than 6 packets for the classification. For the Leipzig-II data set, it is possible to achieve over 93% classification accuracy by utilizing, on average, less than 4 packets for the classification.

Next, we generated random training and testing data sets to conduct experiments. In the process of random data set generation, we divided all the flows into two sets randomly, one for training and one for testing. We have repeated this process five times for the Auckland-VI data set and five times for the Leipzig-II data set. Then we ran experiments with the randomly generated data sets. Average results for five randomly generated Auckland-VI data sets in terms of classification accuracy are given in Fig. 6.8 and in terms of required number of packets are given in Fig. 6.9. The corresponding results for the Leipzig-II data set are given in Fig. 6.10 and Fig. 6.11. These results also have the same pattern as in the results given in Fig. 6.4, Fig. 6.5, Fig. 6.6 and Fig. 6.7.

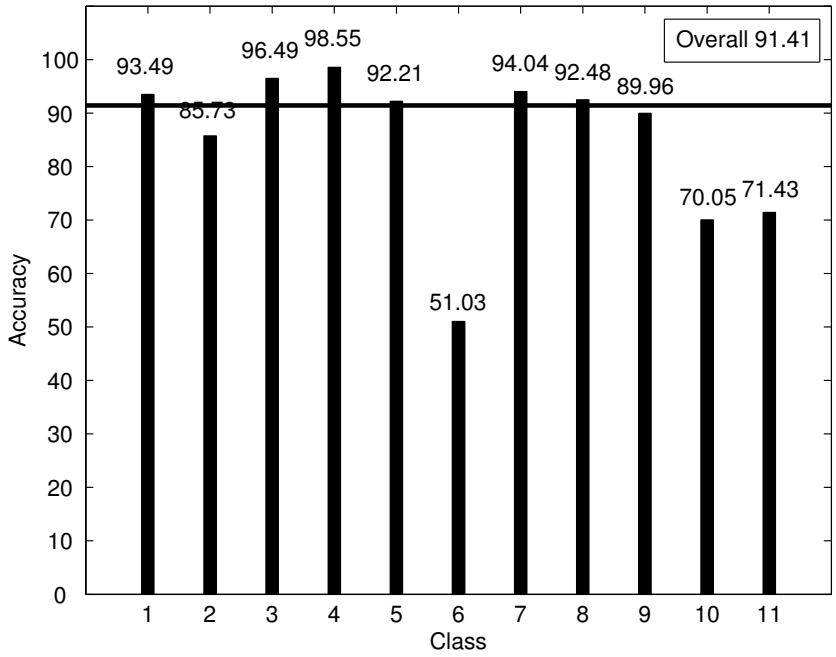

Figure 6.4: Classifier Performance - Auckland-VI: Classification Accuracy.

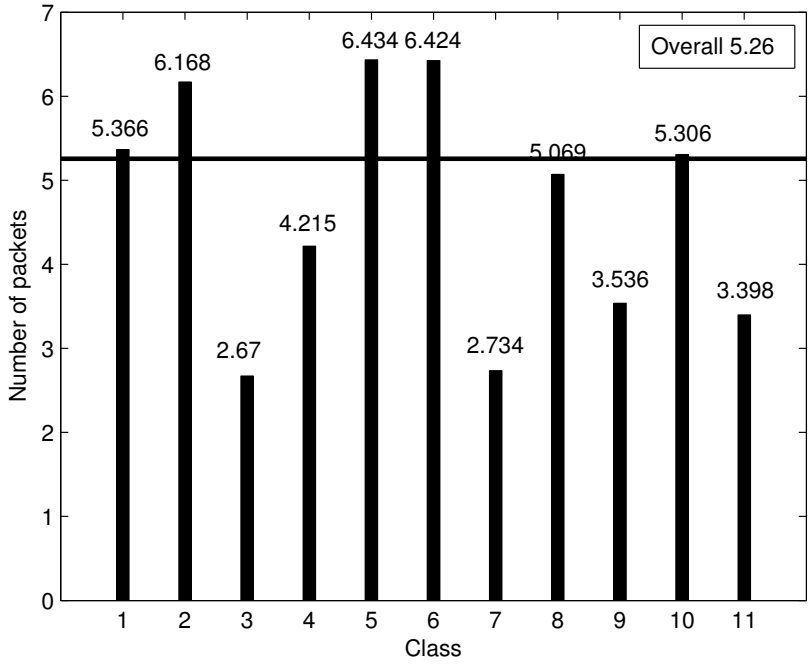

Figure 6.5: Classifier Performance - Auckland-VI: Average Required Number of Packets.

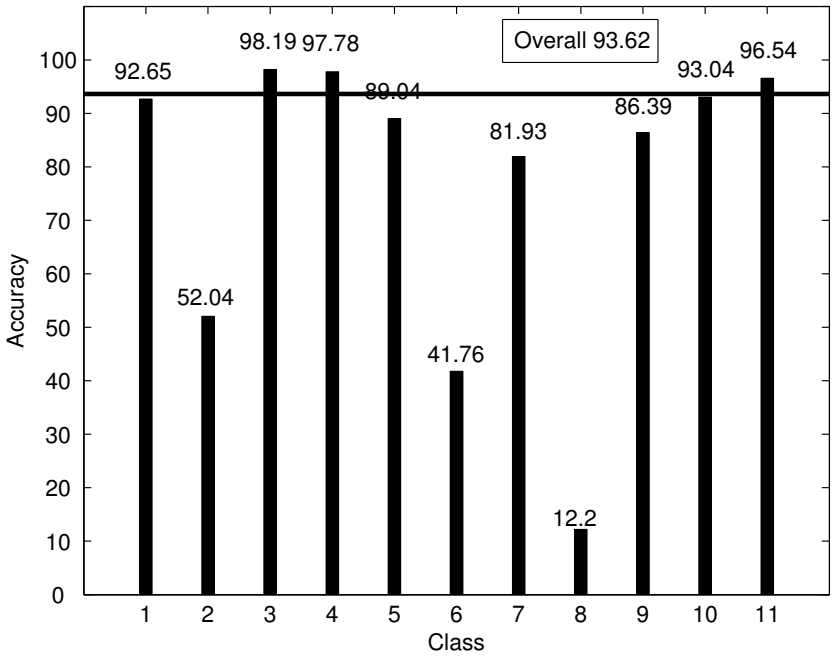

Figure 6.6: Classifier Performance - Leipzig II: Classification Accuracy.

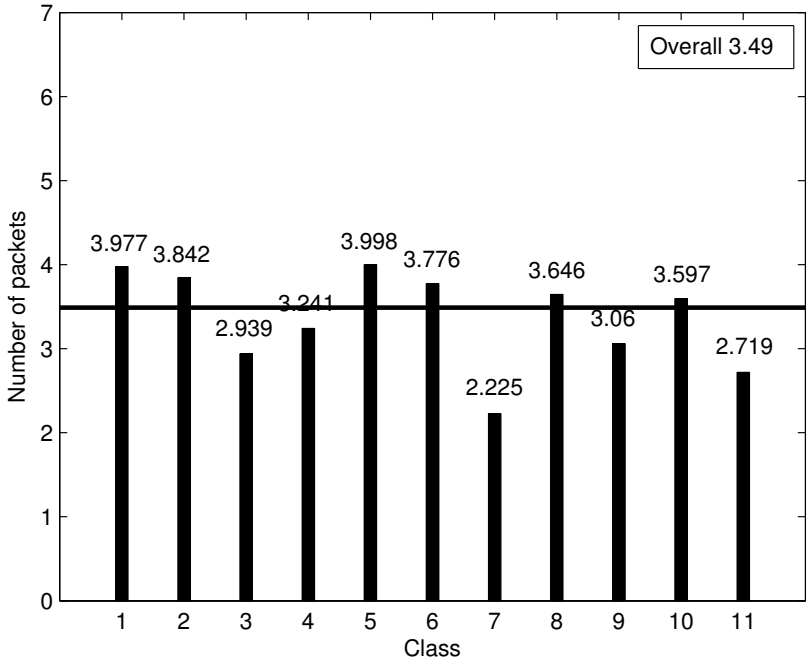

Figure 6.7: Classifier Performance - Leipzig II: Average Required Number of Packets.

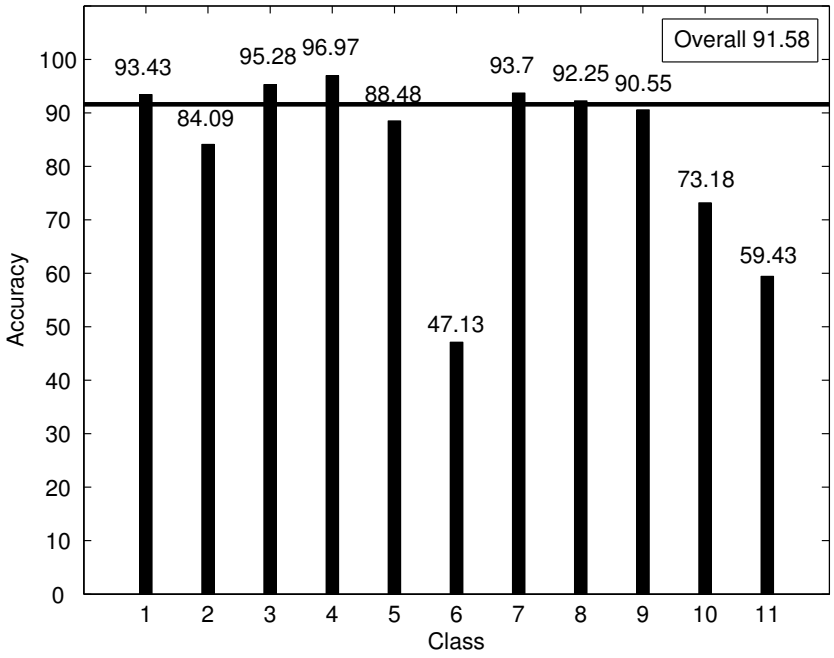

Figure 6.8: Classifier Performance with Random Data Sets - Auckland-VI: Classification Accuracy.

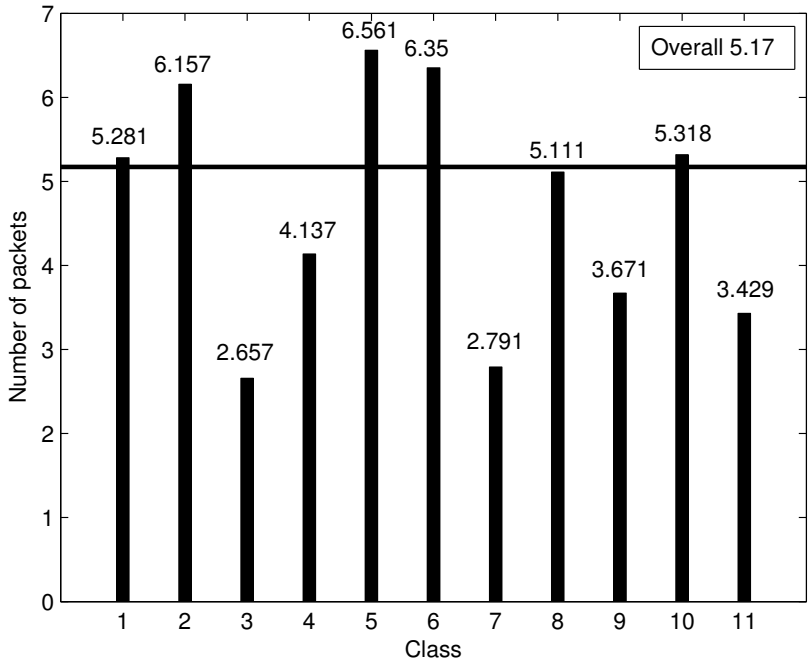

Figure 6.9: Classifier Performance with Random Data Sets - Auckland-VI: Average Required Number of Packets.

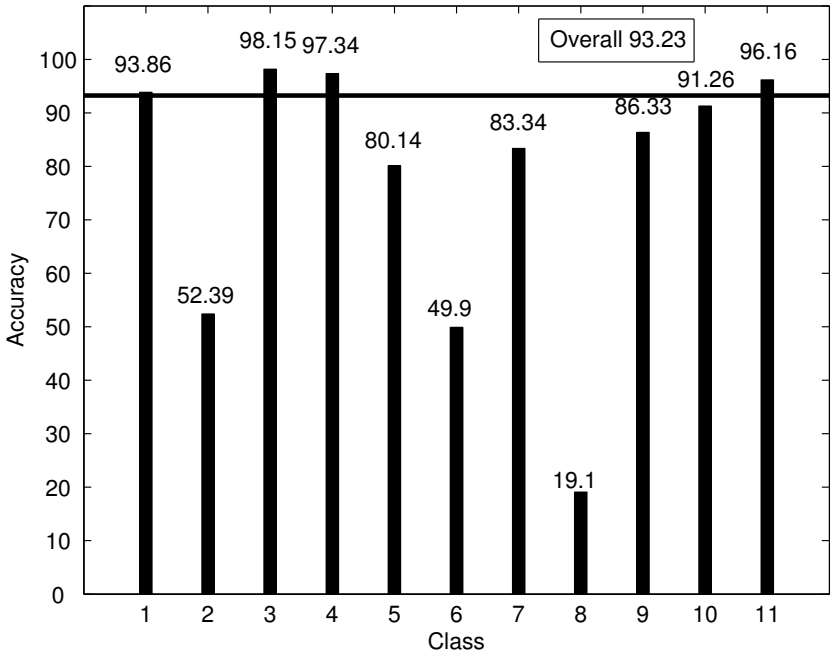

Figure 6.10: Classifier Performance with Random Data Sets - Leipzig II: Classification Accuracy.

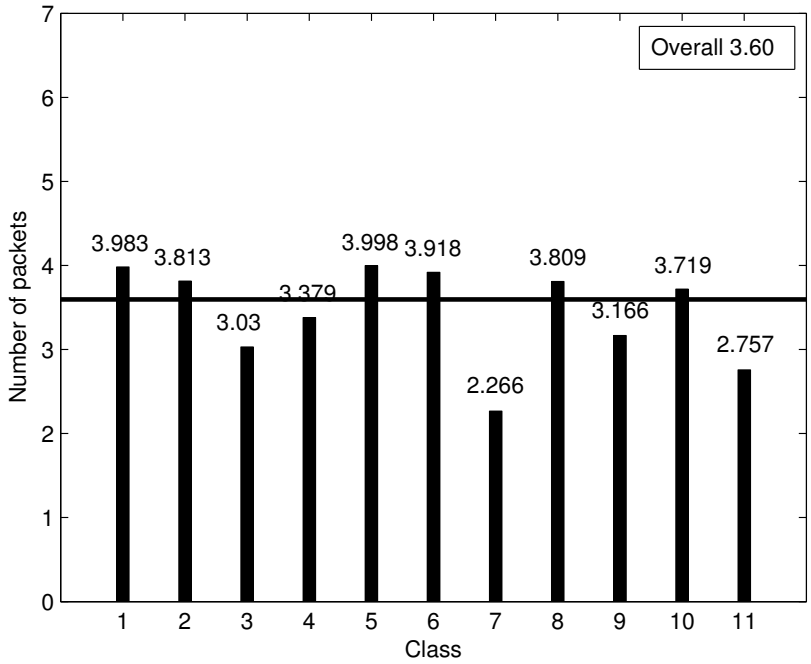

Figure 6.11: Classifier Performance with Random Data Sets - Leipzig II: Average Required Number of Packets.

6.3.5 Performance Comparison With Available Approaches

Classification accuracies for the considered data sets were compared with a few well-established classification algorithms. The objective of these experiments is to demonstrate the advantages of the proposed approach over the available approaches. It is difficult to compare the performance of the proposed approach with other approaches since the other approaches use the full flow information or information from a fixed number of packets. In order to perform a fair comparison, we have considered cases in which the other classification methods can only use the first m number of packets in the classification. Moreover, we have tested these other available classification approaches with different settings of m ($m = 2, \ldots, 10$). Three available classification approaches were tested for performance comparison.

1. **Naïve Bayes Method with Gaussian Estimation:**

   As discussed in 3.1.3 Naïve Bayes [MZ05] is often proposed to perform Internet traffic classification. In the experiment, conditional probability distributions were estimated with Gaussian distributions. Parameters of the distribution (mean and the standard deviation) are estimated in the training phase [MZ05]. This approach was originally proposed for full flow information based classification. However, in order to perform a fair comparison, we have calculated feature values for naïve Bayes using only the first m number of packets of the flow. Several experiments were conducted by selecting $m = 2, \ldots, 10$. The classification accuracies for naïve Bayes with Gaussian estimation for the Auckland-VI data set are given in Fig. 6.12. The corresponding results for the Leipzig-II data set are given in Fig. 6.13. It is clear from the results that our proposed algorithm has higher accuracies than the naive Bayes with Gaussian estimation for both

data sets while requiring only around five packets on average for Auckland-VI and around four packets on average for Leipzig II.

2. **Naïve Bayes Method with Discretized Feature Values:**

   It is clear from the results of [WZA06] that the classification accuracies can be improved by dicretizing the feature values. In this experiment, the naïve Bayes method is utilized with discretized feature values. The same discretized levels as used in the proposed algorithm are used in the experiment. The MLE Method is used for the conditional probability distribution estimation. Again, several experiments were conducted for $m = 2, \ldots, 10$. The classification accuracies for naïve Bayes with discretized feature values for the Auckland-VI data set are given in Fig. 6.14. The corresponding results for the Leipzig-II data set are given in Fig. 6.15. It is clear from the results that our proposed algorithm has higher accuracies for both data sets.

3. **k-Means Clustering:**

   Clustering algorithms have been used for partial flow based classification by several authors [BTA$^+$06, BTS06, EMA$^+$07]. We have compared the proposed approach with k-means clustering. Again, several experiments were conducted for different m values. In the training phase, training flows were clustered into 400 clusters [EMA$^+$07]. In the testing phase, for a new flow, the minimum distance cluster is determined by calculating the distance to the centroids of each cluster. Then the highest probable class within the minimum distance cluster is selected as the class of the new flow. The classification accuracies for k-means clustering for the Auckland-VI data set are given in Fig. 6.16. The corresponding results for the Leipzig-II data set are given in Fig. 6.17. The

overall accuracies of the k-means clustering algorithm are comparable with those of the proposed algorithm. However, it is clear in the results that the k-means clustering algorithm has very low classification accuracies for minor classes. Higher overall classification accuracies are mainly due to the high classification accuracies of major classes. When we consider classification accuracies for all of the classes, it is clear from the results that the proposed approach has better accuracies than k-means clustering.

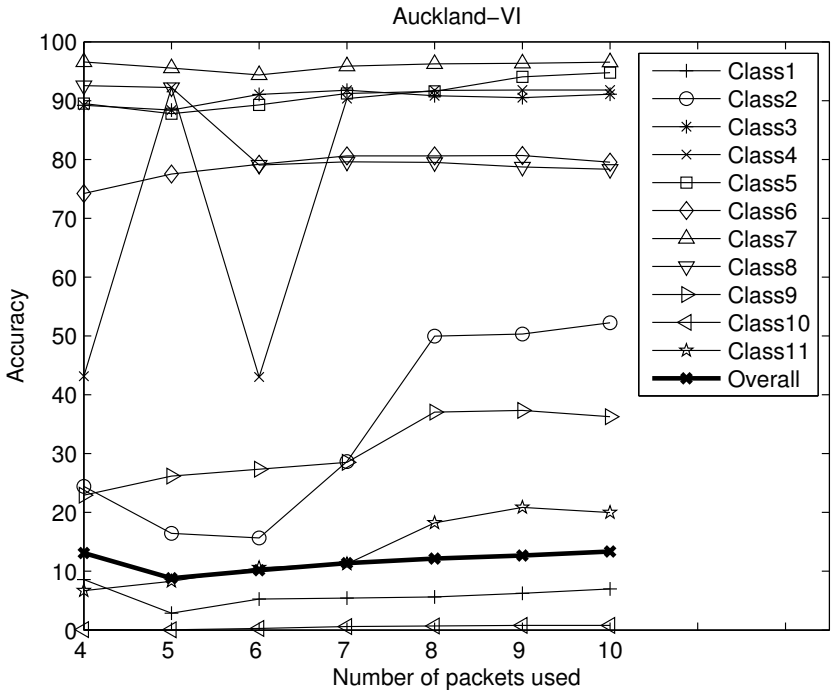

Figure 6.12: Classification Accuracies for Naïve Bayes with Gaussian Estimation - Auckland-VI.

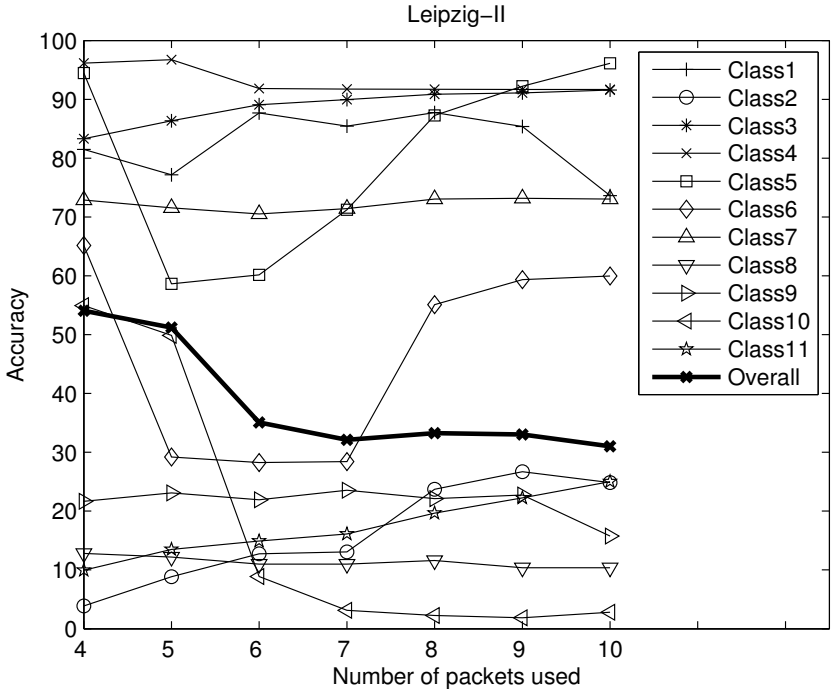

Figure 6.13: Classification Accuracies for Naïve Bayes with Gaussian Estimation - Leipzig-II.

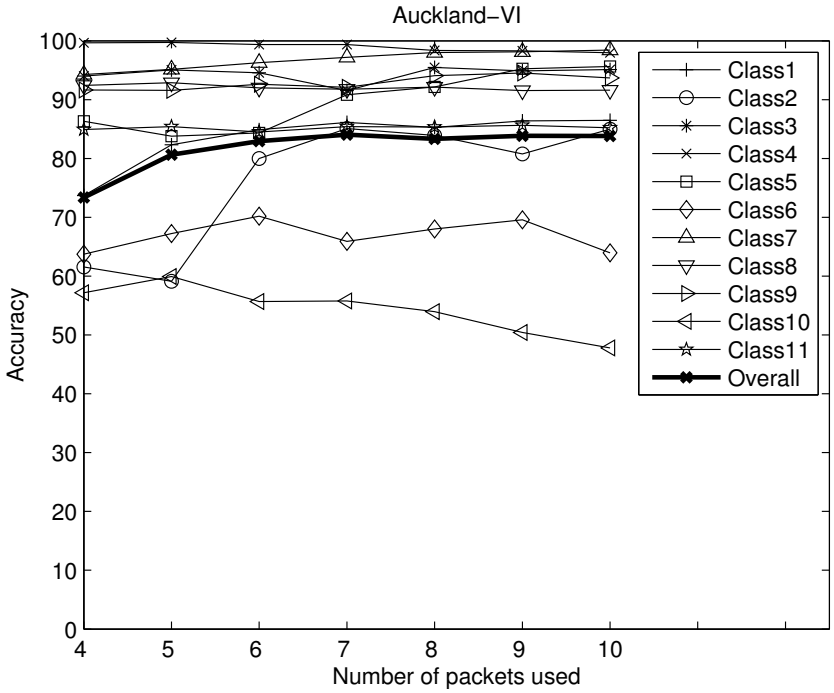

Figure 6.14: Classification Accuracies for Naïve Bayes with Discretized Feature Values - Auckland-VI.

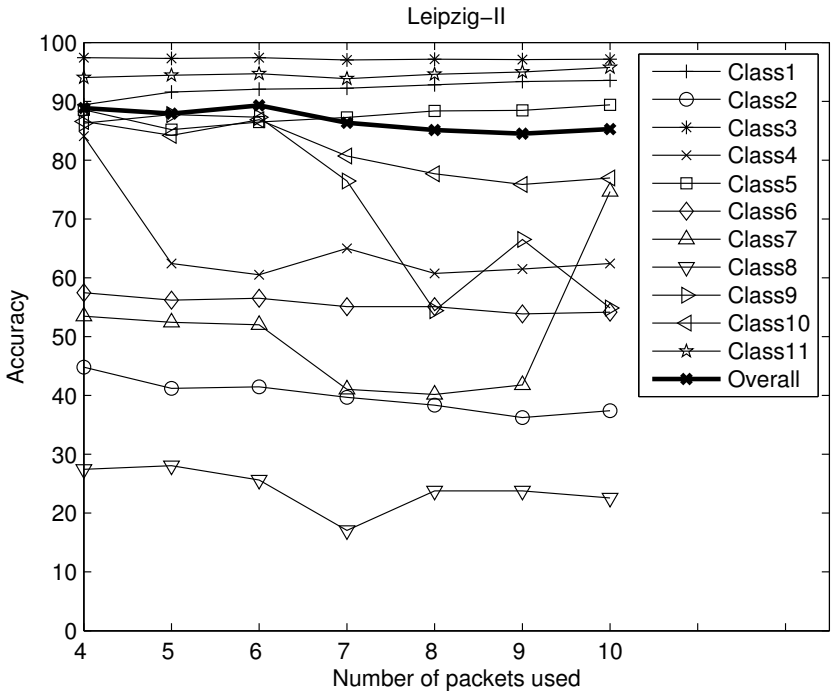

Figure 6.15: Classification Accuracies for Naïve Bayes with Discretized Feature Values - Leipzig-II.

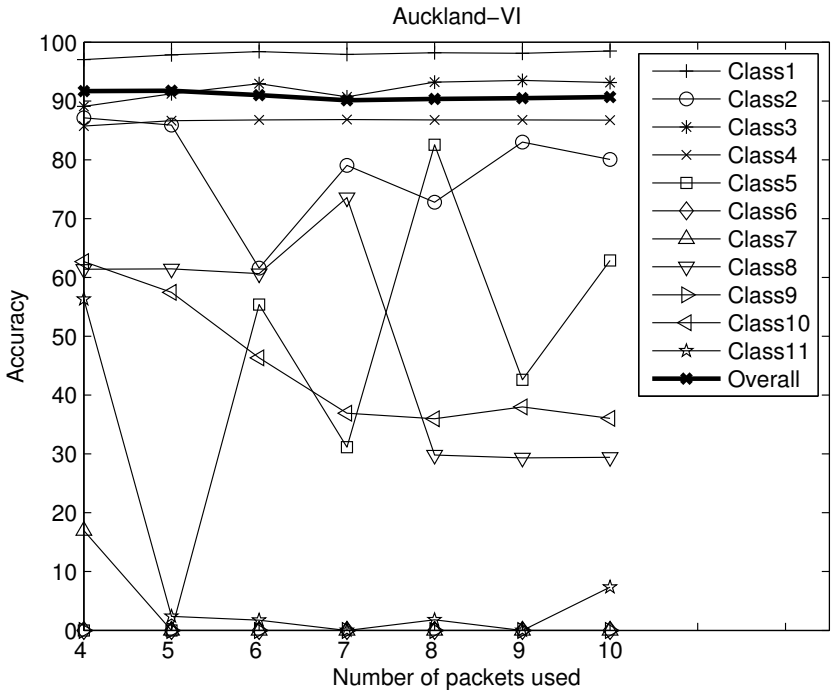

Figure 6.16: Classification Accuracies for k-Means Clustering - Auckland-VI.

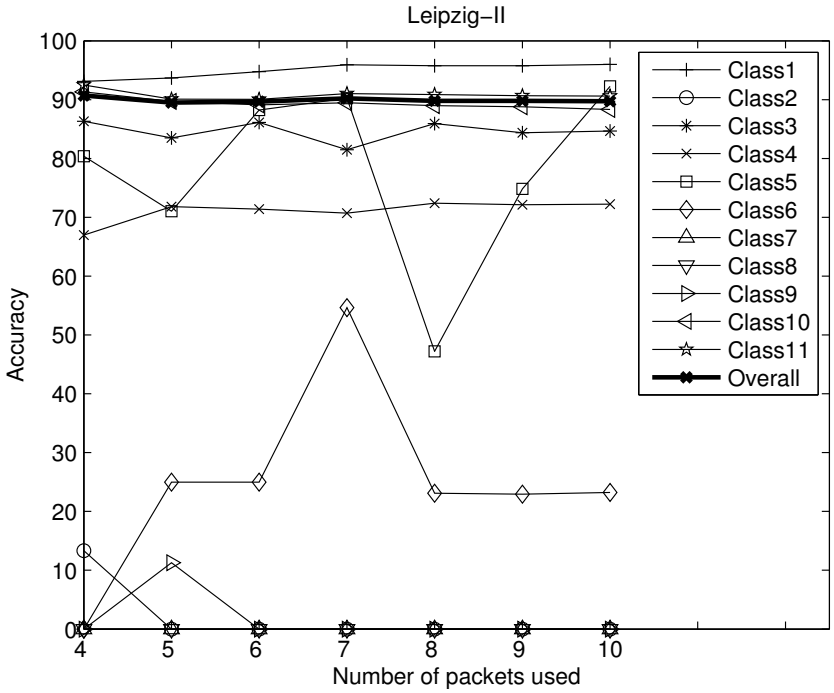

Figure 6.17: Classification Accuracies for k-Means Clustering - Leipzig-II.

6.4 Soft Decision for Classification of Minority Classes

It is clear from the results of the above section that some classes, particularly minority classes, have lower classification accuracies. Poor classification performance corresponding to minority classes is a stubbornly persistent problem associated with existing flow classification algorithms [AMG07, MZ05]. On the other hand, identification of minority classes with a reasonable prediction accuracy is of critical importance because potentially harmful and/or destructive flows do typically belong to minority classes. In this section, we exploit DS theory's capabilities to propose a soft decision strategy to improve the classification accuracies of minority classes. In soft decision, the classification decision is not necessarily a single class (singleton). Instead the decision can be a set of classes (composite proposition). If the classification decision is a set of classes, it indicates that the flow can belong to any class in the given set. For example, if the classification decision is $(1, 2)$, then the actual class to which the flow belongs to can be either 1 or 2. In DS theory, leaving the classification decision as the composite proposition $(1, 2)$ is indicative of the lack of sufficient evidence to discern between the two singletons 1 and 2.

In the previous approach, we assigned non-zero masses only to singletons, which is a single class, and to the whole set, which is the complete ambiguity. However, now we propose to assign non-zero masses to singletons, doubletons, which is a set of two classes, tripletons, which is a set of three classes, and to the whole set. The rationale behind the assignment of non-zero masses to composite propositions is that there may be situations where the available information is insufficient to make a singleton decision. In those situations we will be able to make a decision in favor of a composite proposition by assigning non-zero masses to composite propositions.

6.4.1 Modified Flow Classification

By comparing results from Fig. 6.4 and Fig. 6.6, one can see that lower accuracies for minority classes are more observable in the results of the Leipzig-II data set. So we will focus our discussion to the Leipzig-II data set. However we hasten to add that, one can apply the analytical results of this section to the other data sets as well. By analyzing the results in the Leipzig-II data set, we observe that classes 2, 6 and 8 (which constitute the minority classes) have lower classification accuracies compared to the other classes. Furthermore, we observed that most of the wrongly classified flows of the minority classes are classified into one of three majority classes, namely, *HTTP* (class 1), *All other TCP* (class 10) and *All other UDP* (class 11). By keeping the above observations in mind, we propose a new classification strategy, which can be summarized in the following steps:

Step 1: Calculate W_n

This step is identical to Step 1 of the previous strategy (see Section 6.2.3).

Step 2: BN Evidence Propagation

This step is similar to **Step 2** of the previous approach where $P_n^{(m)}, \quad \forall m = 1, \ldots, num\_cls$, values are calculated.

Step 3: Evidence Update

The following mass assignment is adopted in this approach.

$$M_n(C) = \begin{cases} K_{n,1}\,\gamma_n\,P_n^{(m)}, & \text{for } C = \{m\} \in \Theta; \\ K_{n,2}\,(1-\gamma_n)\,(P_n^{(m_1)} + P_n^{(m_2)}) & \text{for } C = \{m_1, m_2\}; \\ & \quad m_1 \in \Theta_{min}; m_2 \in \Theta_{maj}; \\ K_{n,3}\,(1-\gamma_n)\,(P_n^{(m_1)} + P_n^{(m_2)} + P_n^{(m_3)}) & \text{for } C = \{m_1, m_2, m_3\}; \\ & \quad m_1, m_2 \in \Theta_{min}; m_3 \in \Theta_{maj}; \\ K_{n,4} \sum_{k=1}^{num\_cls} (1-\gamma_n)\,P_n^{(k)}, & \text{for } C = \Theta; \\ 0, & \text{otherwise.} \end{cases} \tag{6.12}$$

Here, Θ_{min} is the set of minor classes and Θ_{maj} is the set of majority classes. The normalizing parameters $K_{n,1}$, $K_{n,2}$, $K_{n,3}$ and $K_{n,4}$ should be selected in a way that the above mass assignment is valid, in other words, summation over all of the masses should be one. Note the following regarding (6.12). Masses are assigned as follows:

- Single classes (e.g., (1)): This assignment represents the confidence one can place on a singleton proposition. This should be proportional to the calculated classification probability in order to better represent the confidence.

- Sets of two classes consisting of classes from both the minority and majority (e.g., $(1, 2)$ where 1 is a majority class while 2 is a minority class): This assignment represents the confidence one can place on a doubleton proposition. In other words, this assignment represents the confidence one can place on two classes even though sufficient information is not available to discern between two

classes. In this work, we are trying to model the unavailability of information to discern between minority and majority classes. Hence, in this assignment we should assign non-zero masses only to the doubletons with both minority and majority classes. Moreover, note that the assigned mass of a doubleton should consist of proportions of calculated classification probabilities of the two individual classes. Hence, we propose to assign a masses to doubletons in a way that the mass is proportional to the sum total of the individual classification probabilities of the classes within the doubleton.

- Sets of three classes consisting of classes from both the minority and majority (e.g., $(1, 2, 6)$ where 1 is a majority class while 2 and 6 are minority classes): This assignment represents the confidence one can place on a tripleton proposition. In other words, this assignment represents the confidence one can place on three classes even though sufficient information is not available to discern between three classes. Again, as we are trying to model the unavailability of sufficient information to discern between minority and majority classes, we assign non-zero masses only to the tripletons which are mixtures of both minority and majority classes. Furthermore, we have ignored tripletons with more than one majority class in it. For the same reason as explained in the above item, we propose to assign a masses to tripletons in a way that the mass is proportional to the sum total of the individual classification probabilities of the classes within the tripleton.

- Complete ambiguity Θ: This assignment represents the lack of information to assign masses to any proposition.

Note that the BoE associated with the above mass assignment is $\{\Theta, F_n, M_n(\bullet)\}$ which captures the evidence one gathers from n packets, where the core F_n is allowed to have doubletons and tripletons as well. In this approach also we once again use *DCR* to combine this evidence with the previously constructed BoE $\{\Theta, \mathfrak{F}_{n-1}, \mathfrak{M}_{n-1}(\bullet)\}$. Now the classification decision is achieved based on the BoE $\{\Theta, \mathfrak{F}_n, \mathfrak{M}_n(\bullet)\}$.

Step 4: Classification Decision

The classification decision can be either a singleton or a composite proposition. In order to proceed let us define following terms:

$$MS_1 = \arg \max_{C:|C|=1} m_n(C)$$

$$MV_1 = \max_{C:|C|=1} m_n(C)$$

$$MS_2 = \arg \max_{C:|C|=2} m_n(C)$$

$$MV_2 = \max_{C:|C|=2} m_n(C)$$

$$MS_3 = \arg \max_{C:|C|=3} m_n(C)$$

$$MV_3 = \max_{C:|C|=3} m_n(C)$$

Here, $m_n(C)$ is the mass of the set C based on BoE $\{\Theta, F_n, M_n(\bullet)\}$. The strategy that we employed to make the classification decision is given in Table 6.5.

Table 6.5: Decision Criteria

if ($n = PK_{max}$ or $n =$ length of the flow) **then**
  if $\max(MV_1, MV_2, MV_3) = MV_1$ **then**
    if $MS_1 \in \Theta_{maj}$ **then**
      if $MV_2 < \delta_1$ and $MV_2 < \delta_1$ **then**
        $Decision = MS_1$
      else
        if $MV_2 > \delta_2 MV_3$ **then**
          $Decision = MS_2$
        else
          $Decision = MS_3$
        end if
      end if
    else
      $Decision = MS_1$
    end if
  else
    if $MV_2 > \delta_2 MV_3$ **then**
      $Decision = MS_2$
    else
      $Decision = MS_3$
    end if
  end if
else
  if $\max(MV_1, MV_2, MV_3) = MV_1$ **then**
    if $MS_1 \in \Theta_{maj}$ **then**
      if $MV_2 < \delta_1$ and $MV_2 < \delta_1$ and $MV_1 > \psi_n$ **then**
        $Decision = MS_1$
      else
        Repeat form **Step 1**
      end if
    else
      if $MV_1 > \psi_n$ **then**
        $Decision = MS_1$
      else
        Repeat form **Step 1**
      end if
    end if
  else
    Repeat form **Step 1**
  end if
end if

In this approach, if the flow does not reach the PK_{max} packets maximum or the end of the flow, the algorithm makes a decision only if the maximum mass is assigned to a singleton. Rationale behind this is that, if we do not have sufficient information to make a singleton decision, we should wait for more packets to be received. Even though the maximum mass is assigned to a singleton, all composite propositions need to have significantly lower masses in order to classify a flow into a majority class. This is important to have a higher confidence on singleton decisions. On the other hand, if the flow reaches the PK_{max} packets maximum or the end of the flow, the algorithm must make a decision. This decision can be either a singleton or a composite. At this point also, in order to make a singleton decision, the maximum mass assigned proposition must be a singleton. Furthermore, in order to classify a flow as being from a majority class, all composite propositions must have significantly lower masses. This is important to minimize minority classes from being classified classify as a majority class. In order to make a doubleton decision, all tripletons must have significantly lower masses than the doubletons. Again this is important to improve the confidence of the doubleton decisions.

6.4.2 Results

Simulations were conducted with the Leipzig-II data set in order to evaluate the performance of the modified flow classification approach. The same set-up of training and testing as discussed in Section 6.3.1 is used for this simulation. The parameter PK_{max} is set to 10 in these simulations as well.

Since, it is possible to have composite propositions as the classification decision, it is important to analyze the distribution of decisions among singletons, doubletons and tripletons. This distribution is given in Fig. 6.18. In the figure, heights of black bars

correspond to the percentage of flows classified as singletons. Furthermore, heights of gray bars correspond to the percentage of flows classified as doubletons. Similarly, heights of white bars correspond to the percentage of flows classified as tripletons. In this figure, and all other figures in this section, minority classes are identified with a circle. Note that minority classes (classes 2, 6, and 8) have significantly higher gray and white bars. Hence, it is clear from the results that the minority classes have a higher percentage of flows classified as composite propositions. Moreover, note that classes 10 and 11 also have significantly higher gray and white bars. Hence, it is clear that a higher percentage of flows in classes 10 and 11 are also classified as composite propositions. Higher composite proposition percentages are expected for classes 10 and 11 since these two classes correspond to *All other TCP* and *All other UDP* respectively.

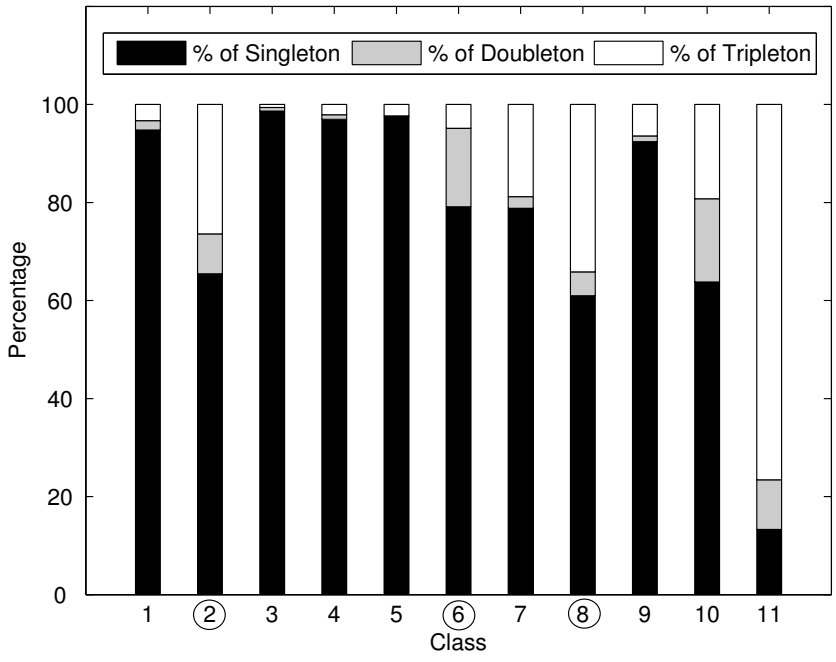

Figure 6.18: Distribution of decisions among Singletons, Doubletons and Tripletons.

Next, it is important to evaluate the classification performance. In this experiment it is inappropriate to calculate the classification accuracy, since the classification decision is not always a singleton. So, we calculate the percentage of incorrectly classified flows in order to evaluate the performance. If the decision of the classification does not contain the actual class, then we categorize it as an incorrectly classified flow. The percentage of incorrectly classified flows for individual classes and the overall incorrectly classified percentage are given in Fig. 6.19. It is clear from the results that the percentage of incorrectly classified flows is very low. Hence, it is clear that the modified approach has improved the performance, particularly for minority classes.

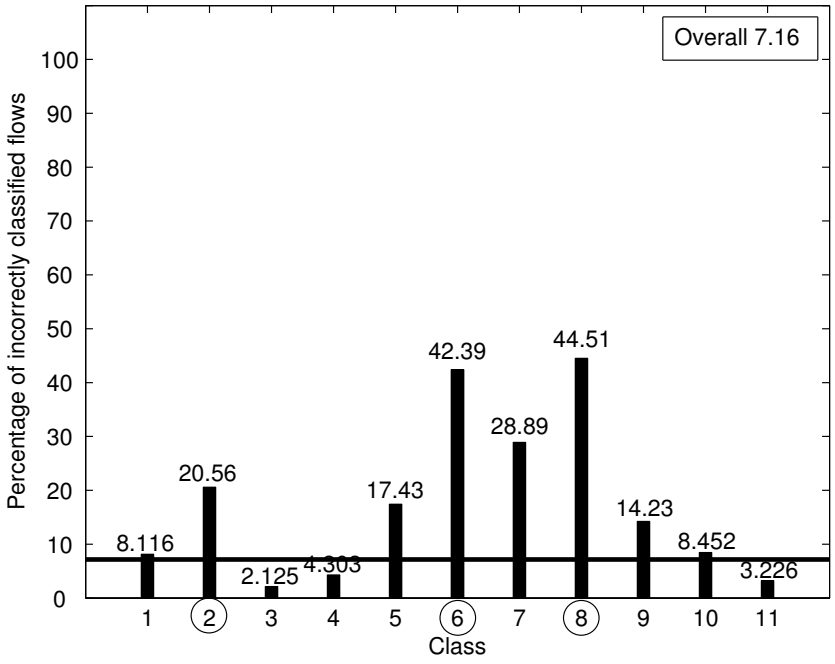

Figure 6.19: Percentage of incorrectly classified flows.

Our next experiment is based on randomly generated training and testing data sets. Here, we have randomly divided the Leipzig-II data set into two, one for training and one for testing. This process is repeated five times in order to generate five randomly generated data set pairs. Average results for randomly generated data sets are given in Fig. 6.20. These results also exhibit the same pattern as the results given in Fig. 6.19.

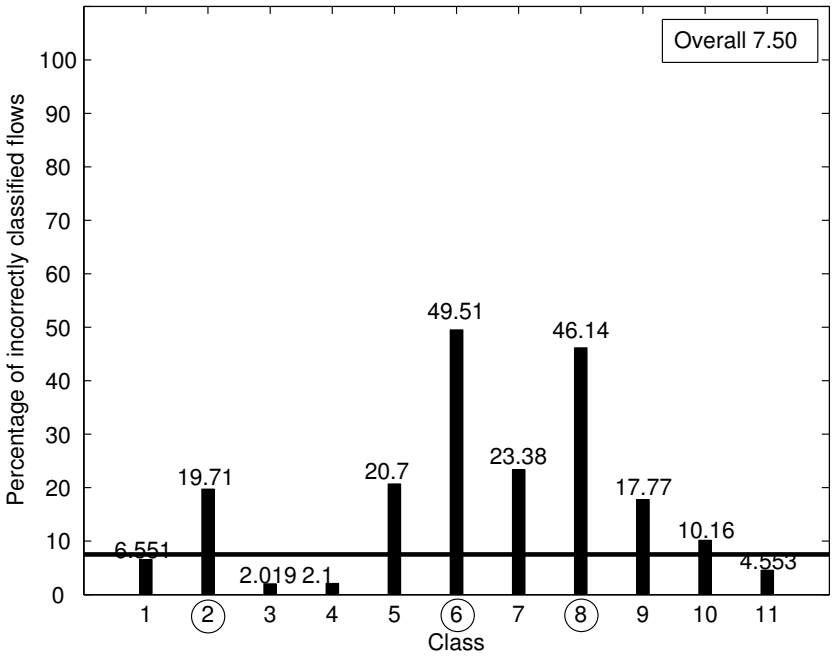

Figure 6.20: Percentage of incorrectly classified flows - Random Data Sets.

CHAPTER 7

Rate Allocation Among Set of Flows

A novel traffic classification algorithm is proposed in Chapter 6. Once the flows are adequately well classified, a rate allocation algorithm as explained in Chapters 4 and 5 can be utilized to optimally allocate the resources among flows of a particular class. In other words, rate allocation algorithms as explained in Chapters 4 and 5 can be utilized for intra-class resource allocation. Now, we should develop a methodology to allocate resources among a set of flows corresponding to different classes. In other words, we should develop a resource allocation algorithm to perform inter-class resource allocation.

We believe that one way to allocate the available bandwidth to the set of flows is by enforcing a rate ratio between different flows depending on the importance of those flows. For example, if we can classify flows into three classes ($class1$, $class2$, $class3$), then we can utilize the rate allocation algorithm to allocate rates in the ratios of, say, $class1 : class2 : class3 = 1 : 2 : 1$. In this chapter, we focus on how to utilize the rate allocation algorithm to maintain the rate ratio requirements between set of flows. First, a new utility function needs to be defined in order to take the rate ratio requirements into consideration. Then, a rate controller can be derived in the form of a window update function to maximize the aggregate utility of the network.

In the process of rata ratio allocation, it is desirable to consider a set of sources. This set of sources is named as a *coordination* group(CG). Each CG must consist of one 'sink' node and more than one sources participating in the rate coordination. These member sources of the CG are referred to as *coordinated flows*. When we consider a CG the 'sink' node decides the desirable rate ratio and it should send feedback to each member of the CG in order to adopt there transmission rates in a way that the required rate ratio is maintained. To further understand the situation, consider the shared network given in the Fig. 7.1. This network has two CGs CG#1 and CG#2. The group CG#1 has five members while CG#2 has only four. Further note that two members are common to both the CGs.

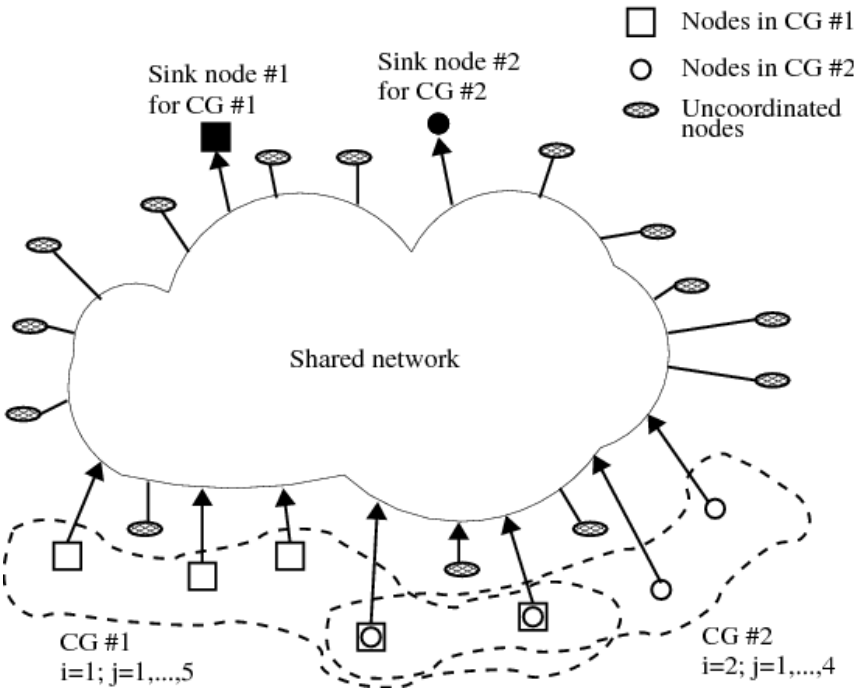

Figure 7.1: A scenario with 2 sink nodes with their corresponding CGs.

Now the objective is to find a rate control algorithm by which the transmission rate ratio between members of the group must be maintained. This same mechanism must be effective in establishing and maintaining the requested bit-rate ratios among

the nodes within this CG while ensuring that these *coordinated flows* can co-exist in a shared network with many different applications having potentially competing requirements. Traditional network congestion control methods such as TCP Reno and TCP Vegas are not designed to handle such requirements.

7.1 A New Utility Function

The first step towards the solution is to define a utility function by which the quality of the rate ratio maintenance can be measure. Note that it is important to have the utility function as a concave function of the transmission rates in order to directly use the results of NUM framework. To address the possibility of one source being a member of more than one CG (irrespective of whether they correspond to the same sink node or not), consider the following utility function for source j:

$$U_j(\mathbf{f}) = V_j(f_j) - \sum_{i\in\mathfrak{G}} K^{(i)}[D_j^{(i)}(\mathbf{f})]^2,\ \forall j \in \mathfrak{T}, \quad \text{where} \quad D_j^{(i)}(\mathbf{f}) = f_j - \gamma_j^{(i)} \sum_{k\in\mathfrak{G}^{(i)}} f_k. \tag{7.1}$$

The notation we adapt is the following: the sets $\mathfrak{T}$, $\mathfrak{C}$ and $\mathfrak{U}$ (with $|\mathfrak{T}| = |\mathfrak{C}|+|\mathfrak{U}|$) index the total, coordinated and uncoordinated sources in the entire network; $\mathbf{f}$ denotes the column vector containing the current source rates $f_j,\ j \in \mathfrak{T}$; $\mathfrak{G}$ is a set that indexes all the distinct CGs (of all the sink nodes); the set $\mathfrak{G}^{(i)}$ indexes the member sources belonging to CG $\#i,\ i \in \mathfrak{G}$; $\alpha > 0$, $K^{(i)} > 0$, $\gamma_j^{(i)} \in [0, 1]$ are all real parameters; and $\sum_{k\in\mathfrak{G}^{(i)}} \gamma_k^{(i)} = 1,\ i \in \mathfrak{G}$. For example, in Fig. 7.1, we have the following: $|\mathfrak{G}| = 2$ (# of CGs); $|\mathfrak{G}^{(1)}| = 5$ (# of members in CG #1); $|\mathfrak{G}^{(2)}| = 4$ (# of members in CG #2); and 2 flows are members of both CGs.

Now observe the following regarding (7.1). The first term $V_j(f_j)$ is a concave function of the source data rate f_j and it addresses rate maximization, and is from the

standard (original) utility function. The second term addresses rate ratio coordination within each CG; the term $D_j^{(i)}(\mathbf{f})$ indicates the difference between the rate of source f_j and its allocated ratio; the parameter $K^{(i)}$ determines the emphasis placed upon rate ratio maintenance over the standard utility maximization within $\mathfrak{G}^{(i)}$ ($K^{(i)} = 0$ whenever source j is either uncoordinated or does not belong to CG $\#i$); the parameter $\gamma_j^{(i)}$ determines the proportion of rate allocated to source $j \in \mathfrak{G}^{(i)}$ (e.g., for CG #2 in Fig. 7.1, $\{\gamma_1^{(2)}, \gamma_2^{(2)}, \gamma_3^{(2)}, \gamma_4^{(2)}\} = \{0.1, 0.1, 0.2, 0.6\}$ describes a rate allocation in the ratio of 1:1:2:6 to its member sources). Note that $D_j^{(i)}(\mathbf{j}) = 0,\ \forall j \in \mathfrak{G}^{(i)}$, when all rates achieve the ratio given by $\gamma_j^{(i)}$ values. When the specified rate ratios are not achieved, $|D_j^{(i)}(\mathbf{f})| > 0,\ \forall j \in \mathfrak{G}^{(i)}$, meaning that the utility of the coordinated sources is reduced. Therefore, in (7.1), the utility function reflects the twin goals of raw data rate maximization, and the attainment of the specified rate ratios defined by the $\gamma_j^{(i)}$ values. Moreover, for uncoordinated sources, the utility function reduces to the standard data transfer utility function. We express the network flow utility maximization problem as

$$\max_{\mathbf{f}\geq 0} \sum_{j\in\mathfrak{T}} U_j(\mathbf{f}) \quad \text{subject to} \quad \sum_{j\in S_\ell} f_j \leq C_\ell,\ \forall \ell \in \mathfrak{L}, \tag{7.2}$$

where $\mathfrak{L}$ is a set that indexes all the links in the network; S_ℓ and C_ℓ are the set of flows and capacity along the link $\ell \in \mathfrak{L}$. Since $U_j(\mathbf{f})$ is a concave function of $\mathbf{f}$, this optimization problem can be solved via the primal-dual approach with the duality gap zero [Ber99, LPW02, Low03]. So, consider the Lagrangian

$$\begin{aligned} L(\mathbf{f}, \mathbf{p}) &= \sum_{j\in\mathfrak{T}} U_j(\mathbf{f}) + \sum_{\ell\in\mathfrak{L}} p_\ell \left(C_\ell - \sum_{j\in S_\ell} f_j \right) \\ &= \sum_{j\in\mathfrak{T}} U_j(\mathbf{f}) + \sum_{\ell\in\mathfrak{L}} p_\ell \left(C_\ell - \sum_{j\in\mathfrak{T}} R_{\ell j} f_j \right) \\ &= \sum_{j\in\mathfrak{T}} \left(U_j(\mathbf{f}) - f_j \sum_{\ell\in\mathfrak{L}} R_{\ell j} p_\ell \right) + \sum_{\ell\in\mathfrak{L}} C_\ell p_\ell. \end{aligned} \tag{7.3}$$

Here, $\mathbf{p}$ is a column vector containing the link 'prices' $p_\ell,\ \ell \in \mathfrak{L}$; $R_{\ell j} = 1$ if source $j \in \mathfrak{T}$ uses link $\ell \in \mathfrak{L}$ and it is 0 otherwise. Note that $\sum_{\ell \in \mathfrak{L}} R_{\ell m} p_\ell$ is the total price of all the links used by flow $m \in \mathfrak{T}$.

Utility maximization is achieved by maximizing this Lagrangian and the corresponding source rate vector is

$$\mathbf{f}^* = \arg \max_{\mathbf{f} \geq 0} L(\mathbf{f}, \mathbf{p}^*). \tag{7.4}$$

One must find the partial derivatives of the Lagrangian in order to solve the optimization problem. Consider the partial derivative of $L(\mathbf{f}, \mathbf{p})$ w.r.t. $f_m,\ m \in \mathfrak{T}$:

$$\frac{\partial}{\partial f_m} L(\mathbf{f}, \mathbf{p}) = \frac{\partial}{\partial f_m} \left(\sum_{j \in \mathfrak{T}} U_j(\mathbf{f}) \right) - \sum_{\ell \in \mathfrak{L}} R_{\ell m} p_\ell. \tag{7.5}$$

Here,

$$\sum_{\ell \in \mathfrak{L}} R_{\ell m} p_\ell = q_m;$$

$$\frac{\partial}{\partial f_m} \left(\sum_{j \in \mathfrak{T}} U_j(\mathbf{f}) \right) = \frac{\partial}{\partial f_m} \left(\sum_{j \in \mathfrak{T} \setminus m} U_j(\mathbf{f}) \right) + \frac{\partial}{\partial f_m} U_m(\mathbf{f}). \tag{7.6}$$

To proceed, we need to specify $V(f_j)$ within $U_j(\mathbf{f})$. We let $V(f_j) = \alpha \log(f_j)$, which is the utility function of TCP Vegas [LPW02]. Note that we can use the utility functions of any other rate control TCP variants, as long as the chosen function is concave.

Now consider the last two terms on the right-hand side of (7.6):

$$\frac{\partial}{\partial f_m} U_m(\mathbf{f}) = \frac{\alpha}{f_m} - \sum_{i \in \mathfrak{G}} 2K^{(i)} D_m^{(i)}(\mathbf{f}) \left(1 - \gamma_m^{(i)}\right) I_m^{(i)}, \tag{7.7}$$

where $I_m^{(i)} = 1$ if $m \in \mathfrak{G}^{(i)}$ and 0 otherwise. We also have

$$\frac{\partial}{\partial f_m} \left(\sum_{j \in \mathfrak{T} \setminus m} U_j(\mathbf{f}) \right) = \sum_{j \in \mathfrak{T} \setminus m} \sum_{i \in \mathfrak{G}} 2K^{(i)} \gamma_j^{(i)} D_j^{(i)}(\mathbf{f})\, I_j^{(i)} I_m^{(i)}. \tag{7.8}$$

Using (7.5-7.8), we get

$$\frac{\partial}{\partial f_m} L(\mathbf{f}, \mathbf{p}) = \frac{\alpha}{f_m} - q_m - \sum_{i \in \mathfrak{G}} \left\{ 2K^{(i)} \left(D_m^{(i)}(\mathbf{f}) - \sum_{k \in \mathfrak{G}^{(i)}} \gamma_k^{(i)} D_k^{(i)}(\mathbf{f}) \right) I_m^{(i)} \right\}. \quad (7.9)$$

To proceed, we express (7.9) as

$$\frac{\partial}{\partial f_m} L(\mathbf{f}, \mathbf{p}) = \frac{\alpha}{f_m} - q_m - \xi_m(\mathbf{f}), \quad (7.10)$$

where

$$\xi_m(\mathbf{f}) = \sum_{i \in \mathfrak{G}} \xi_m^{(i)}(\mathbf{f})\, I_m^{(i)} \quad , \qquad \xi_m^{(i)}(\mathbf{f}) = 2K^{(i)} \left(D_m^{(i)}(\mathbf{f}) - \sum_{k \in \mathfrak{G}^{(i)}} \gamma_k^{(i)} D_k^{(i)}(\mathbf{f}) \right), \quad (7.11)$$

For uncoordinated flows (i.e., $k \notin \mathfrak{G}^{(i)},\ \forall i$), this reduces to $\alpha/f_m - q_m$ yielding an equilibrium rate that is of the same form as that obtained with TCP Vegas.

7.2 Iterative Rate Update Function

The gradient projection can be used to get the following rate update function.

$$f_m(t+1) = \left[f_m(t) + s \left(\frac{\alpha}{f_m(t)} - q_m(t) - \xi_m(\mathbf{f}(t)) \right) \right]^+. \quad (7.12)$$

Note that to calculate $\xi_m(\mathbf{f})$, the f_i values for all the sources are required. Only the sink node has all the f_i values. Hence, $\xi_m(\mathbf{f})$ is the scalar feedback information sent by the sink node to the source m. Further note that, $\xi_m(\mathbf{x}) = 0$ whenever $K^{(i)} = 0,\ \forall i$, i.e., additional feedback is unnecessary for the ordinary sources. Therefore, a coordinated source updates its rate to $f_m(t+1)$ using its current rate $f_m(t)$, the current queuing delay q_m, and the rate coordination feedback $\xi_m(\mathbf{f}(\mathbf{t}))$ from the sink node as in (7.12). None of the sources need to know the rates of any other source. The sink node needs to keep track of the individual rates of the sources in the coordinated group, and provide per-flow feedback to each source.

Recall the transmission rate of source $m \in \mathfrak{T}$ given by

$$f_m(t) = \frac{w_m(t)}{RTT_m(t)}, \tag{7.13}$$

We can modify the rate control equation (7.12) to an ideal window update function.

$$w_m(t+1) = RTT_m(t+1)\left[\frac{w_m(t)}{RTT_m(t)} + s\left(\frac{\alpha\, RTT_m(t)}{w_m(t)} - q_m(t) - F_m(\mathbf{f}(t))\right)\right]^+. \tag{7.14}$$

For implementation purposes, we use the approximation $RTT_m(t+1) \approx RTT_m(t)$ which yields

$$w_m(t+1) = \left[w_m(t) + s\, RTT_m(t)\left(\frac{\alpha\, RTT_m(t)}{w_m(t)} - q_m(t) - \xi_m(\mathbf{f}(t))\right)\right]^+. \tag{7.15}$$

Here, the current transmission rate $f_m(t)$ is estimated at the source; the queuing delay $q_m(t)$ is also estimated at the source as the difference between the current RTT and the minimum observed RTT. The quantity $\xi_m(\mathbf{f})$ is calculated at the sink node for each coordinated flow and sent to the source as feedback information.

7.3 Simulations

Simulations based on the ns-2 environment were conducted in order to attest the performance of the proposed rate ratio allocation scheme. The simulations were designed in a way that the performance of the proposed algorithm can be compared with the performance of the standard rate control algorithms, particularly TCP Vegas. Two experiments were conducted.

7.3.1 Experiment #1

In the simulation set-up, a common sink node attempts to coordinate 2 CGs to achieve equal rate allocation. It has 7 coordinated flows: 5 are members of CG #1

while 4 are members of CG #2, i.e., 2 flows belong to both the groups. The simulation set-up is shown in Fig. 7.2. Note that, 2 coordinated sources share one path while the

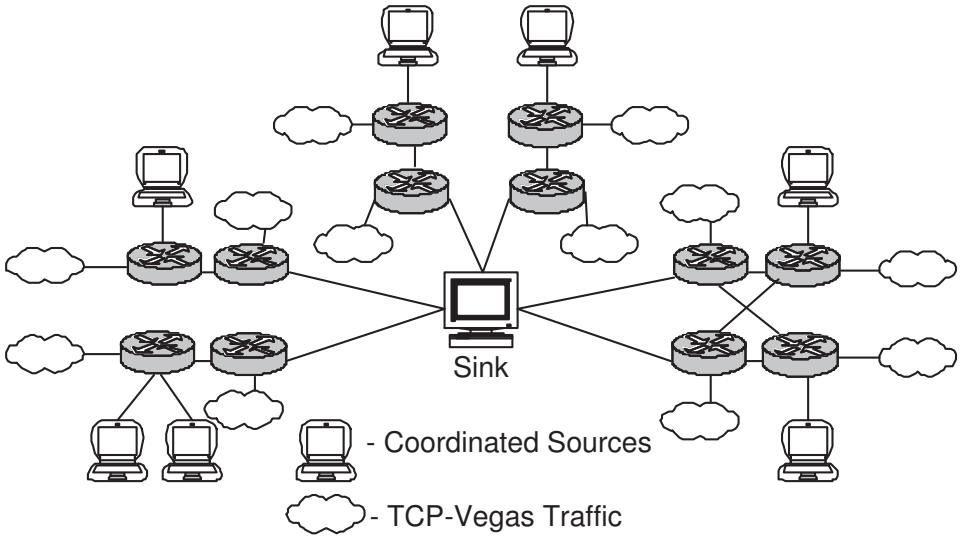

Figure 7.2: Simulation Set-up.

other 5 use five different paths. All six paths also carry TCP Vegas flows. Coordination is required to achieve equal rate allocation among members within each CG since different paths have different capacities. Two different scenarios were considered:

- **Scenario 1**

  All flows run TCP Vegas.

- **Scenario 2**

  Coordinated flows run the new protocol; others employ TCP Vegas. For this scenario, we used the following parameters:

$$\{K_j^{(i)}, s, \alpha, \gamma_j^{(i)}\} = \begin{cases} \{0.01, 1.0, 2.0, 0.2\}, & \text{for CG \#1;} \\ \{0.01, 1.0, 2.0, 0.25\}, & \text{for CG \#2.} \end{cases}$$

Note that the CG #1 has $\gamma_j^{(i)}$ values set to 0.2. This assignment of $\gamma_j^{(i)}$ values indicate the equal rate allocation requirement among members of the CG #1 since it

has only five members. Similarly, $\gamma_j^{(i)}$ values of 0.25 in the CG #2 represents equal rate allocation since it has only four members.

Now it is required to measure the performance of the rate ratio allocation. Hence, the performance of the new algorithm is measured via the *rate ratio non-conformity factor* $NC^{(i)}$ which we define for the i^{th} group as

$$NC^{(i)} = \sqrt{\frac{1}{|\mathfrak{G}^{(i)}|} \sum_{k \in \mathfrak{G}} [D_k^{(i)}]^2}. \tag{7.16}$$

Note that a lower $NC^{(i)}$ value indicates that the actual rate ratio allocations are closer to the desired rate ratios. The $NC^{(1)}$ values for the simulation set-up appear in Fig. 7.3. The $NC^{(2)}$ values for the simulation set-up appear in Fig. 7.4.

Observe that both $NC^{(1)}$ and $NC^{(2)}$ values are lowered with the introduction of the new protocol. That means the allocated rate ratios are closer to the desired rate ratios for both CG #1 and CG #2 with the new protocol. Hence, the performance in each CG has improved under the new protocol. The average improvements are 69% and 74% for CG #1 and CG #2, respectively.

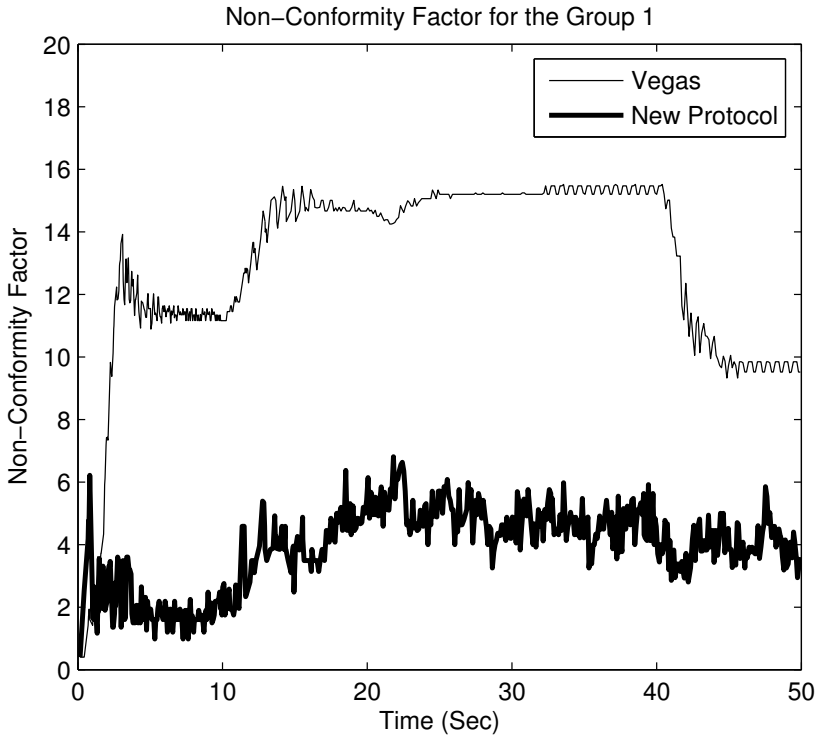

Figure 7.3: Experiment #1: $NC^{(1)}$ values for TCP Vegas and the new protocol.

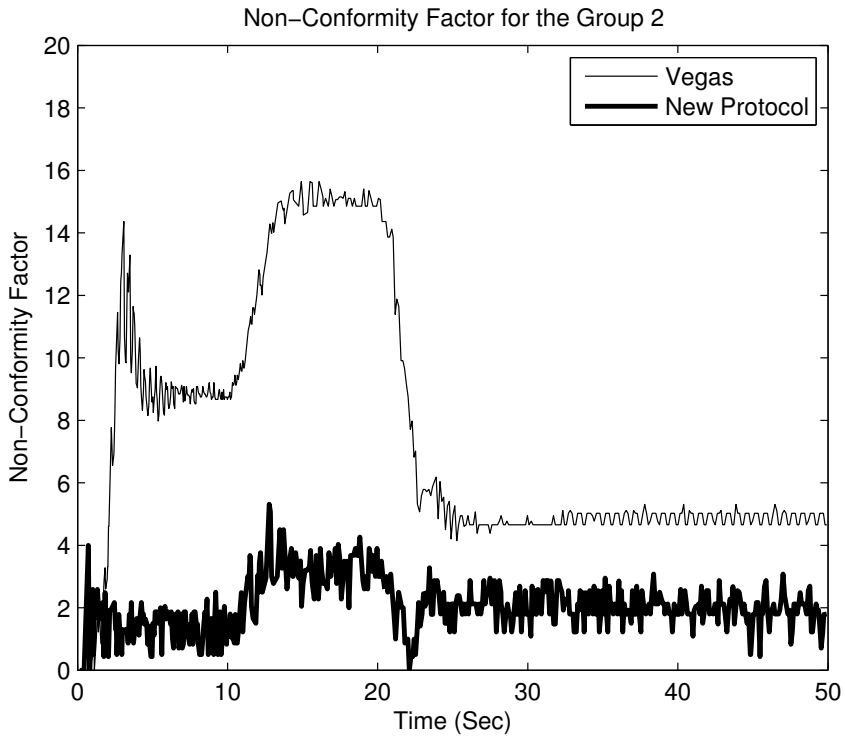

Figure 7.4: Experiment #1: $NC^{(2)}$ values for TCP Vegas and the new protocol.

7.3.2 Experiment #2

Four sensors — 01 color, 01 gray level and 02 ultrasound sensors — are used to gather data of an image. These data are sent to a sink node where the image is to be reconstructed. The color sensor reads the color components as $COLOR = \{R, G, B\}$; the gray level sensor reads the gray scale intensity as $GRAY = \{I\}$; the 2 ultrasound sensors read the corresponding location as $POSITION = \{X, Y\}$. The sensor bundle moves in a zigzag path to get these readings.

The sensors use 4 different paths to send the collected data to the sink. All 4 paths also contain ordinary FTP flows (see Fig. 7.5). Each position reading, color

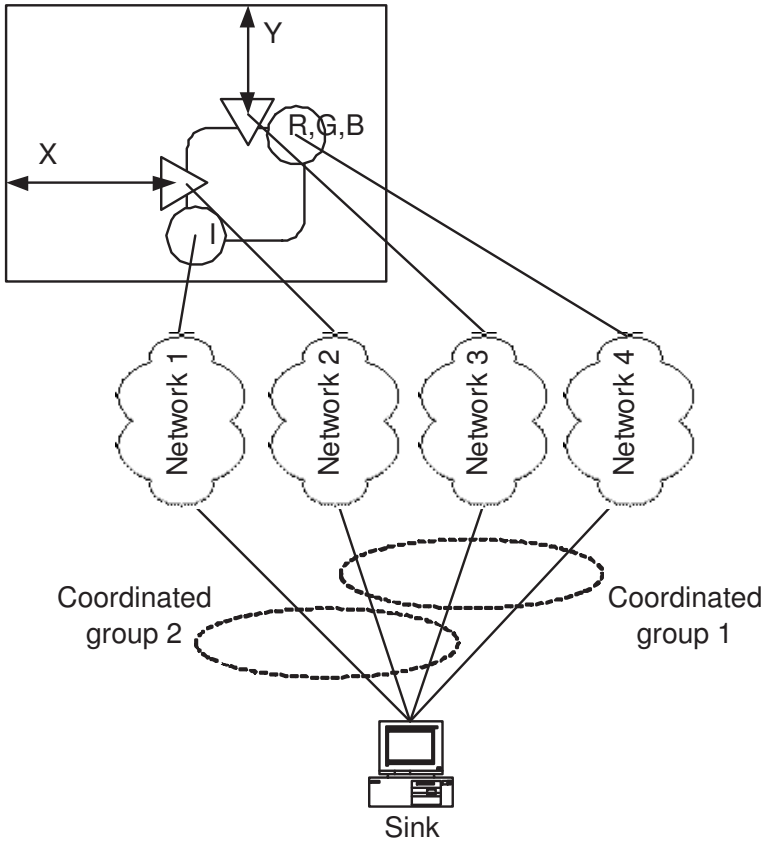

Figure 7.5: Experiment #2: Set-up.

component and gray level is assumed to occupy the same number of bits.

The sink uses two CGs to reconstruct a color and gray level image — CG #1 for

the color image and CG #2 for the gray level image. The characteristics of each CG appear in Table 7.1. Note that, 2 flows are in common to both CGs.

Table 7.1: Experiment #2: Characteristics of the CGs

CG	Member Flows	Rate Ratios	γ Values
#1	$\{X, Y, COLOR\}$	$\{1, 1, 3\}$	$\{1/5, 1/5, 3/5\}$
#2	$\{X, Y, GRAY\}$	$\{1, 1, 1\}$	$\{1/3, 1/3, 1/3\}$

For constructing the color image, one value each from $\{X, Y, R, G, B\}$ are grouped together to reconstruct the location and image color that corresponds to that group of readings; similarly, for constructing the gray scale image, one value each from $\{X, Y, I\}$ are grouped together. The quality of both the reconstructed images depends on the number of such groups created for the image.

The image reconstructed after the sink node receives all the data generated from the sensor bundle when it scans the full image area is termed a *scan.* Fig. 7.6 shows the reconstructed images after $\{1, 2, 3, 4\}$ scans. The $NC^{(1)}$ values for the simulation set-up appear in Fig. 7.7. The $NC^{(2)}$ values for the simulation set-up appear in Fig. 7.8. It is clear from the results that non-conformity factor is reduced for both CGs.

Figure 7.6: Experiment #2: Performance comparison.

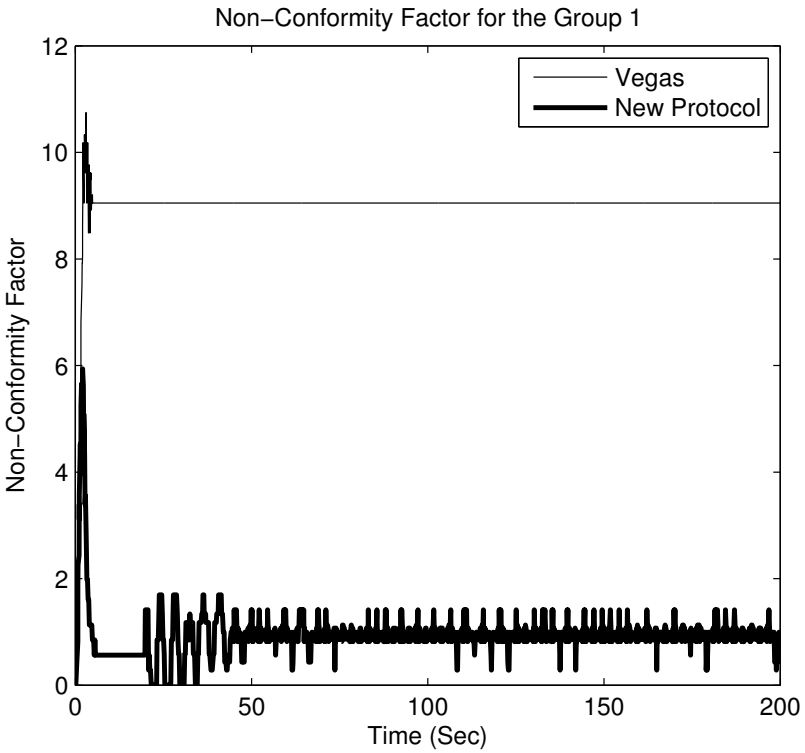

Figure 7.7: Experiment #2: $NC^{(1)}$ values for TCP Vegas and the new protocol.

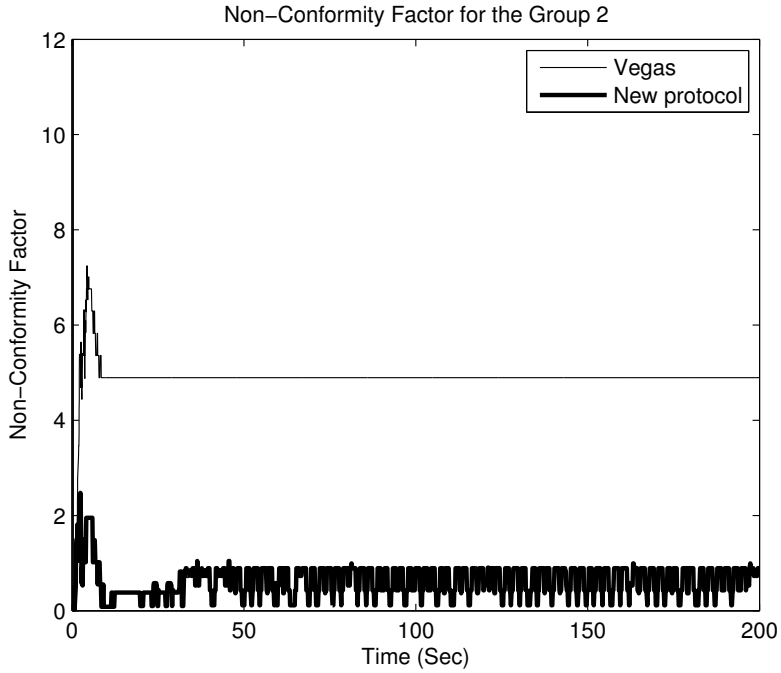

Figure 7.8: Experiment #2: $NC^{(2)}$ values for TCP Vegas and the new protocol.

CHAPTER 8

Conclusion and Future Work

In this work we have proposed a framework for network resource allocation. We have studied how the NUM framework can be adopted to implement rate allocation directives in networks. First we have studied the problem of rate control of multi-sensor target tracking algorithms operating over a network shared by other applications. This work has attempted to bridge the gap between target tracking QoS objectives and network rate allocation. The notion of network utility was defined for the multi-sensor target tracking application, and derived for both the MF and SF methods. The modification of a standard data rate utility function to accommodate the target tracking utility has allowed for the construction of a network resource allocation problem encompassing all data emitting sources in the network. By solving this convex optimization problem using the gradient projection method, we obtained an iterative, distributed solution in the form of a rate control algorithm of the data sources which do not need to communicate with each other. As we have seen, this new rate control algorithm provided better target tracking performance than a more conventional rate control algorithm, while the ordinary data transfer flows are allowed to continue utilizing standard rate/congestion control methods such as TCP Reno.

Next, we have studied how to allocate resources in a way that multiple objectives can be achieved simultaneously. In this part, we have considered a situation where

a sink node is attempting to both track and classify a target. We have shown that the two objectives—target tracking and target classification—have two different rate allocation requirements. We have derived a utility function in a way that the target classification performance can be maximized. Furthermore, we have shown the relationship between the derived utility function and the K-L divergence of the actual and the estimated classification probability distributions. Then, the target classification utility function is incorporated into the previously derived target tracking utility function as an additive term. A modified constrained optimization problem is formulated and the solution is implemented as an iterative rate update function.

We have noted that a shared network can have flows corresponding to various applications with various rate allocation requirements. Hence, it is important to accurately identify flows in order to implement the corresponding rate allocation requirements. This necessitates an online traffic classification algorithm, which is capable of classifying a flow well before the termination of the flow. In this work, a novel online traffic classification algorithm is proposed for this purpose. The main objective of the proposed algorithm is to make a classification decision about the flow well before the flow termination. The use of partial flow information together with a growing window of packets to enable online classification is one of the most important and innovative aspects proposed in this work. Utilization of a set of BNs to calculate class probabilities and utilization of DS theoretic notions to capture uncertainties in partial flows and to update classification probabilities are other novel notions proposed in this work. Two approaches are proposed to make a classification decision. The first approach is capable of classifying flows into singletons, while the second approach enables one to classify flows into composite propositions as well. The second classification approach facilitates better classification performance for minority

classes. The performance of the proposed approach is evaluated by simulations. Two publicly available data sets were used for the simulations. It is clear from the simulation results that the proposed approach is capable of classifying flows accurately by utilizing only a few packets. Moreover, the proposed algorithm appear to provide better classification performance than the available well-established flow classification algorithms.

Once we have accurately identified the flows, we can utilize an intra-class rate allocation algorithm as explained in this work. However, we should consider how to implement the inter-class rate allocation. In other words, how can we allocate the available resources among different applications? We believe one option we have available to address this problem is to allocate the available resources based on a predetermined ratio. We have proposed a utility function in a way that the rate ratio maintenance can be achieved. Then we incorporated the new utility function as an additive term to the standard utility function in order to obtain a new constrained optimization problem. The solution to the optimization problem is implemented as an iterative rate update function. We have shown in our simulations that the rate ratio maintenance can be achieved by implementing the proposed rate update function.

Note that the target tracking algorithms used in this work is based on a set of identical sensors. As mentioned in Section 4.1, MF1 and MF2 methods of multi-sensor target tracking are functionally identical because of the fact that identical sensors are used. It is important to consider a situation when sensors are non-identical. In such a situation, MF1 and MF2 methods of multi-sensor target tracking need to be considered separately in order to derive utility functions. Hence, three utility functions one for SF and two for MF, need to be derived. In the derivation of these utility functions, the measurement model matrix cannot be considered as fixed, instead it

should be considered as a variable. Once the utility functions are derived, an iterative rate update function can be derived by following the same steps as explained in Section 4.2.

The proposed rate allocation algorithm can be extended to other applications as well. Battlefield data fusion is one possible application where the proposed algorithm can be used to allocate resources. One can derive a utility function in a way that data fusion performance is maximized. With the utility function selected to be a concave function of transmission rates, the same approach as used in the proposed work can be utilized to derive an iterative rate update function.

Throughout this work, we assume a fixed network infrastructure. Hence, as it stands, this work is only applicable to wired networks. To extend this work to wireless networks, one can easily start with one of the utility functions proposed in this work and solve the NUM problem for wireless networks in order to obtain a distributed rate control algorithm. This is a possible and important future extension for the proposed rate allocation approach.

All of the utility functions derived in this work are functions of transmission rates. So it is clear that only the transmission rates are considered in the process of obtaining the optimal rate controller. However, factors other than transmission rate may need to be taken into account to measure the service quality of an application. For example, service quality of applications such as multimedia streaming can be heavily dependent on the delay and the latency. So, for such applications one must incorporate delay and latency as service quality measures. Utility functions that incorporate these additional measures must be developed if the proposed rate allocation algorithm is to be extended to accommodate a wider variety of applications.

After deriving an iterative rate update function for the wireless networks, the next step should be to consider non-stationary sensors in the target tracking. In such a situation, the sensor locations available at the sink node are not accurate as we assumed in the proposed work. Then several additional challenges need to be addressed. The first challenge is how to model the error covariances of the sensor measurements as the exact location of the sensors are unknown. The next challenge is how to predict the next location and the measurement error covariance of a sensor in the next time instance as the location of the sensor can be changed.

In Chapter 5, we have derived an iterative rate update function to facilitate target classification together with target tracking. In this approach we considered only the naïve Bayes method of classification. This is not the only classification approach available. In particular, naïve Bayes may not be the best classification algorithm for some scenarios. For example, one can consider a BN based target classification. However, other classification algorithm may have different rate allocation requirements. This nictitates different rate update functions based on the classification algorithm. Hence, it is required to derive utility functions for other classification approaches and follow the same steps as explained in the proposed work to obtain corresponding iterative rate update functions.

Note that all rate allocation schemes proposed in this work are based on a feedback form the sink node. In other words, the proposed rate allocation algorithms are not completely decentralized. However, it is even more useful if one can derive a completely decentralized rate allocation algorithm where the feedback from the sink node is not an input to the rate update function. If such a rate allocation algorithm can be implemented, then the sink node does not need to keep track of all the sources

in order to generate the appropriate feedback. It is clear that the corporation among participating sources is very important to implement a rate allocation algorithm to satisfy rate allocation requirements given by the underlying application. Now the challenge becomes how to implement the corporation among different sources in order to implement the rate allocation requirements. One possible approach to address this issue is by allowing minimal message exchanges between sources. However, it is vital to minimize the message exchanges between sources. Hence, one must utilize an approach to implement the rate allocation requirement while minimizing the communication among sources.

In the flow classification approach, one can utilize different mass assignment schemes. It may be possible to obtain better performances by selecting a different mass assignment scheme. Moreover, note that we have used the DCR to combine masses as the size of the growing window of packets is changed. One can utilize other combination and updating mechanisms for this purpose. Such schemes may yield better performance. Furthermore, it is possible to utilize a feature reduction algorithm in order to select the most appropriate set of features out of the proposed set of features. This may yield better performance for the flow classification algorithm while reducing the complexity.

We have shown in this work that DS theory can be effectively utilize to improve the performance of Internet flow classification. We demonstrated the applicability of DS theory in a BN model based traffic classification. However, one can utilize DS theory in other classification schemes as well. For example, a DS theoretic approach can be used to make the classification decision in k-means clustering, after the minimum distance cluster is determined. Moreover, we can utilize other well-established

classification algorithms to calculate the classification probabilities after the arrival of the n-th packet and use a DS theoretic uncertainty modeling and evidence updating mechanism to obtain the combined inferences.

Rate ratio, as proposed in this work is not the only possible way to implement inter-class resource allocation. In fact, rate ratio may not be the best way to deal with this challenge. In deriving a better approach to deal with inter-class resource allocation, one must consider some other factors as well. For example, there may be unresponsive flows in a way that they do not adopt their transmission rates as the sink node desires. Hence, identification of unresponsive flows is very important to implement a robust inter-class resource allocation algorithm. Moreover, there may be situations where minimum transmission rates need to be satisfied for some classes of flows. Hence, minimum rate enforcement capability should also be incorporated in the inter-class rate allocation algorithm.

APPENDIX A

Proof of Chapter 4 Claim 3

(i) Note that

$$a^{MF}(k)\,c^{MF}(k) - b^{MF}(k)^2 =$$

$$\left[p_1^{MF}(k)\,p_4^{MF}(k) - p_2^{MF}(k)^2\right] + q\,\Delta^2 p_1^{MF}(k) + 2\,q\,\Delta^3 p_2^{MF}(k) + \frac{q}{4}\Delta^4 p_4^{MF}(k).$$

Now it is enough to show that

$$p_1^{MF}(k)\,p_4^{MF}(k) - p_2^{MF}(k)^2 > 0;\ \text{and}\ p_\ell^{MF}(k) \geq 0,\ \ell = 1, 2, 4,\ \forall k. \tag{A.1}$$

The proof is by induction. Note that (A.1) is valid for $k = 0$ since $p_1^{MF}(0) = p_4^{MF}(0) = 1$ and $p_2^{MF}(0) = 0$. Now suppose (A.1) is valid for k. Use (4.20) to show that

$$p_1^{MF}(k+1)\,p_4^{MF}(k+1) - p_2^{MF}(k+1)^2 = \frac{a^{MF}(k)\,c^{MF}(k) - b^{MF}(k)^2}{a^{MF}(k)/r(k+1) + 1} > 0,$$

and $p_\ell^{MF}(k+1) \geq 0$, $\ell = 1, 2, 4$. Hence, by induction, (A.1) is true for all $k > 0$.

(ii) Simple, yet somewhat lengthy, manipulations yield

$$\rho_1(k)\,\rho_4(k) - \rho_2(k)^2 = \frac{2}{\sigma_{N,N}} + 2\sum_{i=1}^{N-1}\left[\frac{1}{\sigma_{i,i}} + 2\sum_{j=i+1}^{N}\frac{\sigma_{i,j}}{\sigma_{i,i}\sigma_{j,j}}\right],$$

where

$$\sigma_{i,j} = a_i^{SF}(k)\,c_j^{SF}(k) + a_j^{SF}(k)\,c_i^{SF}(k) - 2\,b_i^{SF}(k)\,b_j^{SF}(k).$$

Now it is enough to prove that, for all i, j, k,

$$\sigma_{i,j} > 0;\ p_{1i}^{SF}(k) \geq 0,\ p_{2i}^{SF}(k) \geq 0,\ p_{4i}^{SF}(k) \geq 0. \tag{A.2}$$

Consider $\sigma_{i,j}$:

$$\begin{aligned} a_i^{SF}(k)\, c_j^{SF}(k) + a_j^{SF}(k)\, c_i^{SF}(k) - 2\, b_i^{SF}(k)\, b_j^{SF}(k) \\ = \left(p_{1i}^{SF}(k)\, p_{4j}^{SF}(k) + p_{1j}^{SF}(k)\, p_{4i}^{SF}(k) - 2\, p_{2i}^{SF}(k)\, p_{2j}^{SF}(k)\right) \\ + q\,\Delta^2\left(p_{1i}^{SF}(k) + p_{1j}^{SF}(k)\right) + q\,\Delta^3\left(p_{2i}^{SF}(k) + p_{2j}^{SF}(k)\right) \\ + \frac{q}{4}\Delta^4\left(p_{4i}^{SF}(k) + p_{4j}^{SF}(k)\right). \end{aligned}$$

So, it is enough to prove that

$$\begin{aligned} p_{1i}^{SF}(k)\, p_{4j}^{SF}(k) + p_{1j}^{SF}(k)\, p_{4i}^{SF}(k) - 2\, p_{2i}^{SF}(k)\, p_{2j}^{SF}(k) > 0;\ \text{and} \\ p_{1i}^{SF}(k) \geq 0,\ p_{2i}^{SF}(k) \geq 0,\ p_{4i}^{SF}(k) \geq 0,\ \forall i, j, k. \end{aligned} \tag{A.3}$$

The proof is by induction. Note that (A.3) is valid for $k = 0$ since $p_{1i}^{SF}(0) = p_{1j}^{SF}(0) = p_{4i}^{SF}(0) = p_{4j}^{SF}(0) = 1$ and $p_{2i}^{SF}(0) = p_{2j}^{SF}(0) = 0,\ \forall i, j$. Suppose (A.3) is valid for k. Use (4.23) to show that

$$\begin{aligned} p_{1i}^{SF}(k+1)\, p_{4j}^{SF}(k+1) + p_{1j}^{SF}(k+1)\, p_{4i}^{SF}(k+1) \\ - 2\, p_{2i}^{SF}(k+1)\, p_{2j}^{SF}(k+1) \\ = \left[a_i^{SF}(k)\frac{\sigma_{j,j}}{2\, r_j(k+1)} + a_j^{SF}(k)\frac{\sigma_{i,i}}{2\, r_i(k+1)} + \sigma_{i,j}\right] \\ \div \left[\left(1 + \frac{a_i^{SF}(k)}{r_i(k+1)}\right)\left(1 + \frac{a_j^{SF}(k)}{r_j(k+1)}\right)\right] > 0, \end{aligned}$$

and

$$p_{1i}^{SF}(k+1) \geq 0,\ p_{2i}^{SF}(k+1) \geq 0,\ p_{4i}^{SF}(k+1) \geq 0,$$

for all i. Hence, by induction, (A.2) for all $k > 0$. ■

APPENDIX B

Proof of Chapter 4 Lemma 1

First, we note that the function

$$h(y) \equiv \frac{-2}{y + A(k)}$$

is a concave non-decreasing scalar function of $y > 0$ because

$$\frac{d^2 h(y)}{d^2 y} = \frac{-4}{(y + A(k))^3} \text{ and } A(k) > 0.$$

In addition, $1/r(k+1)$ is concave w.r.t. $\mathbf{f}$ because

$$\frac{\partial^2}{\partial f_j \partial f_i} \left(\frac{1}{r(k+1)} \right) = \begin{cases} \dfrac{-2\, d\, \gamma_i(k+1)}{(r_i(k+1)\, f_i)^3}, & \text{for } j = i; \\ 0, & \text{for } j \neq i, \end{cases}$$

and $-2\, d\, \gamma_i(k+1)/(r_i(k+1)\, f_i)^3 \leq 0$.

Therefore, $S(k+1) = h(y)|_{y=1/r(k+1)}$ is also a concave function of $\mathbf{f}$ as claimed. ■

APPENDIX C

Derivation of Chapter 5 Equation 5.8

To proceed let us note 5.2 and define the following.

$$\mathfrak{Q} = \sum_{j=1}^{\Theta} Q(j) \text{ and } \mathfrak{Q}' = \sum_{j=1}^{\Theta} Q'(j). \tag{C.1}$$

Now one can rewrite 5.7 as

$$|\Upsilon'(i)| \leq \frac{1}{\mathfrak{Q}\mathfrak{Q}'} \left[\sum_{l=1}^{N} Q(i) \sum_{j=1}^{\Theta} \frac{Q(j)}{Pr(\chi_l|C_j)} |\epsilon_{jl}| + \sum_{j=1}^{\Theta} Q(j) \sum_{l=1}^{N} \frac{Q(i)}{Pr(\chi_l|C_i)} |\epsilon_{il}| \right]. \tag{C.2}$$

Now let us consider the following summation.

$$\begin{aligned}
\sum_{i=1}^{\Theta} |\Upsilon'(i)| \leq & \frac{1}{\mathfrak{Q}\mathfrak{Q}'} \sum_{i=1}^{\Theta} Q(i) \left[\sum_{l=1}^{N} \sum_{j=1}^{\Theta} \frac{Q(j)}{Pr(\chi_l|C_j)} |\epsilon_{jl}| \right] \\
& + \frac{1}{\mathfrak{Q}\mathfrak{Q}'} \sum_{j=1}^{\Theta} Q(j) \left[\sum_{l=1}^{N} \sum_{i=1}^{\Theta} \frac{Q(i)}{Pr(\chi_l|C_i)} |\epsilon_{il}| \right] \qquad \text{(C.3)} \\
\leq & \frac{1}{\mathfrak{Q}\mathfrak{Q}'} \mathfrak{Q} \left[2 \sum_{l=1}^{N} \sum_{j=1}^{\Theta} \frac{Q(j)}{Pr(\chi_l|C_j)} |\epsilon_{jl}| \right] \\
\leq & \frac{1}{\mathfrak{Q}'} \left[2 \sum_{l=1}^{N} \sum_{j=1}^{\Theta} \frac{Q(j)}{Pr(\chi_l|C_j)} |\epsilon_{jl}| \right] \\
\leq & \frac{\sum_{l=1}^{N} \sum_{j=1}^{\Theta} B_{jl} |\epsilon_{jl}|}{\sum_{j=1}^{\Theta} Q'(j)}. \qquad \text{(C.4)}
\end{aligned}$$

Bibliography

[AG92] M. A. Abidi and R. C. Gonzalez, *Data Fusion in Robotics and Machine Intelligence*, Academic Press, San Diego, CA, 1992.

[AMG07] T. Auld, A. W. Moore, and S. F. Gull, *Bayesian neural networks for Internet traffic classification*, IEEE Transactions on Neural Networks **18** (2007), no. 1, 223–239.

[BCKC02] Y.-S. Choi B.-C. Kim and Y.-Z. Cho, *An enhanced marking strategy for explicit congestion notification in the internet*, IEEE International Conference on Communications 2002 (ICC2002) (New York City, USA), April 2002, pp. 2330–2334.

[Ber99] D. P. Bertsekas, *Nonlinear Programming*, second ed., Athena Scientific, Belmont, MA, 1999.

[BP95] L. S. Brakmo and L. L. Peterson, *TCP Vegas: End to end congestion avoidance on a global Internet*, IEEE Journal on Selected Areas in Communications **13** (1995), no. 8, 1465–1480.

[BP99] S. Blackman and R. Popoli, *Design and Analyis of Modern Tracking Systems*, Artech House, Norwood, MA, 1999.

[Bro00] P. Brown, *Resource sharing of TCP connections with different round trip times*, IEEE INFOCOM 2000 (Tel-Aviv, Israel), March 2000, pp. 1734–1741.

[BTA+06] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, *Traffic classification on the fly*, ACM SIGCOMM Computer Communication Review **36** (2006), no. 2, 23–26.

[BTS06] L. Bernaille, R. Teixeira, and K. Salamatian, *Early application identification*, Proc. International Conference On Emerging Networking Experiments And Technologies 2006 (Lisboa, Portugal), December 2006, pp. 2521–2524.

[Bun96] W. Buntine, *Guide to the literature on learning probabilistic networks from data*, IEEE Transactions on Knowledge and Data Engineering **8** (1996), no. 2, 195–210.

[CCC03] Y.-C. Chan, C.-T. Chan, and Y.-C. Chen, *An enhanced congestion avoidance mechanism for TCP Vegas*, IEEE Communications Letters **7** (2003), no. 7, 343–345.

[CFM+09] M. Chen, X. Fan, M. N. Murthi, T.D. Wickramarathna, and K. Premaratne, *Normalized queuing delay: Congestion control jointly utilizing delay and marking*, IEEE/ACM Transactions on Networking (2009), To appear.

[CV05] T. M. Chen and V. Venkataramanan, *Dempster-Shafer theory for intrusion detection in ad hoc networks*, IEEE Internet Computing **9** (2005), no. 6, 35–41.

[CWL04] J. Cheng, D. X. Wei, and S. H. Low, *FAST TCP: Motivation, architecture, algorithms, performance*, Proc. IEEE INFOCOM'04 (Hong Kong), vol. 4, March 2004, pp. 2490–2501.

[CYMBKC00] C. Chee-Yee, S. Mori, W. H. Barker, and C. Kuo-Chu, *Architectures and algorithms for track association and fusion*, IEEE Aerospace and Electronic Systems Magazine **5** (2000), no. 1, 5–13.

[DBP06a] D. A. Dewasurendra, P. H. Bauer, and K. Premaratne, *Distributed evidence filtering in networked embedded systems*, Networked Embedded Sensing and Control (P. J. Antsaklis and P. Tabuada, eds.), Lecture Notes in Control and Information Sciences, vol. 331, Springer, Berlin/Heidelberg, Germany, 2006, pp. 183–198.

[DBP06b] ———, *Distributed evidence filtering: The recursive case*, Proc. IEEE International Symposium on Circuits and Systems (ISCAS'06) (Kos, Greece), May 2006, pp. 4731–4734.

[DBP07] ———, *Evidence filtering*, IEEE Transactions on Signal Processing **55** (2007), no. 12, 5796–5805.

[DL04] F.-B. Duh and C.-T. Lin, *Tracking a maneuvering target using neural fuzzy network*, IEEE Transaction on Systems, Man and Cybernetics, Part B: Cybernetics **34** (2004), no. 1, 16–33.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society, Series B (Methodological) **39** (1977), no. 1, 1–38.

[DS04] F. Delmotte and P. Smets, *Target identification based on the transferable belief model interpretation of Dempster-Shafer model*, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans **34** (2004), no. 4, 457–471.

[EMA06] J. Erman, A. Mahanti, and M. Arlitt, *Internet traffic identification using machine learning*, IEEE Global Telecommunications Conference (GLOBECOM'06) (San Francisco, CA), November 2006, pp. 1–6.

[EMA+07] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, *Offline/realtime traffic classification using semi-supervised learning*, Performance Evaluation **64** (2007), no. 9-12, 1194–1213.

[FB00] V. Firoiu and M. Borden, *A study of active queue management for congestion control*, IEEE INFOCOM 2000 (Tel Aviv, Israel), March 2000, pp. 1435–1444.

[FF99] L. Frenkel and M. Feder, *Recursive expectation-maximization (EM) algorithms for time-varying parameters with applications to multiple target tracking*, IEEE Transactions on Signal Processing **47** (1999), no. 2, 306–320.

[FJ93] S. Floyd and V. Jacobson, *Random early detection gateways for congestion avoidance*, IEEE/ACM Transactions on Networking **1** (1993), no. 4, 397–413.

[Flo03] S. Floyd, *Highspeed TCP for large congestion windows*, Tech. Rep. RFC 3649 Experimental, IETF,, December 2003.

[GGB+02] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, *Particle filters for positioning, navigation, and tracking*, IEEE Transactions on Signal Processing **5** (2002), no. 1, 425–437.

[GH01] Q. Gan and C. J. Harris, *Comparison of two measurement fusion methods for Kalman-filter-based multisensor data fusion*, IEEE Transactions on Aerospace and Electronic Systems **37** (2001), no. 1, 273–279.

[GK01] R. Gibbens and P. Key, *Distributed control and resource marking using best-effort routers*, IEEE Network **15** (2001), no. 3, 54–59.

[HD06] M. Huang and S. Dey, *Dynamic quantizer design for hidden Markov state estimation via multiple sensors with fusion center feedback*, IEEE Transactions on Signal Processing **54** (2006), no. 8, 2887–2896.

[HL01] D. L. Hall and J. Llinas (eds.), *Handbook of Multisensor Data Fusion*, CRC Press, Boca Raton, FL, 2001.

[HL08] D. He and H. Leung, *Network intrusion detection using CFAR abrupt-change detectors*, IEEE Transactions on Instrumentation and Measurement **57** (2008), no. 3, 490–497.

[HLG06] W. Hu, J. Li, and Q. Gao, *Intrusion detection engine based on Dempster-Shafer's theory of evidence*, Proc. 2006 International Conference on Communications, Circuits and Systems (Guilin, China), 2006.

[HPS07] K. K. R. G. K. Hewawasam, K. Premaratne, and M.-L. Shyu, *Rule mining and classification in a situation assessment application: A belief theoretic approach for handling data imperfections*, IEEE Transaction on Systems, Man and Cybernetics, Part B: Cybernetics **37** (2007), no. 6, 1446–1459.

[JE07] A. Josang and Z. Elouedi, *Intperpreting belief functions as dirichlet distributions*, Proc. 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (Hammamet, Tunisia), October 2007, pp. 393–404.

[KBKL03] A. Tang K. B. Kim and S. H. Low, *A stabilizing AQM based on virtual queue dynamics in supporting TCP with arbitrary delays*, 42nd IEEE Conference on Decision and Control 2003 Proceedings (Hawaii, USA), December 2003, pp. 3665–3670.

[KBS03] T. Kirubarajan and Y. Bar-Shalom, *Kalman filter versus IMM estimator: When do we need the latter?*, IEEE Transactions on Aerospace and Electronic Systems **39** (2003), no. 4, 1452–1457.

[Kel03] T. Kelly, *Scalable TCP: Improving performance in highspeed wide area networks*, Computer Communication Review **32** (2003), no. 2.

[KMT98] F. P. Kelly, A. Maulloo, and D. Tan, *Rate control in communication networks: shadow prices, proportional fairness and stability*, Journal of the Operational Research Society **49** (1998), 237–252.

[KPF05] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, *BLINC: multilevel traffic classification in the dark*, Proc. Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'05) (Philadelphia, PA), August 2005, pp. 229–240.

[KR04] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*, Addison-Wesley, Menlo Park, CA, 2004.

[LL99] S. H. Low and D. E. Lapsley, *Optimization flow control—I: basic algorithm and convergence*, IEEE/ACM Transactions on Networking **7** (1999), no. 6.

[LM01] I. K. Leung and J. K. Muppala, *Packet marking strategies for explicit congestion notification (ECN)*, IEEE International Conference on Performance, Computing, and Communications 2001 (Phoenix, Arizona, USA), April 2001, pp. 17–23.

[LM06] W. Li and A. W Moore, *Learning for accurate classification of real-time traffic*, Proc. Conference on Future Networking Technologies (CoNEXT'06) (Lisboa, Portugal), December 2006.

[Log] C. Logg, *Characterization of the traffic between SLAC and the Internet*, http://www.slac.stanford.edu/comp/net/slac-netflow/html/SLAC-netflow.html.

[Low03] S. H. Low, *A duality model of TCP and queue management algorithms*, IEEE/ACM Transactions on Networking **11** (2003), no. 4, 525–536.

[LPW02] S. H. Low, L. L. Peterson, and L. Wang, *Understanding TCP Vegas: a duality model*, Journal of the ACM **49** (2002), no. 2, 207–235.

[MHLB04] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, *Flow clustering using machine learning techniques*, Proc. International Workshop on Passive and Active Measurement (PAM'04) (C. Barakat and I. Pratt, eds.), Lecture Notes in Computer Science, vol. 3015, Springer-Verlag, Berlin, Germany, 2004, pp. 205–214.

[MKK+01] D. Moore, K. Keys, R. Koga, E. Lagache, and K. C. Claffy, *The CoralReef software suite as a tool for system and network administrators*, Proc. USENIX Conference on System Administration (San Diego, CA), December 2001, pp. 133–144.

[MW00] J. Mo and J. Walrand, *Fair end-to-end window-based congestion control*, IEEE/ACM Transactions on Networking **8** (2000), no. 5, 556–567.

[MZ05] A. W. Moore and D. Zuev, *Internet traffic classification using Bayesian analysis techniques*, Proc. ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (Baniff, Alberta, Canada), June 2005, pp. 50–60.

[NA06] T. T. T. Nguyen and G. Armitage, *Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world IP networks*, Proc. IEEE Conference on Local Computer Networks (LCN'06) (Tampa, FL), November 2006, pp. 369–376.

[NLA] *NLANR PMA: Special Traces Archive*, National Laboratory for Applied Network Research, Measurement and Network Analysis Group (NLANR/MNAG), San Diego Supercomputer Center (SDSC), University of California (San Diego, CA), http://pma.nlanr.net/Special/.

[ns2] *The Network Simulator—ns-2*, Information Sciences Institute, Viterbi School of Engineering, University of Southern California (Los Angeles, CA), http://www.isi.edu/nsnam/ns.

[ONV06] O. Ozdemir, R. Niu, and P. K. Varshney, *Channel aware particle filtering for tracking in sensor networks*, Proc. of the Thirty-Ninth Annual Asilomar Conference on Signals, Systems, and Computers (Pacific Grove, CA), October 2006, pp. 290–294.

[OS07] R. Olfati-Saber, *Distributed Kalman filtering for sensor networks*, Proc. IEEE Conference on Decision and Control (CDC'07) (New Orleans, LA), December 2007, pp. 5492–5498.

[OSST04] A. Oveissian, K. Salamatian, A. Soule, and N. Taft, *Fast flow classification over Internet*, Proc. Conference on Communication Networks and Services Research (CNSR'04) (Fredericton, N.B., Canada), May 2004, pp. 235–242.

[Pea88] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Francisco, CA, 1988.

[PTK06] J. Park, H.-R. Tyan, and C.-C. J. Kuo, *GA-based Internet traffic classification technique for QoS provisioning*, Proc. IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'06) (Pasadena, CA), December 2006, pp. 251–254.

[RFC97] RFC, *RFC 2001: TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms*, Tech. report, January 1997.

[RFC99] ———, *RFC 2581: TCP congestion control*, Tech. report, April 1999.

[RGR06] A. Ribeiro, G. B. Giannakis, and S. I. Roumeliotis, *SOI-KF: Distributed Kalman filtering with low-cost communications using the sign of innovations*, IEEE Transaction of Signal Processing **54** (2006), no. 12, 4782–4795.

[RMP06] M. C. Ranasingha, M. N. Murthi, and K. Premaratne, *A congestion control method to support coordinated bandwidth allocation*, Proc. Conference on Information Sciences and Systems (CISS'06) (Princeton, NJ), March 2006, pp. 597–602.

[RMPF08] M. C. Ranasingha, M. N. Murthi, K. Premaratne, and X. Fan, *Transmission rate allocation in multi-sensor target tracking*, Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'08) (Las Vegas, NV), March 2008.

[RMPF09] ———, *Transmission rate allocation in multisensor target tracking over a shared network*, IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics **39** (2009), no. 2, 348–362.

[RSSD04] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, *Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification*, Proc. ACM SIGCOMM Conference on Internet Measurement (Taormina, Sicily, Italy), October 2004, pp. 135–148.

[Sah96] R. K. Saha, *Track-to-track fusion with dissimilar sensors*, IEEE Transactions on Aerospace and Electronic Systems **32** (1996), no. 3, 1021–1029.

[SC06] V. Saligrama and D. Castanon, *Reliable tracking with intermittent communications*, Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'06) (Toulouse, France), vol. 5, May 2006, pp. V–V.

[Sha76] G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, NJ, 1976.

[SM08] H. Shah and D. Morrell, *Non-myopic sensor scheduling for a distributed sensor network*, Proc. IEEE International Conference on Acoustics, Speech and Signal Processing 2008 (ICASSP 2008) (Las Vegas, NV), March 2008, pp. 2541–2544.

[Sme99] P. Smets, *Practical uses of belief functions*, Proc. Conference on Uncertainty in Artificial Intelligence (UAI'99) (K. B. Laskey and H. Prade, eds.), Morgan Kaufmann, San Francisco, CA, 1999, pp. 612–621.

[SSF+04] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, *Kalman filtering with intermittent observations*, IEEE Transactions on Automatic Control **49** (2004), no. 9, 1453–1464.

[Wat98] G. A. Watson, *Multisensor ESA resource management*, Proc. IEEE Aerospace Conference, vol. 5, March 1998, pp. 13–27.

[WF99] I. H. Witten and E. Frank, *Data mining: Practical machine learning tools and techniques with java implementations*, Morgan Kaufmann, San Francisco, CA, 1999.

[WMM04] C. Wright, F. Monrose, and G. Masson, *HMM profiles for network traffic classification*, Proc. ACM Workshop on Visualization and Data Mining for Computer Security (Washington, DC), October 2004, pp. 9–15.

[WZA06] N. Williams, S. Zander, and G. Armitage, *A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification*, ACM SIGCOMM Computer Communication Review **36** (2006), no. 5, 5–16.

[XHR04] L. Xu, K. Harfoush, and I. Rhee, *Binary increase congestion control for fast, long distance networks*, Proc. IEEE INFOCOM'04) (Hong Kong), 2004.

[ZMVM06] L. Zuo, K. Mehrotra, P. K. Varshney, and C. K. Mohan, *Bandwidth-efficient target tracking in distributed sensor networks using particle filters*, Proc. International Conference on Information Fusion (ICIF'06) (Florence, Italy), July 2006, pp. 1–4.

[ZNA05] S. Zander, T. Nguyen, and G. Armitage, *Automated traffic classification and application identification using machine learning*, Proc. IEEE Conference on Local Computer Networks (LCN'05) (Sydney, Australia), November 2005, pp. 250–257.

[ZNV08] L. Zuo, R. Niu, and P. K. Varshney, *A sensor selection approach for target tracking in sensor networks with quantized measurements*, Proc. IEEE International Conference on Acoustics, Speech and Signal Processing 2008 (ICASSP 2008) (Las Vegas, NV), March 2008, pp. 2521–2524.

[ZPB02] J. Zhang, K. Premaratne, and P. H. Bauer, *Local resource management of distributed sensor networks via static output feedback control*, Proc. IEEE International Symposium on Circuits and Systems (ISCAS'02) (Scottsdale, AZ), vol. III, May 2002, pp. 25–28.

[ZSP+04] J. Zhang, S. P. Subasingha, K. Premaratne, M.-L. Shyu, M. Kubat, and K. K. R. G. K. Hewawasam, *A novel belief theoretic association rule mining based classifier for handling class label ambiguities*, Foundations in Data Mining (FDM) Workshop, IEEE International Conference on Data Mining (ICDM'04) (Brighton, UK), November 2004.