1-1-2014

Taming Uncertainties In Real-Time Routing For Wireless Networked Sensing And Control

Xiaohui Liu
Wayne State University,

TAMING UNCERTAINTIES IN REAL-TIME ROUTING FOR WIRELESS NETWORKED SENSING AND CONTROL

by

Xiaohui Liu

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2014

MAJOR: COMPUTER SCIENCE

Approved by:

_____

Advisor Date

DEDICATION

To my family and friends

ACKNOWLEDGMENTS

I owe a great deal of gratitude to my advisor Dr. Hongwei Zhang, for his scrupulous guidance and incessant encouragement and support. I also thank my fellow graduate students to collaborate on this work. Other thanks are due to Dr. Nathan Fisher and Dr. Loren Schwiebert for serving on my committee and providing invaluable feedbacks.

CONTENTS

LIST OF FIGURES

1 Introduction

Besides deployments for open-loop sensing such as environmental monitoring, embedded wireless networks are increasingly being explored for real-time, closed-loop sensing and control. For instance, the wireless networking standard IEEE 802.15.4g has been defined for large scale process control applications such as smart grid sensing and control [3], and wireless networks are expected to serve as major communication infrastructures in neighborhood area networks and home area networks of the smart grid [23, 19, 53, 49]. In addition, wireless networking standards such as the IEEE 802.15.4e, WirelessHART, and ISA SP100.11a have been defined for industrial monitoring and control [4, 11, 45], wireless sensor networks have been deployed for industrial automation [35, 50], and the automotive industry has also been exploring the application of wireless networks to vehicle sensing and control [20, 46]. In wireless networked sensing and control, message passing (or messaging for short) across wireless networks is a basic enabler for coordination among distributed sensors, controllers, and actuators. In supporting mission-critical tasks such as smart grid control and industrial process control, wireless messaging is required to be reliable (i.e., having high delivery ratio) and in real-time [42]. This is because packet loss and large delay usually reduce the system stability (e.g., in proportional-integral control), lengthen the settling time, and increase the maximum overshoot in control [27].

In multi-hop wireless networks, a basis for reliable, real-time messaging is real-time routing which routes data packets from their sources to destinations within specified deadlines. Nonetheless, link/path delays (i.e., the time taken to successfully deliver a packet across a link or a path) are dynamic, uncertain in wireless sensing and control (WSC) networks due to factors such as the spatiotemporal wireless link dynamics and the queueing dynamics along links/paths. The dynamics and uncertainties in link/path delays introduce fundamental *challenges* to real-time routing:

- Firstly, the dynamics and uncertainties make link/path delays probabilistic in nature. Given the delay distributions of individual links along a path, the basic problem of computing the probabilistically guaranteed path delays is NP-hard [25]; in real-time routing where nodes

need to identify paths that ensure certain delay bounds, however, quantifying probabilistic path delays is a basic task, and it may well have to be executed by resource-constrained nodes in a distributed manner.

- Secondly, given that link/path delay is a highly varying metric and that it can change at a short timescale of each packet transmission, it is important to adapt to in-situ delay conditions in routing. Yet the highly-varying nature of link/path delays makes it difficult to accurately estimate path delays in a distributed and agile manner, and it has been observed that delay-adaptive routing can lead to routing instability and low data delivery performance in general [16, 57].

Despite much work in throughput- or energy-efficiency-oriented wireless routing [7, 15, 24, 55], real-time routing is much less studied. Moreover, the existing work that do consider data delivery delay in wireless routing either only try to minimize average path delay without ensuring probabilistic delay bounds [51, 22, 26, 36, 52], or they do not address the challenges that delay uncertainties pose to the task of quantifying probabilistic path delays and the task of addressing instability of delay-adaptive routing [29, 43]. Therefore, how to enable real-time routing in the presence of dynamic, uncertain link/path delays remains an important open problem for real-time wireless networked sensing and control.

Towards enabling routing with probabilistic delay bounds in WSC networks, we propose the *Multi-Timescale Adaptation (MTA)* routing protocol that addresses the aforementioned challenges of dynamic, uncertain link/path delays in real-time routing. In MTA, nodes leverage the different timescales of dynamics to accurately estimate probabilistic path delay bounds in an agile manner and to adapt spatiotemporal data flow control at the same timescales of the dynamics themselves. More specifically, we make the following contributions:

- For accurate, agile estimation of probabilistic path delay bounds, we decompose contributors to path delay uncertainties into two factors: dynamic per-packet transmission time (which we refer to as *packet-time* hereafter) and dynamic queueing along paths. Through detailed

experimental analysis, we find that, given a network condition, the distribution of packet-time is quite stable despite the quick variation of instantaneous packet-time. This enables each node to accurately estimate the mean and variance of packet-time from itself to the next-hop along a path. We also observe that the packet-time for different packet transmissions, whether from the same node or from different nodes, are uncorrelated; this enables each node to compute, for a given time instant, the variance of the path delay from itself to the destination node as the sum of the variances of the packet-times for all the packets queued along the path at that instant. Based on these observations, we develop a multi-timescale approach, denoted by *multi-timescale estimation (MTE)*, to accurately estimate the highly-varying mean and variance of path delay by accurately estimating the mean and variance of packet-time and by adapting to fast-varying queueing in an accurate, agile manner. Using the mean and variance of path delay, we evaluate different methods of upper-bounding quantiles, and we identify Chebyshev Inequality as an effective basis for computing probabilistic delay bounds in constant time.

- For enabling adaptivity while addressing instability and low-performance in real-time routing, we propose the *Multi-Timescale Adaptation (MTA)* routing protocol: to facilitate the aforementioned multi-timescale estimation (MTE) and to avoid detrimental instability while ensuring data delivery performance during adaptation, a directed-acyclic-graph (DAG) is maintained at lower frequencies based on the relatively slow-varying link property ETX (i.e., expected number of transmissions taken to successfully deliver a packet), which reflects network throughput, data delivery reliability, and the overall trend of data delivery delay [15, 17]; at higher frequencies and based on the MTE method, the data flow within the DAG is controlled on a per-packet basis to minimize ETX and to ensure packet delivery within the required probabilistic delay bound. By ensuring overall stability and performance while addressing short-term dynamics at the same time, MTA enables efficient, real-time routing in the presence of complex dynamics and uncertainties.

- We implement MTE and MTA in TinyOS, and we address the challenges of limited memory, limited CPU capability, and the lack of real-time operation support in TinyOS. Besides the running MTA protocol, these implementation strategies may well be of interest to real-time routing in general.

- We evaluate the performance of MTA and and other related work in the high-fidelity sensor network testbeds NetEye [32] and Indriya [2]. We find that MTA significantly outperforms existing protocols, e.g., improving deadline success ratio by 89% and reducing transmission cost by a factor of 9.7.

The rest of the paper is organized as follows. We briefly introduce the NetEye and the Indriya testbeds in Chapter 2. We present the MTE method and the MTA protocol in Chapter 3 and 4 respectively, and then we present the measurement study in Chapter 5. Then we evaluate our design with sparse networks in Chapter 6. We discuss related work in Chapter 7, and we make concluding remarks in Chapter 8.

2 Preliminaries

Our study leverages two publicly available wireless sensor network testbeds NetEye [32] and Indriya [2]. In what follows, we briefly introduce the two testbeds and the traffic patterns we study.

NetEye testbed. NetEye [32] is deployed in a large lab space at Wayne State University as shown in Figure 1. We use a subset of a 15×7 grid of TelosB motes in NetEye, where every two

Figure 1: *NetEye* wireless sensor network testbed

closest neighboring motes are separated by 2 feet. The subset of the grid forms a random network, and it is generated by removing each mote of the 15×7 grid with probability 0.2.

Each of these TelosB motes is equipped with a 3dB signal attenuator and a 2.45GHz monopole antenna. In our measurement study, we set the radio transmission power to be -25dBm (i.e., power level 3 in TinyOS) such that multihop networks can be created and the link reliability is over 90% for links up to 6 feet long. For the transmission power of -25dBm, Figure 2 shows the boxplot of packet delivery ratio (PDR) for links of different length, and Figure 3 shows the histogram of background noise power in NetEye. We see that there is a high degree of variability in PDR for links of equal length and in background noise power. Thus the testbed reflects non-uniform network settings as seen in practice. Given the high availability and high fidelity of NetEye, we mainly use NetEye in our measurement study, but we verify key observations using the Indriya testbed too.

Indriya testbed. Indriya [2] is deployed at three floors of the School of Computing at the National

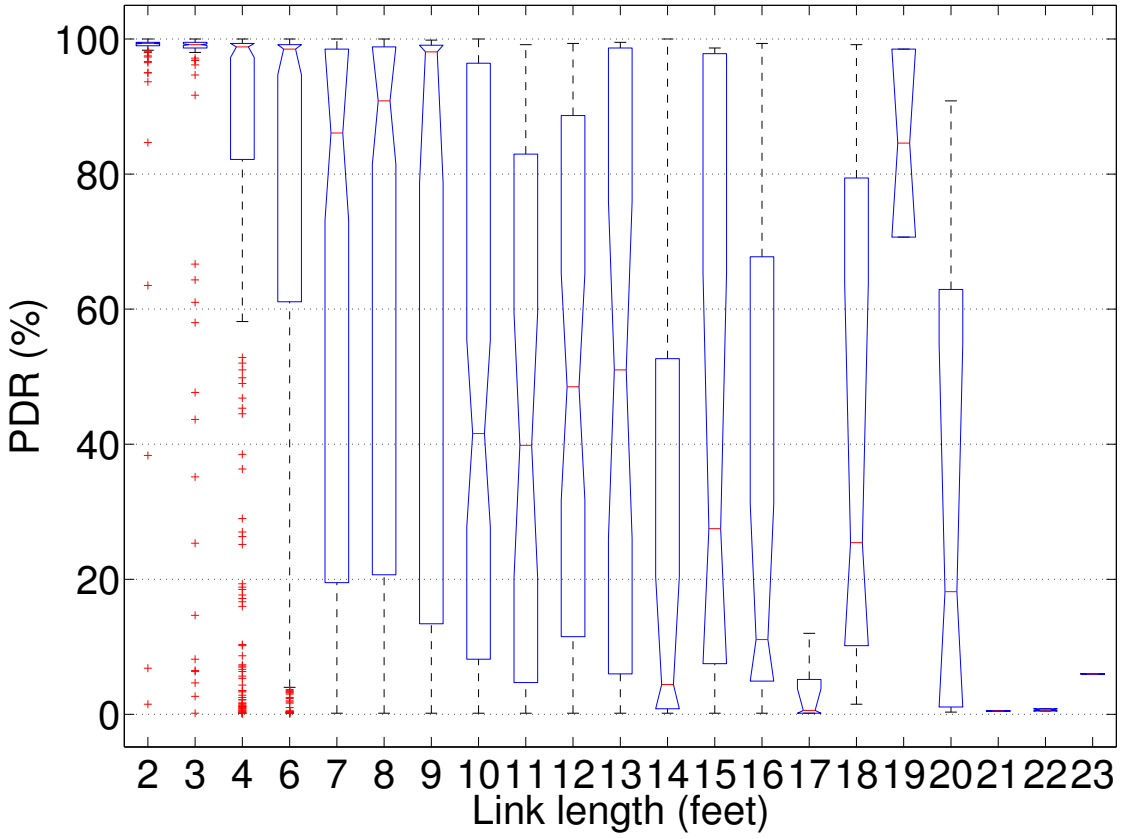

Figure 2: PDR vs. link length in NetEye when transmission power is -25dBm

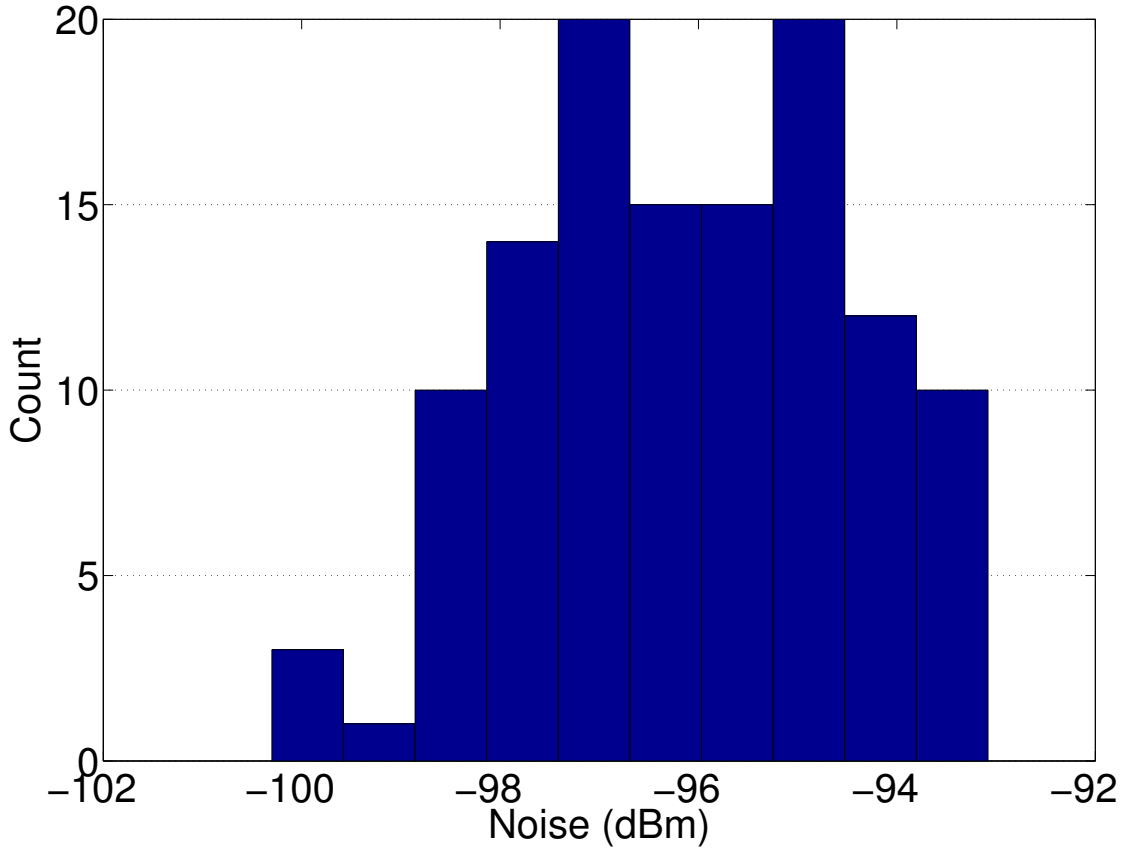

Figure 3: Histogram of background noise power in NetEye

University of Singapore as shown in Figure 4. The Testbed consists of 127 TelosB sensor motes,

Figure 4: *Indriya* wireless sensor network testbed

and Figure 5 shows a typical three-dimensional connectivity graph between the motes.

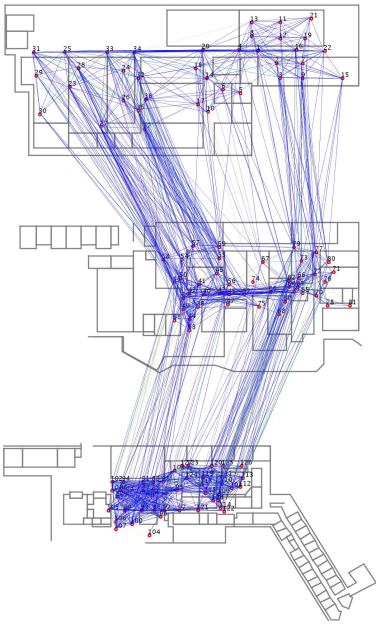

Figure 5: A typical connectivity graph for Indriya

Our measurement study uses all of its 127 TelosB motes, and we use a transmission power of -10dBm (i.e., power level 11 in TinyOS) to generate a well-connected multi-hop network where the link reliability is over 90% for links up to 20 feet long. For the transmission power of -10dBm, Figure 6 shows the boxplot of PDR for links of different length, and Figure 7 shows the histogram of background noise power in Indriya. We see that there is a high degree of variability in link PDRs and background noise power too, which reflects real-world, non-uniform settings.

Traffic pattern. Using the two testbeds, we study both periodic and event traffic patterns.

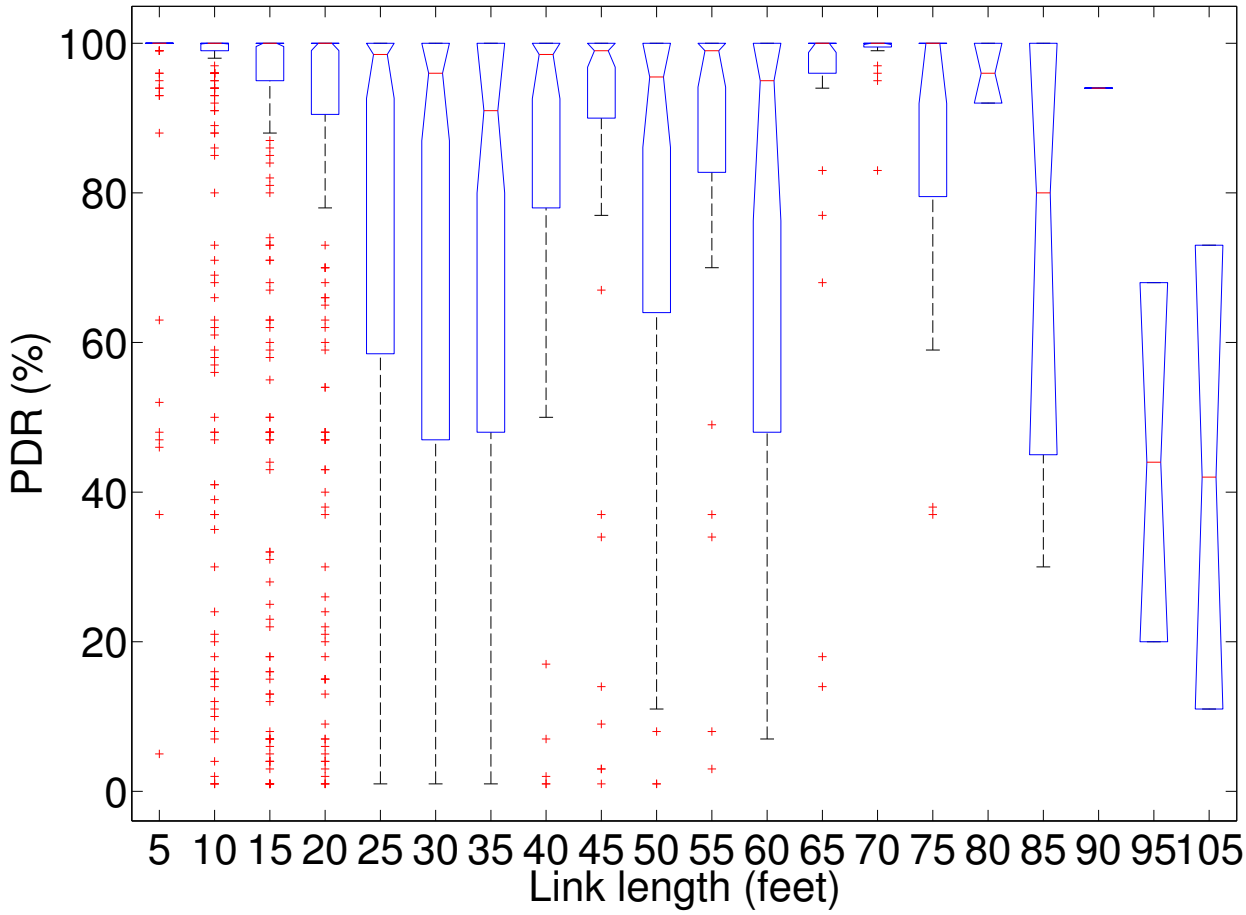

Figure 6: PDR vs. link length in Indriya when transmission power is -10 dBm

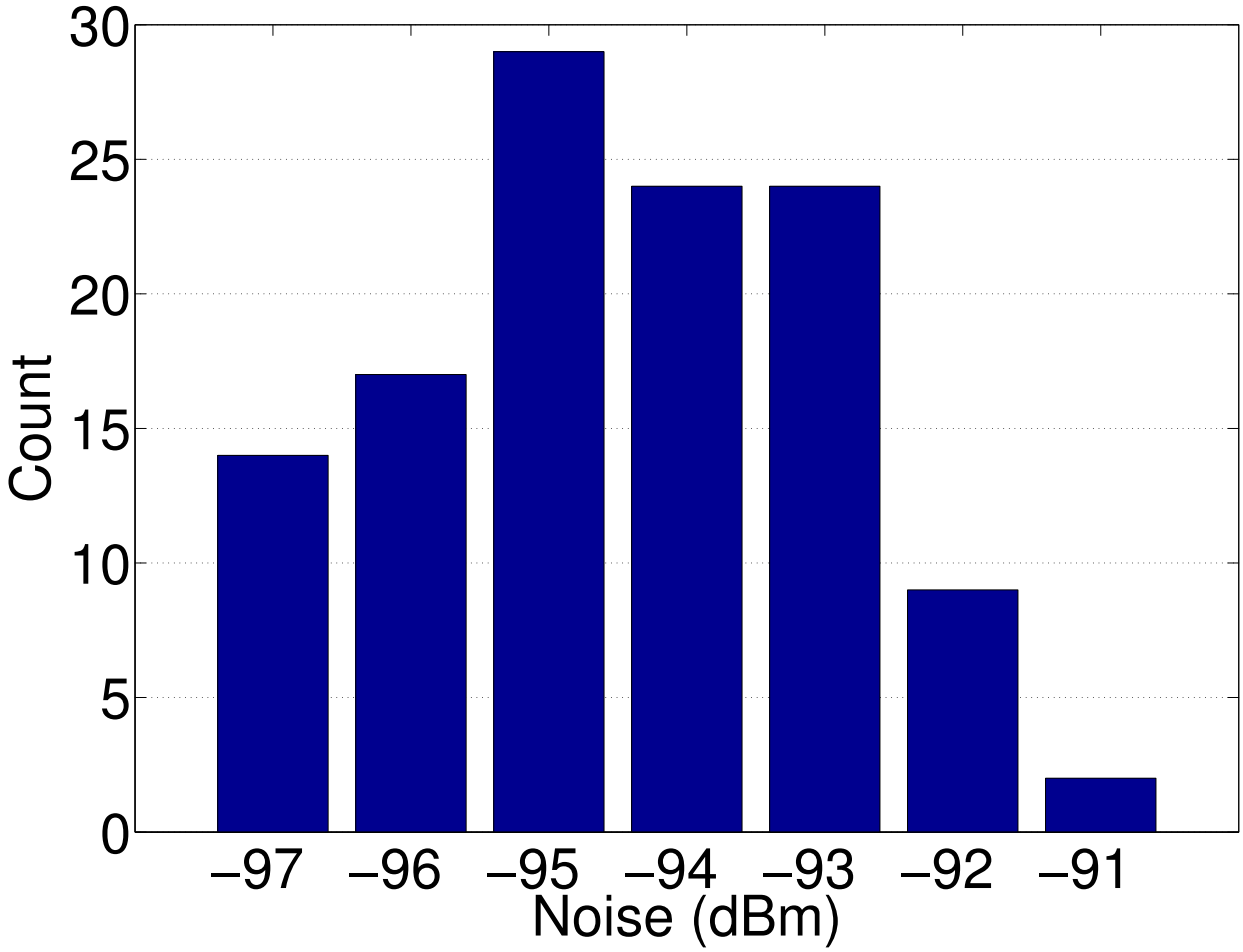

Figure 7: Histogram of background noise power in Indriya

For periodic traffic, we study three types of them based on the amount of queueing they introduce when the default TinyOS routing protocol CTP [24] is used: 1) light traffic: no queueing in network; 2) medium traffic: moderate queueing in network but with very rare queue overflow; 3) heavy traffic: severe queueing in network and with frequent queue overflow. To this end, we select one node as the sink and another 10 nodes as traffic sources; the sink and the sources are nearly at opposite positions in both testbeds to create as many routing hops as possible. In NetEye, more specifically, mote 15 is the sink, and motes 61, 62, 63, 64, 76, 77, 79, 91, 92, and 93 are the sources, with each source generating a packet every 1,000ms, 400ms, and 75ms for light, medium, and heavy traffic respectively. To verify observations from NetEye and given that the medium traffic is the common case in WSC networks, we also study medium traffic scenario in Indriya as follows: mote 105 at the third floor is the sink, and motes $1 \ldots 10$ at the first floor are the sources, with each source generating a packet every 600ms.

For event traffic, we use a publicly available event traffic trace for a field sensor network deployment [54, 1] to evaluate the performance of different protocols. The traffic trace corresponds to the packets generated in the 7×7 grid of a field mote network when a vehicle passes across the middle of the network. When the vehicle passes by, each mote except for the base station detects the vehicle and generates two packets, which correspond to the start and the end of the event detection respectively and are separated by 5-6 seconds on average. Overall, 96 packets are generated each time the vehicle passes by. The cumulative distribution of the number of packets generated during the event is shown in Figure 8. (Interested readers can find the detailed description of the traffic trace in [1].) In NetEye, the 7×7 subgrid in the trace data corresponds to the 49 motes that are the farthest from the sink mote 15. In Indriya, the 7×7 subgrid is mapped to motes $1 \ldots 49$.

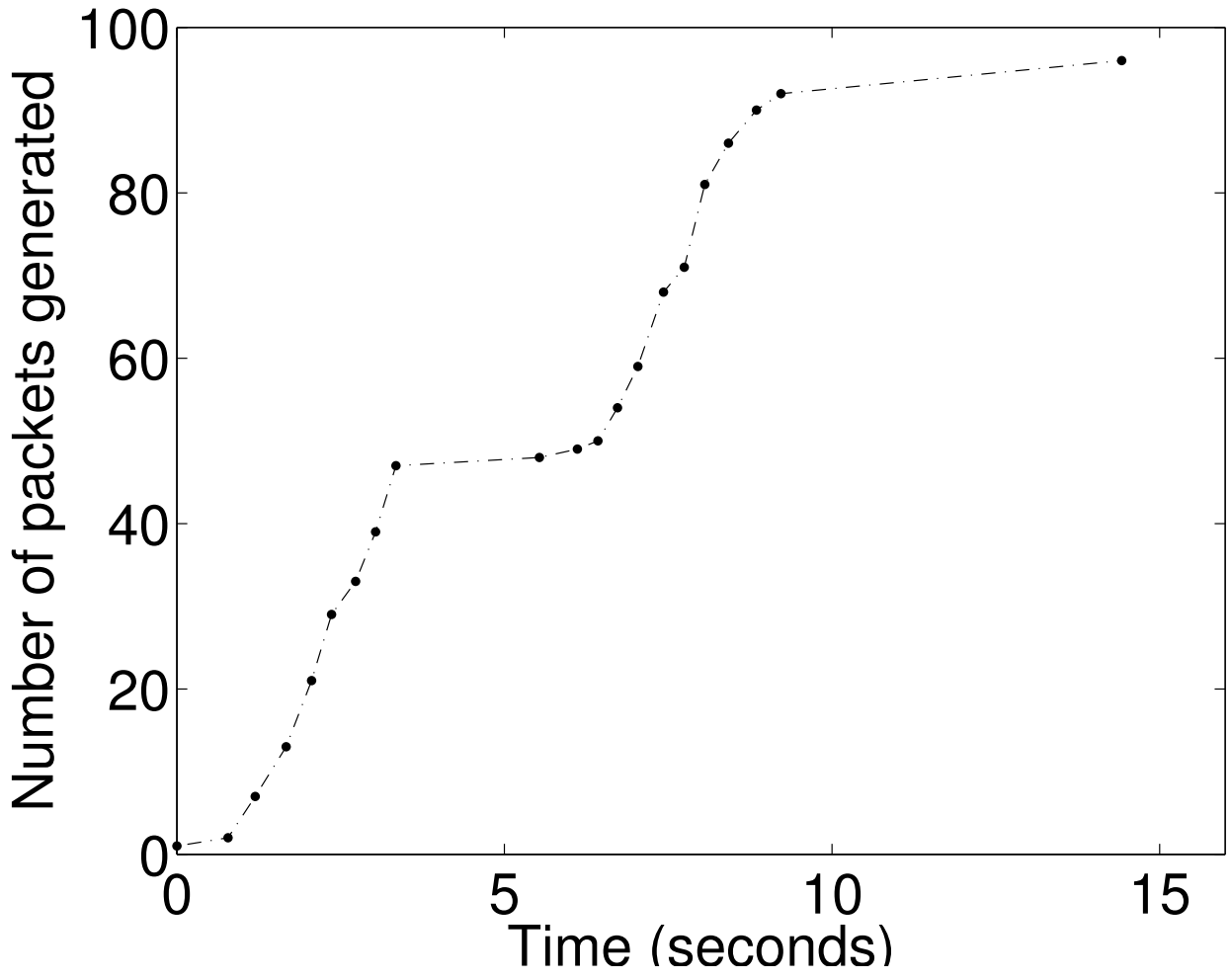

Figure 8: The distribution of packets generated in the event traffic trace

3 Multi-timescale estimation of path delays

In what follows, we first discuss the need for considering probabilistic path delays in real-time routing, then we present our approach to accurate, agile estimation of probabilistic path delay bounds.

3.1 Why probabilistic path delay?

Due to inherent spatiotemporal dynamics and uncertainties in wireless communication [58, 55], the success of a packet transmission is probabilistic instead of deterministic. Thus, data delivery delays across links/paths are probabilistic in nature in wireless sensing and control (WSC) networks, and this is the case even if TDMA- instead of CSMA-based scheduling is used. Accordingly, application requirements on data delivery timeliness tend to be probabilistic in WSC networks. Based on its requirements on stability region and settling time, for instance, a control application can specify a maximum tolerable delay; by treating a deadline miss as the loss of the corresponding signal sample and by analyzing the corresponding impact on system estimation and control, the application can also specify a maximum tolerable probability of deadline miss [48, 27]. Therefore, we consider application requirements on the maximum tolerable delay and the least probability of deadline success in this study.

Most existing delay-aware wireless routing protocols [51, 22, 26, 36, 52] consider mean delay instead of delay quantiles. Yet smaller mean delay does not ensure smaller delay quantiles. For the medium traffic setting in NetEye as discussed in Chapter 2, for instance, Figure 9 shows the non-negligible probability that, when the mean delay across a link ℓ_0 is less than that of another link ℓ_1, the q-quantile of ℓ_0's delay is greater than that of ℓ_1. Therefore, routing based on mean delay does not ensure the use of paths with small probabilistic delays, thus leading to deadline miss and the reduction of network real-time capacity (i.e., amount of data that can be delivered within a certain deadline).

Traditional real-time system design usually considers worst cases, and, in real-time wire-

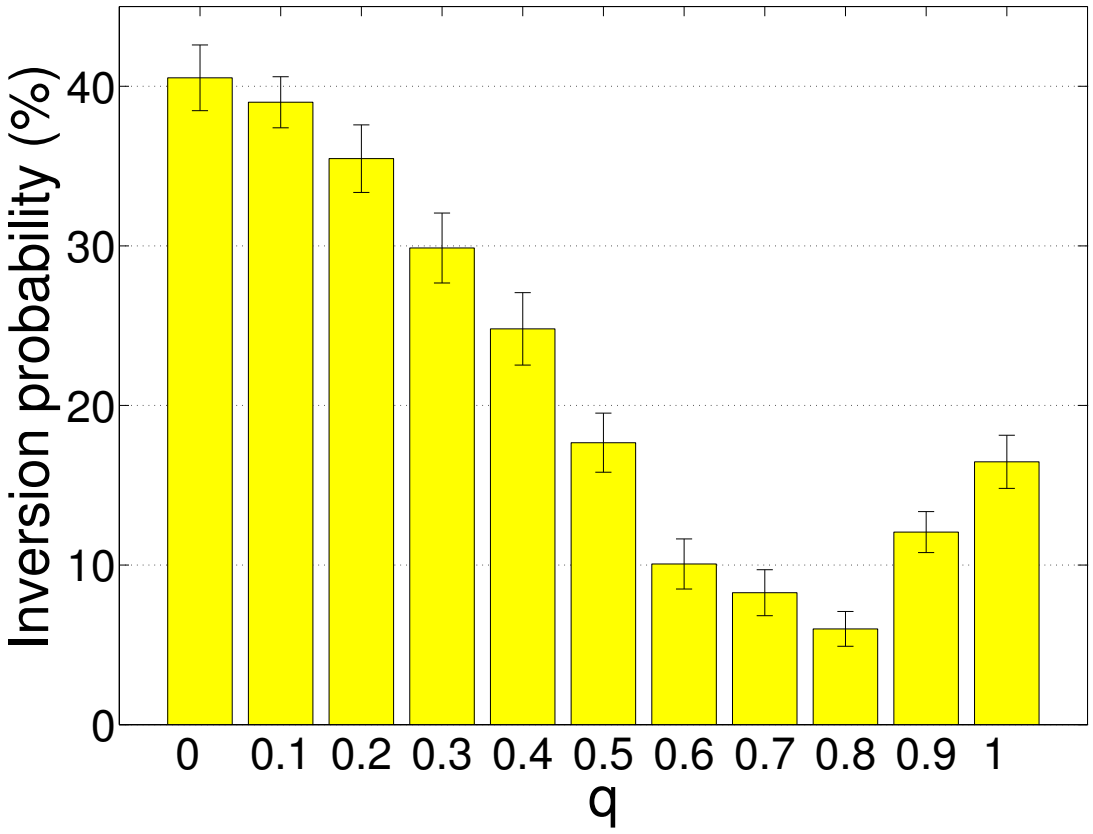

Figure 9: Goodness inversion probability and its 99% confidence interval

less networking, worst-case path delay has also been considered [43]. Yet the distribution of link/path delays tend to be heavy-tailed such that the maximum delay is significantly larger than the common-case delays. For the medium traffic setting in NetEye, for instance, Figure 10 shows the empirical cumulative distribution function (CDF) of the per-packet transmission time (which we refer to as *packet-time*) for successfully delivering a packet across a typical link in NetEye.[1] The median, 90-percentile, and maximum of the packet-time are 23, 86, and 728 milliseconds

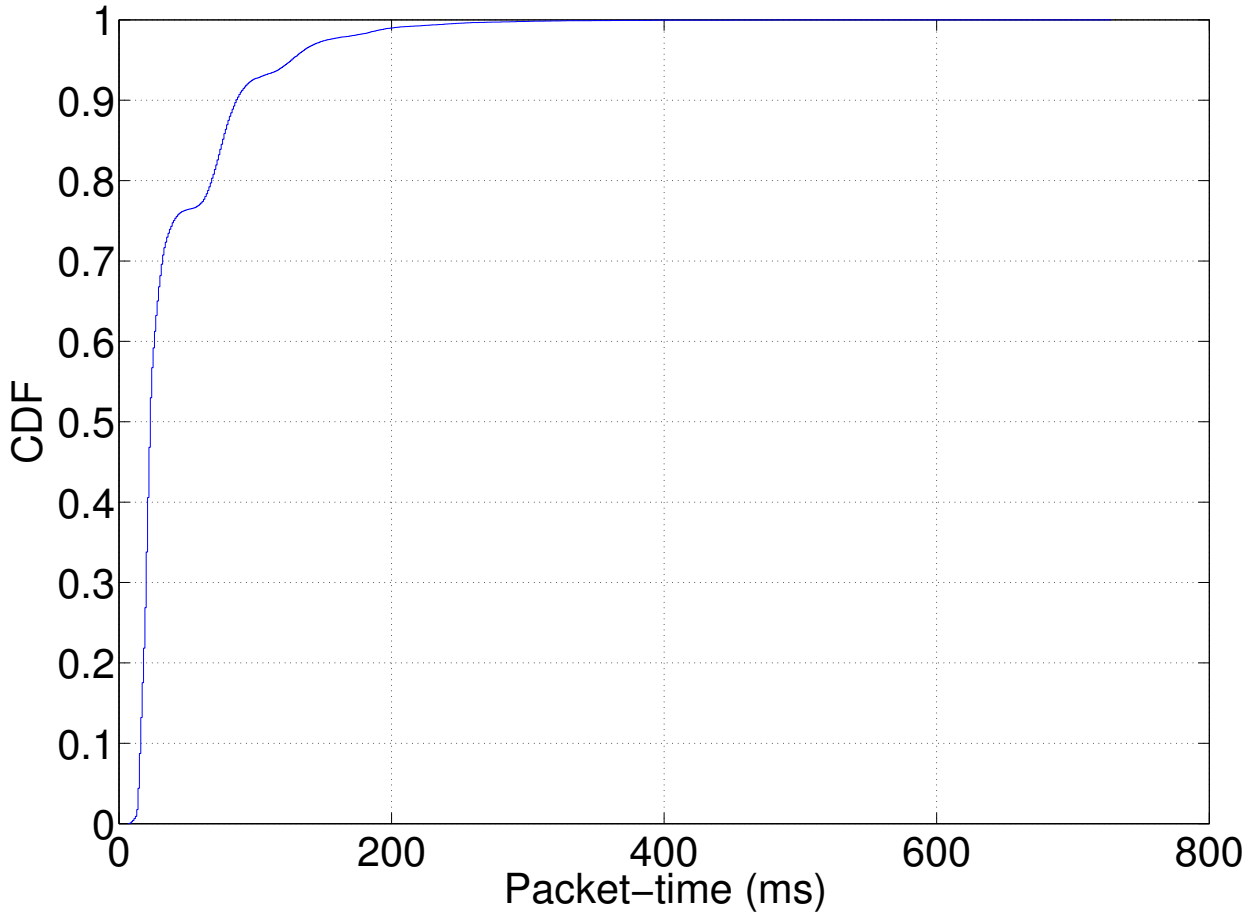

Figure 10: CDF of the packet-time across a typical link

respectively. Therefore, focusing on maximum link/path delays will not be able to utilize many paths that could have been used to support WSC applications where probabilistic delay guarantees suffice, thus reducing the application-usable real-time capacity of the network.

[1]The packet-time includes time spent for channel access contention and possibly packet retransmission.

Therefore, considering probabilistic path delays instead of mean or maximum delays reduces deadline misses and enables effective utilization of application-usable real-time capacity of WSC networks.

3.2 Agile, accurate estimation of probabilistic path delay bound

To enable routing with probabilistic delay guarantees, a basic task is to quantify probabilistic delays along paths. Given a path $P = v_0, v_1, \ldots, v_n, v_{n+1}$ from v_0 to v_{n+1} as shown in Figure 11 and

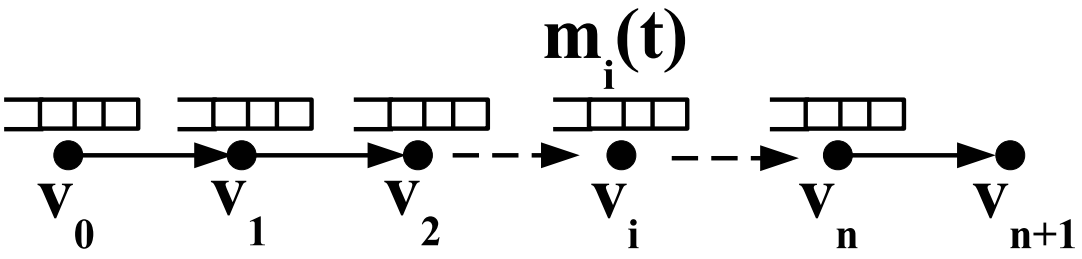

Figure 11: An example path P

assuming that the number of packets queued at node $v_i (i = 0 \ldots n)$ is $m_i(t)$ at time t and that the packet transmission scheduling algorithm at each node is first-come-first-serve (FCFS), the instantaneous delay along path P at time t, denoted by $d_P(t)$, is the delay that a packet arriving at v_0 at time t experiences in reaching the destination node v_{n+1} assuming that the number of queued packets at each node does not change while the packet is being delivered to v_{n+1}. Therefore,

$$d_P(t) = \sum_{i=0}^{n} \sum_{j=1}^{m_i(t)+1} d_i^j(t), \tag{1}$$

where $d_i^j(t)$ is the time taken for v_i to deliver its j-th queued packet to v_{i+1} at time t.

Challenges of highly-varying delay distribution. Given the distributions of $d_i^j(t) (i = 0 \ldots n, j = 1 \ldots m_i(t) + 1)$, it is NP-hard to compute the distribution of $d_P(t)$ [25]. Since nodes

of WSC networks tend to be resource-constrained (e.g., in CPU and memory), it is infeasible to directly compute the distribution of $d_P(t)$ in general. One alternative is to first sample $d_P(t)$ and then use non-parametric approaches (e.g., the P^2 algorithm) to estimate delay quantiles based on these delay samples [31, 38]. Nonetheless, non-parametric quantile estimation usually converges slowly. For instance, Figure 12 shows, as path delay samples are collected for a typical five-hop

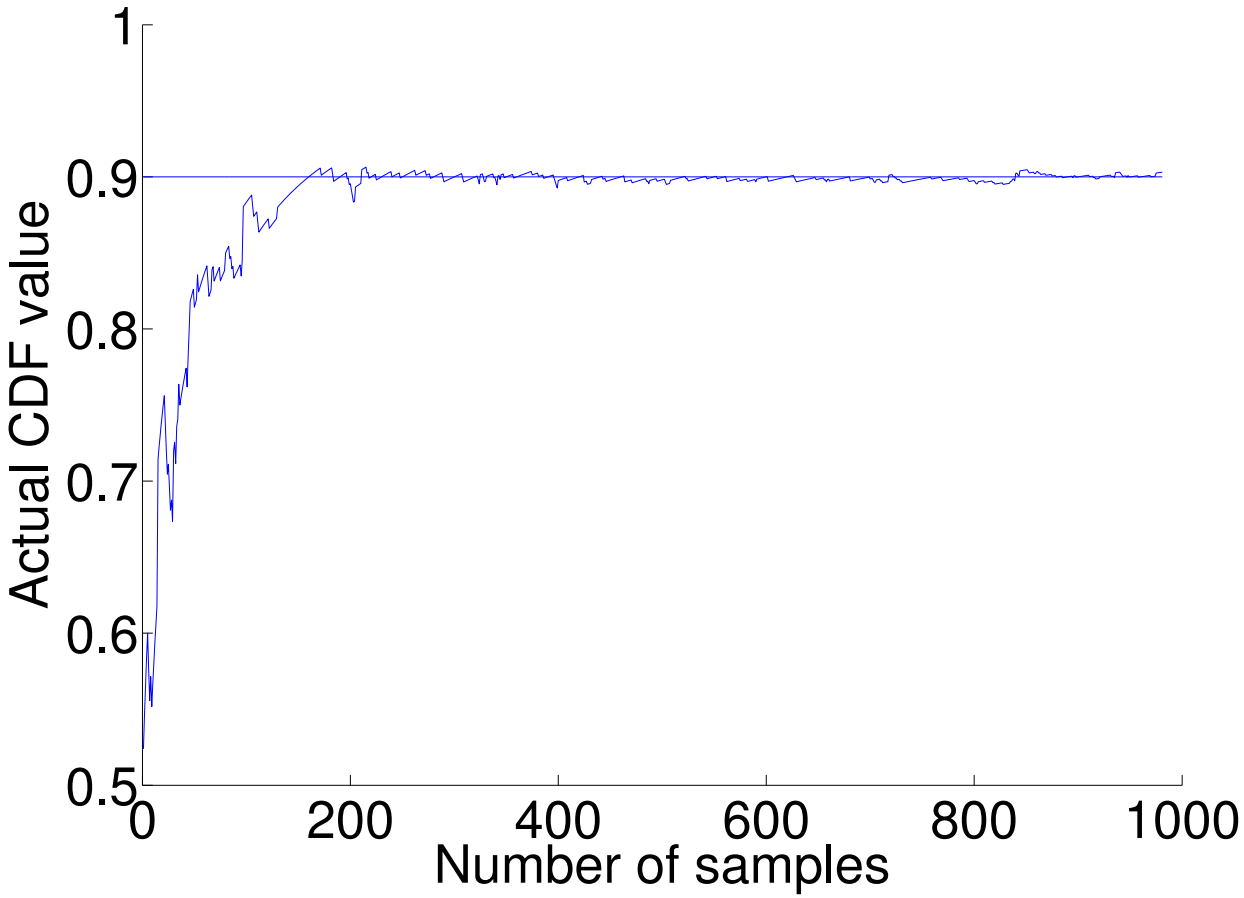

Figure 12: Actual CDF value of the estimated 90 percentile via the P^2 algorithm

path in the NetEye medium traffic scenario, the time series of the actual cumulative distribution function (CDF) values of the estimated 90 percentiles using the P^2 algorithm. We see that it takes more than 200 samples for the estimation to converge, and this observation holds in general [38]. Yet the distribution of path delays varies at a much shorter timescale than the sample size required for non-parametric quantile estimation to converge. As can be seen from (1), in particular, the

distribution of path delays vary with the queueing levels (i.e., $m_i(t)$) along paths, and network queueing can vary quickly over time.[2] For the NetEye medium traffic scenario, for instance, Figures 13 and 14 show the histogram of the coherence window size for node and path queueing

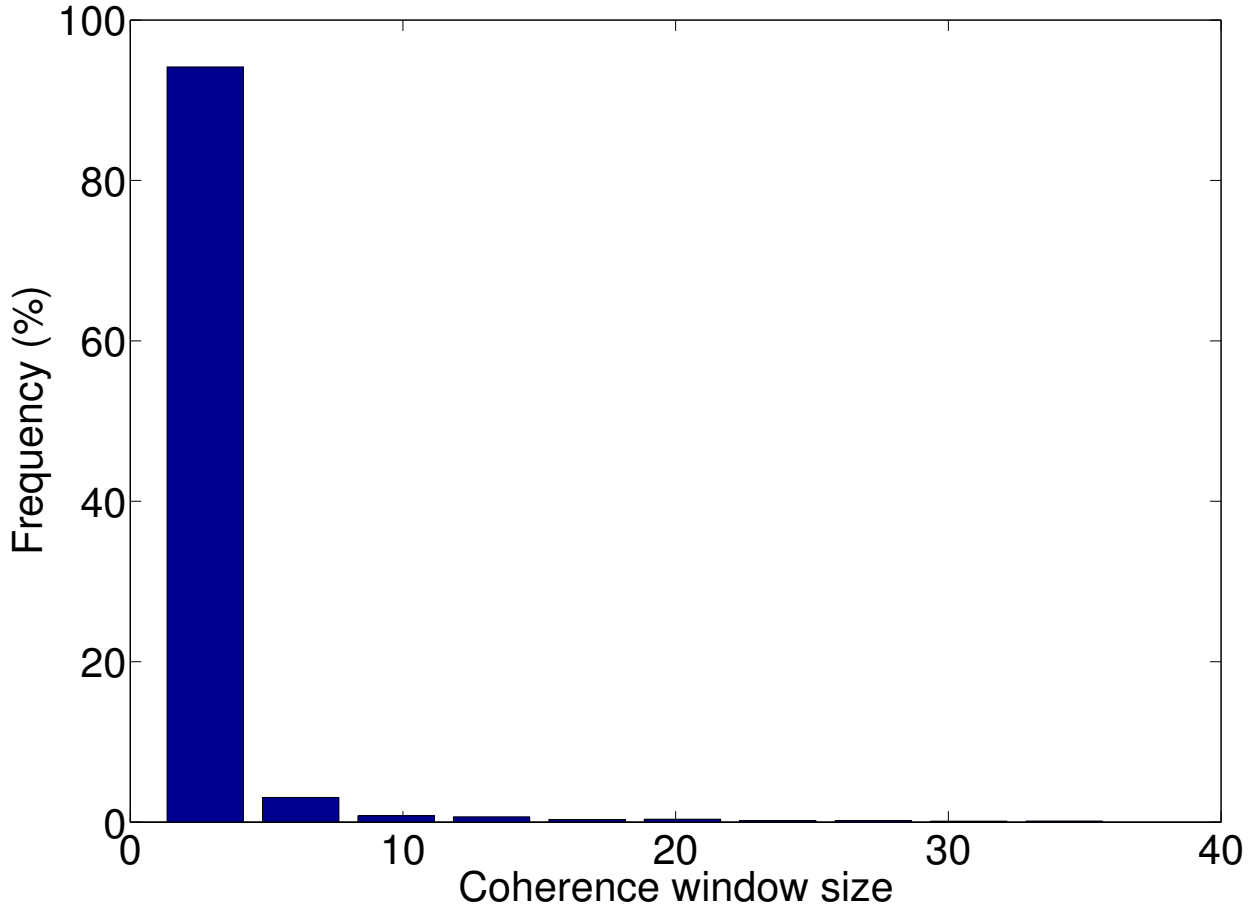

Figure 13: Histogram of the coherence window size for node queueing level

respectively, where each coherence window of a time series is a maximal consecutive segment of the time series where all the sample values are the same. We see that the coherence window size is two orders of magnitude less than the sample size required for non-parametric quantile estimation to converge. Therefore, the highly-varying nature of path delay distribution makes non-parametric quantile estimation unable to accurately estimate instantaneous path delay quantiles in an agile manner; that is, the distribution of the path delay changes before the estimation converges to an

[2] As we will discuss shortly, the distribution of packet-time (i.e., $d_i^j(t)$) is quite stable over time, and we leverage the stability of packet-time distribution in designing our multi-timescale estimation (MTE) method.

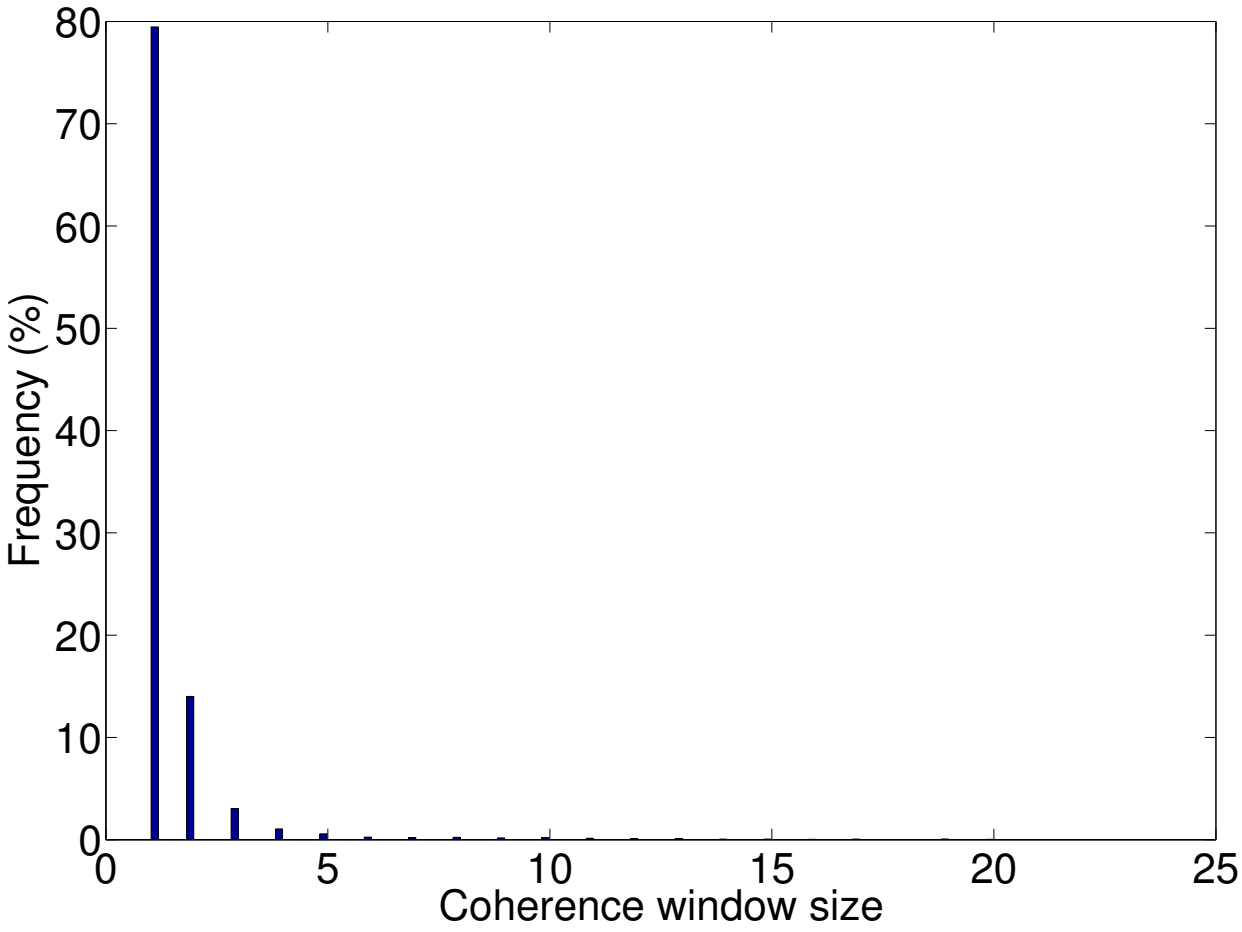

Figure 14: Histogram of coherence window size for path queueing level

accurate value.

To circumvent the computational complexity and the inability of accurate, agile estimation of exact path delay quantiles, we propose to identify upper bounds on probabilistic path delays and to use the delay bounds in identifying paths for real-time data delivery. As we will discuss later in this chapter , upper bounds on path delays can be derived using probability inequalities such as Chebyshev Inequality [44], and the properly identified delay bounds are still orders of magnitude less than the maximum delays, thus enabling effective utilization of network real-time capacity. Most probability inequalities use the mean and/or the standard deviation of the corresponding random variable, thus we need an accurate, agile mechanism of estimating the mean and standard deviation of path delays. Nonetheless, the sample size required for accurate estimation of path delay statistics tend to be quite large. For instance, Figure 15 shows the empirical CDF of the sample size required for estimating mean path delay at 90% accuracy and 90% confidence level. We see that the required sample size is greater than 100 in most cases and can be up to 594, which are significantly greater than the coherence window size of node/path queueing levels. In addition, node and path queueing levels tend to have low autocorrelation, not to mention staying unchanged, for time lags greater than 100 and 40 samples respectively, as shown in Figures 16 and 17. Thus, the large sample size requirement and the highly-varying nature of path delay distribution makes it impossible to accurately estimate instantaneous mean path delay in an agile manner; that is, the mean path delay changes before the estimation converges to an accurate value.

Multi-Timescale Estimation (MTE). Towards addressing the challenges of highly-varying distribution and statistics of path delay, we decompose contributors to path delay variations into two factors: dynamic packet-time and dynamic queueing. By leveraging the different timescales at which packet-time and queueing vary, we propose the multi-timescale-estimation (MTE) method that accurately estimates the highly-varying mean and variance of path delay by 1) accurately estimating the mean and variance of packet-time at a longer timescale and 2) by adapting to fast-varying queueing at a shorter timescale. In what follows, we elaborate on the design of the MTE method.

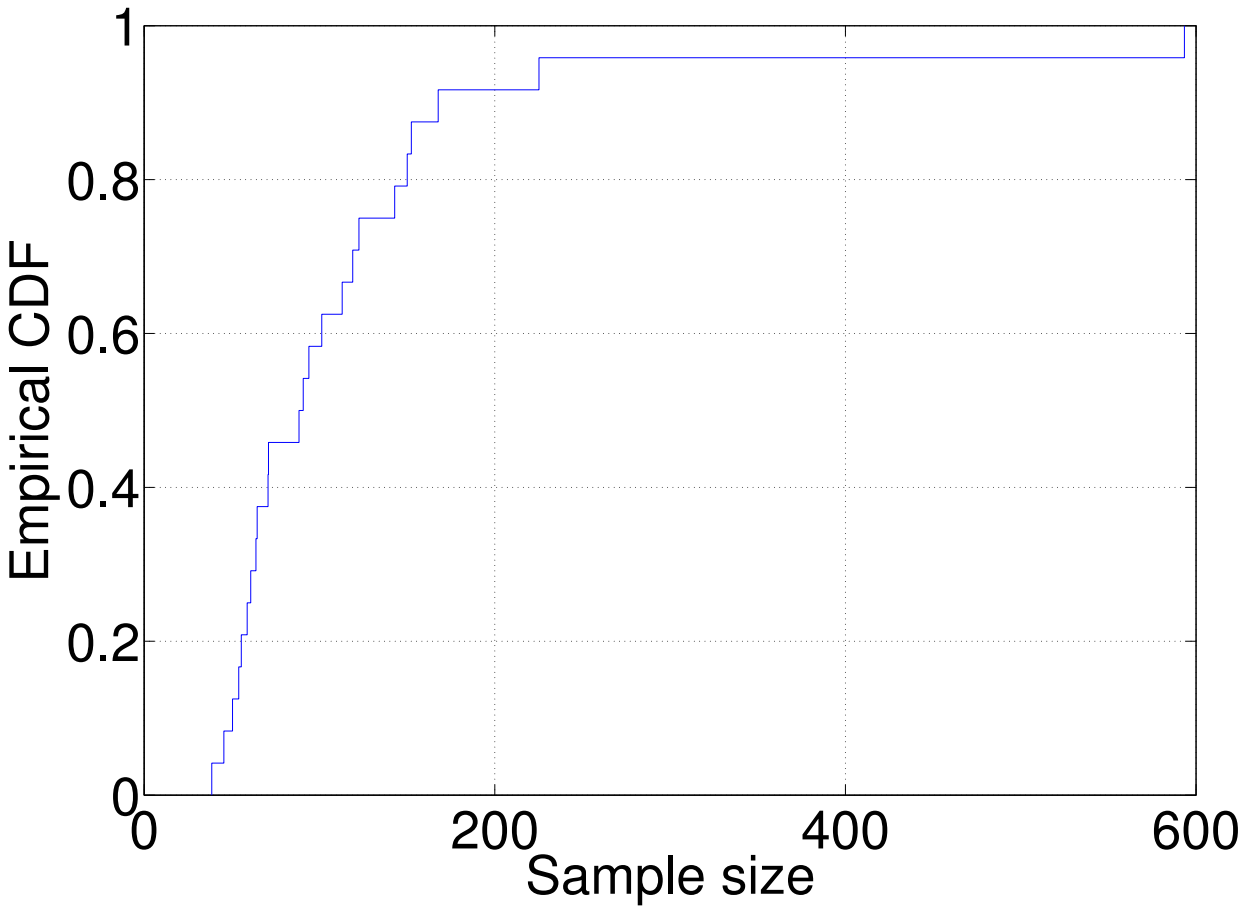

Figure 15: Empirical CDF of the sample size required to estimate mean path delay at 90% accuracy and 90% confidence level

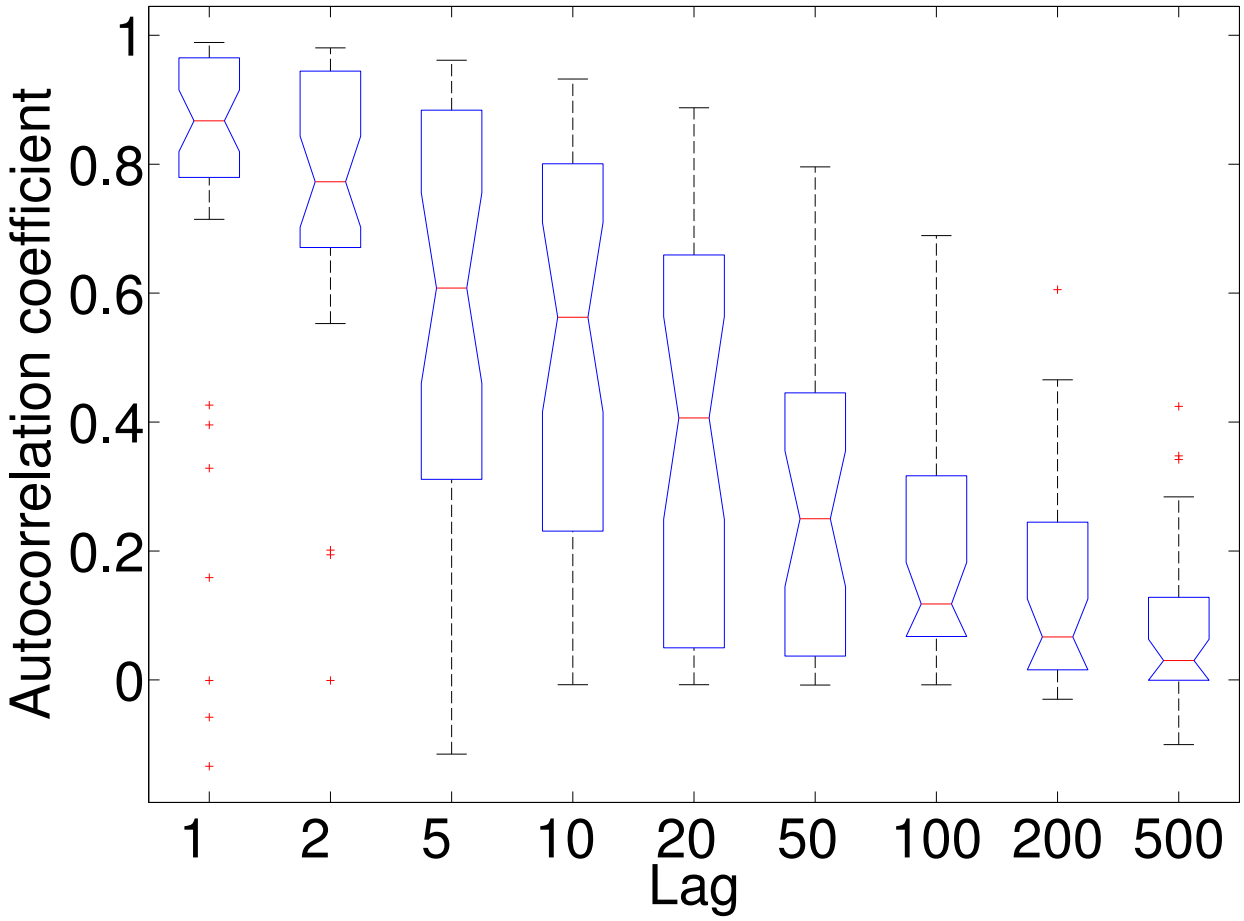

Figure 16: Autocorrelation of node queueing levels

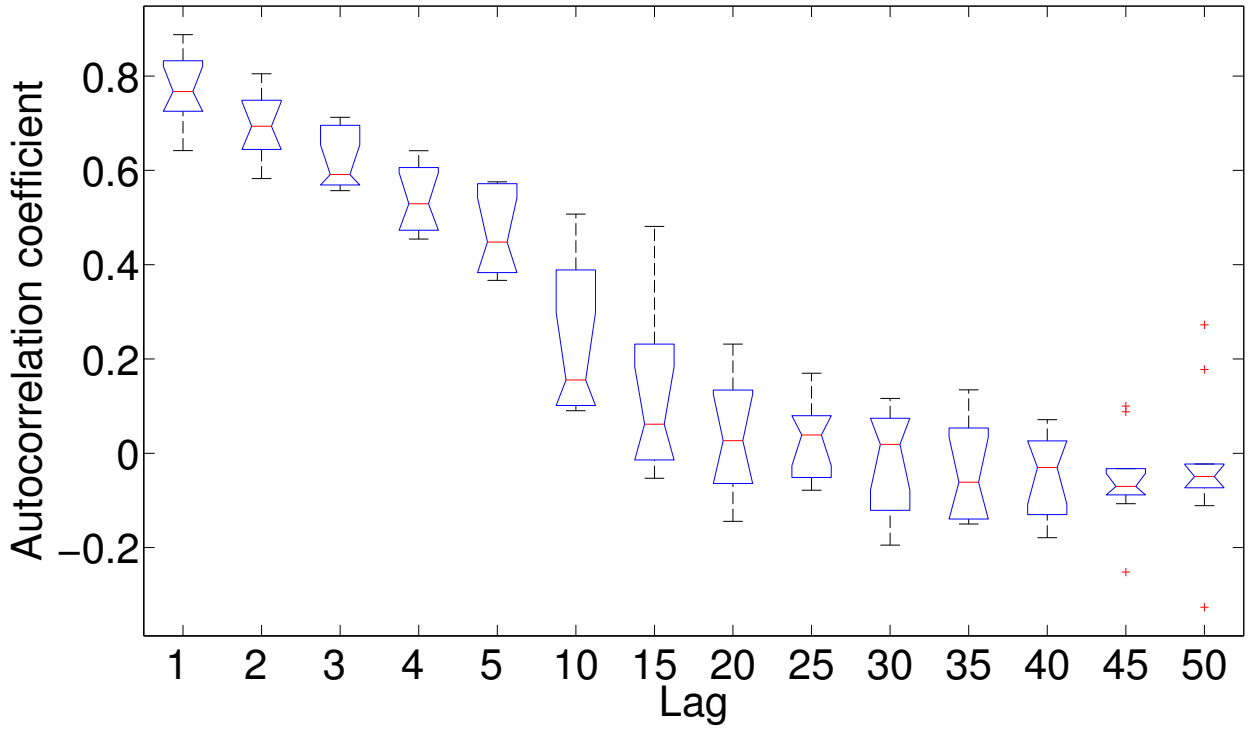

Figure 17: Autocorrelation of path queueing levels

Observation 1. *Packet-time distribution is stable.*

Through detailed experimental analysis, we find that, given a network condition, the distribution of packet-time is quite stable despite the quick variation of instantaneous packet-time. For instance, Figure 18 shows the time series of the packet-time along a typical link in the NetEye medium traffic scenario. Using the Generalized KPSS test [28], we analyze the stationarity window size of packet-time, where each stationary window of a time series is a maximal consecutive segment of the time series that is weak-sense stationary (i.e., with a constant mean and variance over time). Figure 19 shows the histogram of the stationarity window size for packet-time in the NetEye medium traffic scenario. The minimum and the maximum window size are 1,901 and 21,937 respectively, both of which are significantly greater than the sample size required to precisely estimate the mean and variance of packet-time. For instance, Figure 20 shows the empirical CDF of the sample size required for estimating the mean packet-time at 90% accuracy and 90% confidence level. The sample size requirement is less than 360 with over 99% probability, and the

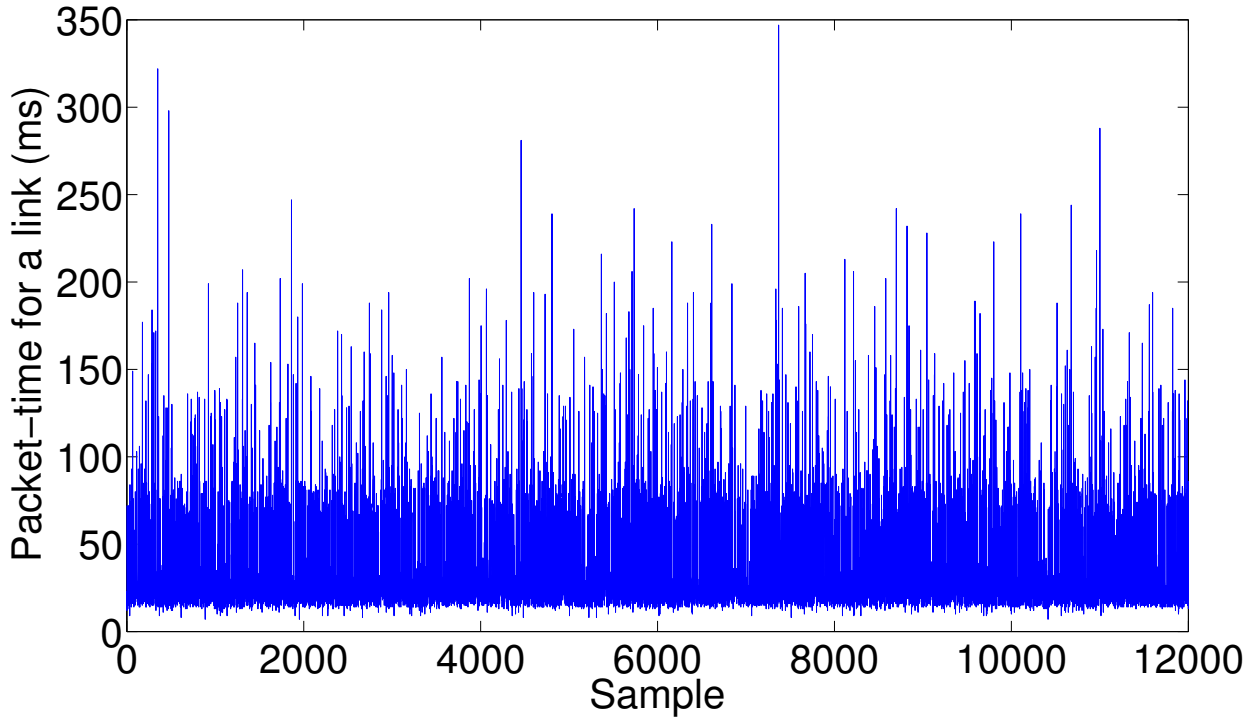

Figure 18: Time series of packet-time along a typical link

maximum sample size requirement is 600.

The stability of the packet-time distribution has two important implications:

- Given the real-time requirement on data delivery and the constrained memory size in embedded WSC networks, the stationarity window size tends to be greater than the maximum node queue size in general. This implies that, in Formula (1), the mean and variance for $d_i^j(t)$'s $(j = 1 \ldots m_i(t) + 1)$ are the same for a given $i (i = 0 \ldots n)$, and we denote them as $\mu(d_i(t))$ and $\sigma^2(d_i(t))$. Therefore, the mean path delay can be computed as follows:

$$\mu(d_P(t)) = \sum_{i=0}^{n} (m_i(t) + 1)\mu(d_i(t)). \tag{2}$$

- The stability of packet-time distribution enables each node to accurately estimate the mean and variance of packet-time from itself to the next-hop along a path, and, with Formula (2), the mean path delay can be accurately estimated.

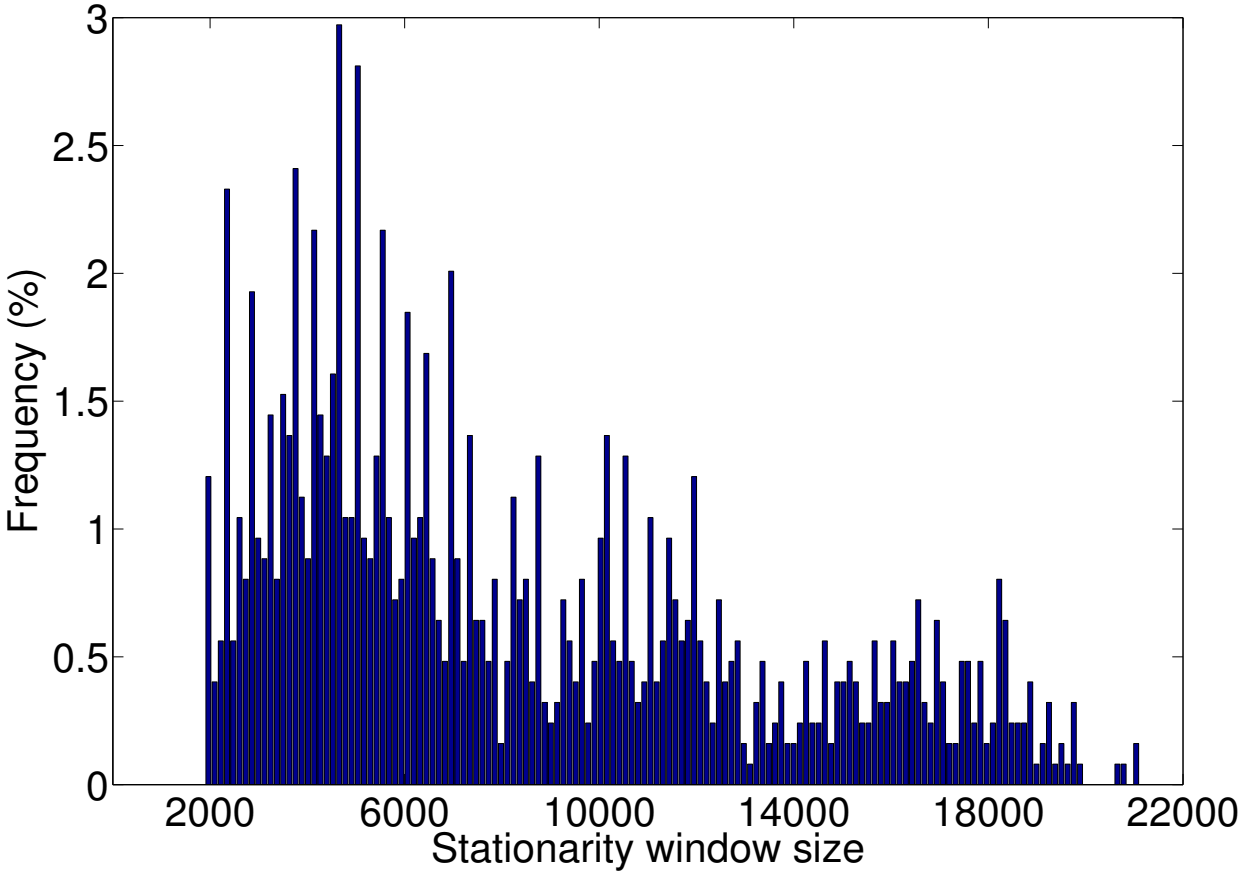

Figure 19: Histogram of stationarity window size for packet-time

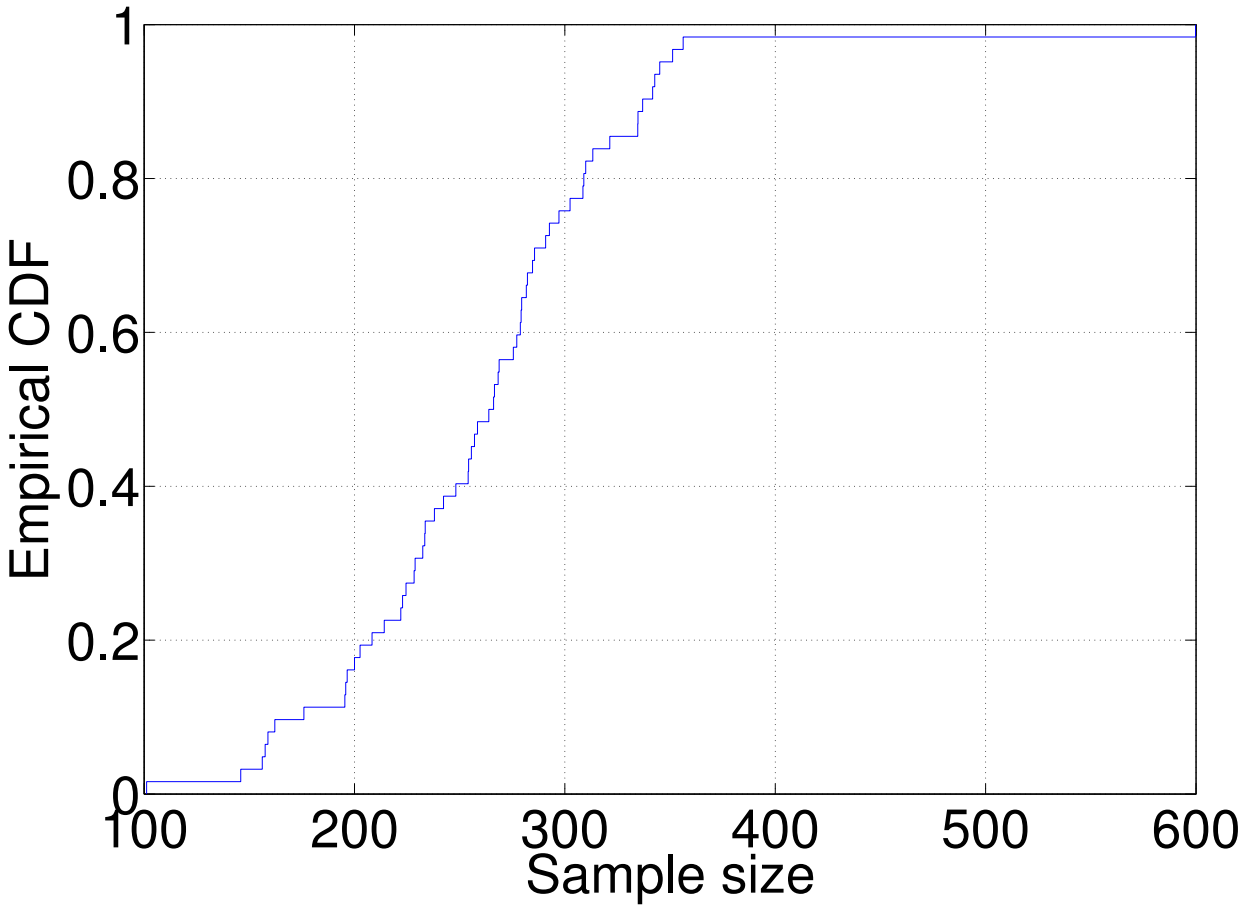

Figure 20: Empirical CDF of the sample size required for estimating the mean packet-time at 90% accuracy and 90% confidence level

Observation 2. *Packet-time is uncorrelated.*

We also observe that the packet-time for different transmissions, whether from the same node or from different nodes, tend to be uncorrelated. For instance, Figure 21 shows the small auto-

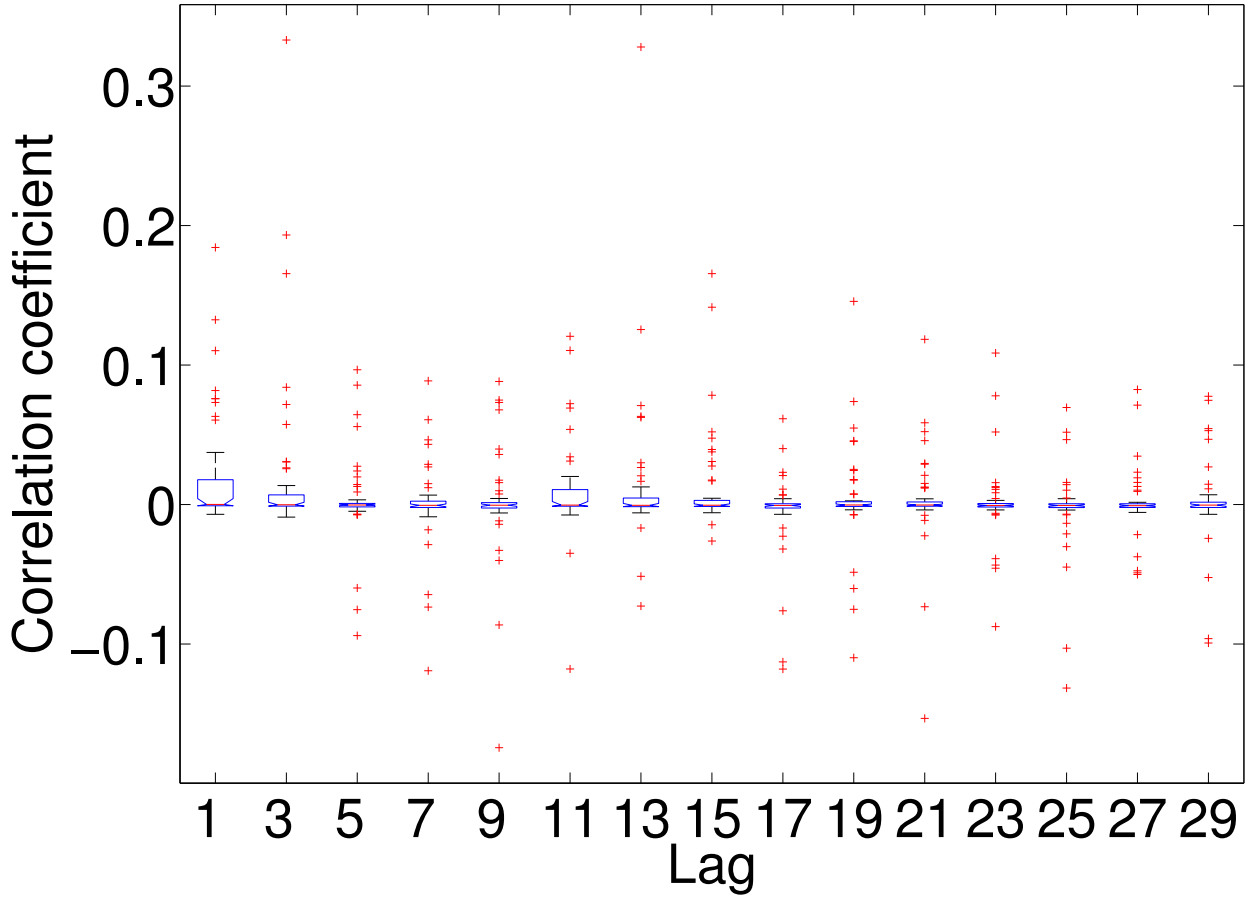

Figure 21: Autocorrelation coefficient of packet-time along the same link

correlation coefficient of packet-time along the same link, with the boxplot showing, for each lag, the distribution of autocorrelation coefficients across different links; Figure 22 shows the small correlation coefficient of packet-time along different links, where the lag is defined as the number of packets queued between two packets at different node queues as shown in Figure 11, assuming that the packets queued at time t flow through the nodes queues in a FIFO manner. We also observe that the median autocorrelation coefficient and the median cross-link correlation coefficient of packet-time is zero at 99% confidence level. An intuition for the uncorrelatedness between

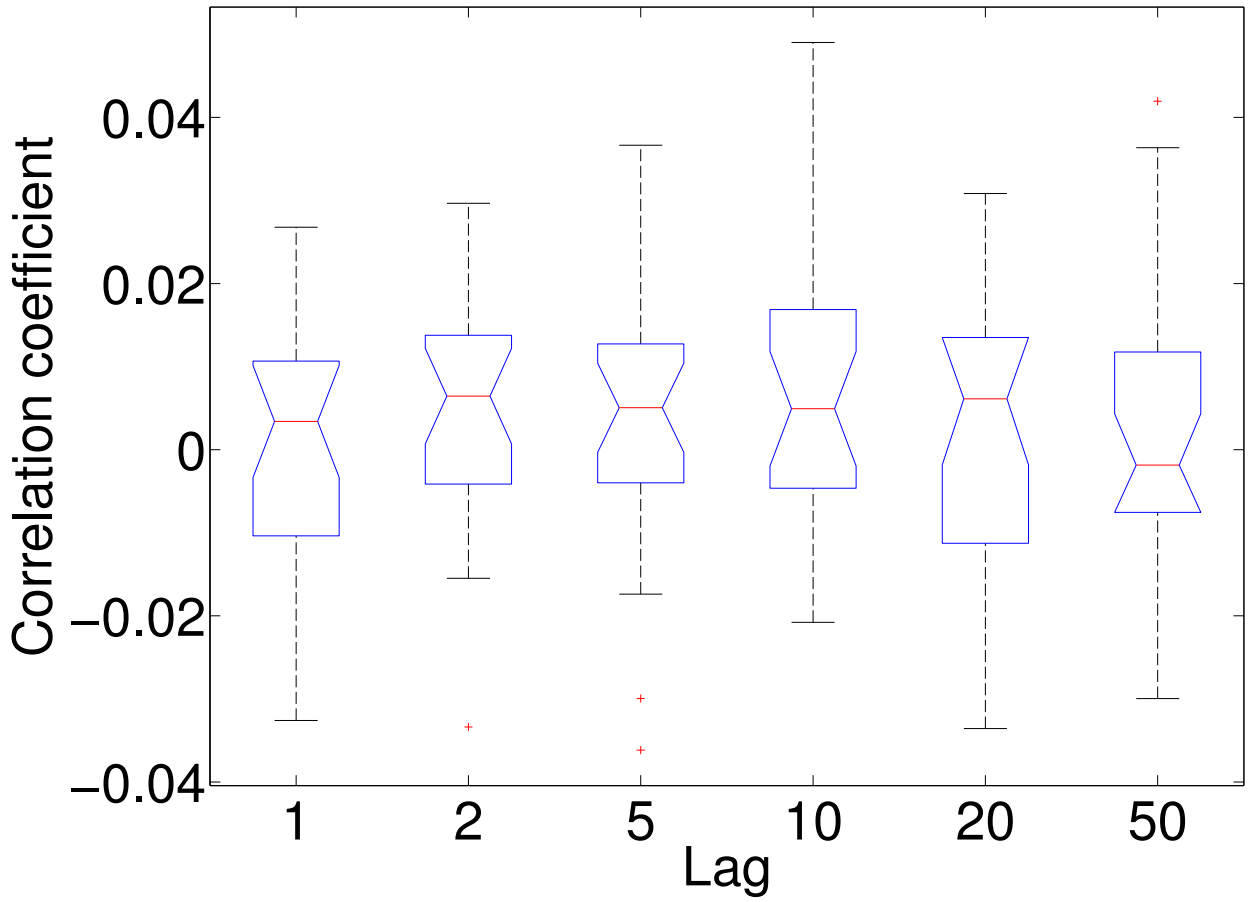

Figure 22: Correlation coefficient between the packet-time across different links along a path

packet-time is that, given a network condition, the behavior of one packet transmission does not have much impact on the behavior of another packet transmission as far as the MAC protocol is concerned.

The Bienaym$\acute{e}$ Formula [40] shows that the variance of the sum of pair-wise uncorrelated random variables is the sum of the variances of the individual random variables. Thus our observations on the uncorrelatedness of packet-time along a path enables each node to compute, for a given time instant, the standard deviation of the path delay from itself to the destination node as the square root of the sum of the variances of the packet-times for all the packets queued along the path at that instant. For path P of Figure 11, for instance, the standard deviation of $d_P(t)$ can be computed as follows:

$$\sigma(d_P(t)) = \sqrt{\sum_{i=0}^{n}(m_i(t)+1)\sigma^2(d_i(t))}. \tag{3}$$

For a typical 5-hop path in NetEye, Figure 23 shows the histogram of the relative errors in estimating the standard deviation of path delay in the presence of different queueing levels along the path. We see that the estimation is quite accurate, with most relative errors within the range of (-0.075, 0.075). Note that, if we directly estimate the variance of the sojourn time $d_i^{so}(t) = \sum_{j=1}^{m_i(t)+1} d_i^j(t)$ at node v_i without decomposing $d_i^{so}(t)$ into its individual components $d_i^j(t)$'s ($j = 1 \ldots m_i(t)+1$), we cannot compute $\sigma(d_P(t))$ as $\sqrt{\sum_{i=0}^{n}\sigma^2(d_i^{so}(t))}$. This is because, for $i1 \neq i2$, $d_{i1}^{so}(t)$ and $d_{i2}^{so}(t)$ are correlated due to the correlation between queueing levels at different nodes of a path.

In the simplified scenario of Figure 11, the next-hops of all the packets in a node queue are the same. In reality, a node may well use different next-hops for different packets, for instance, depending on packet deadlines as we will discuss in Chapter 4. Assume that, at time t, each node $v_i (i = 0 \ldots n)$ has $N_i(t)$ number of next-hops, the number of packets (including the one arriving at v_0 at time t) to be forwarded to its k-th next hop is $m_i^k(t)$, and the packet-time from v_i to its k-th next-hop is $d_{i,k}(t)$. We observe that, given v_i and t, $d_{i,k}(t)$'s tend to be uncorrelated for different ks. For instance, Figure 24 shows the histogram of the correlation coefficient between the packet-time across different outgoing links from the same node. The correlation coefficient is very small and

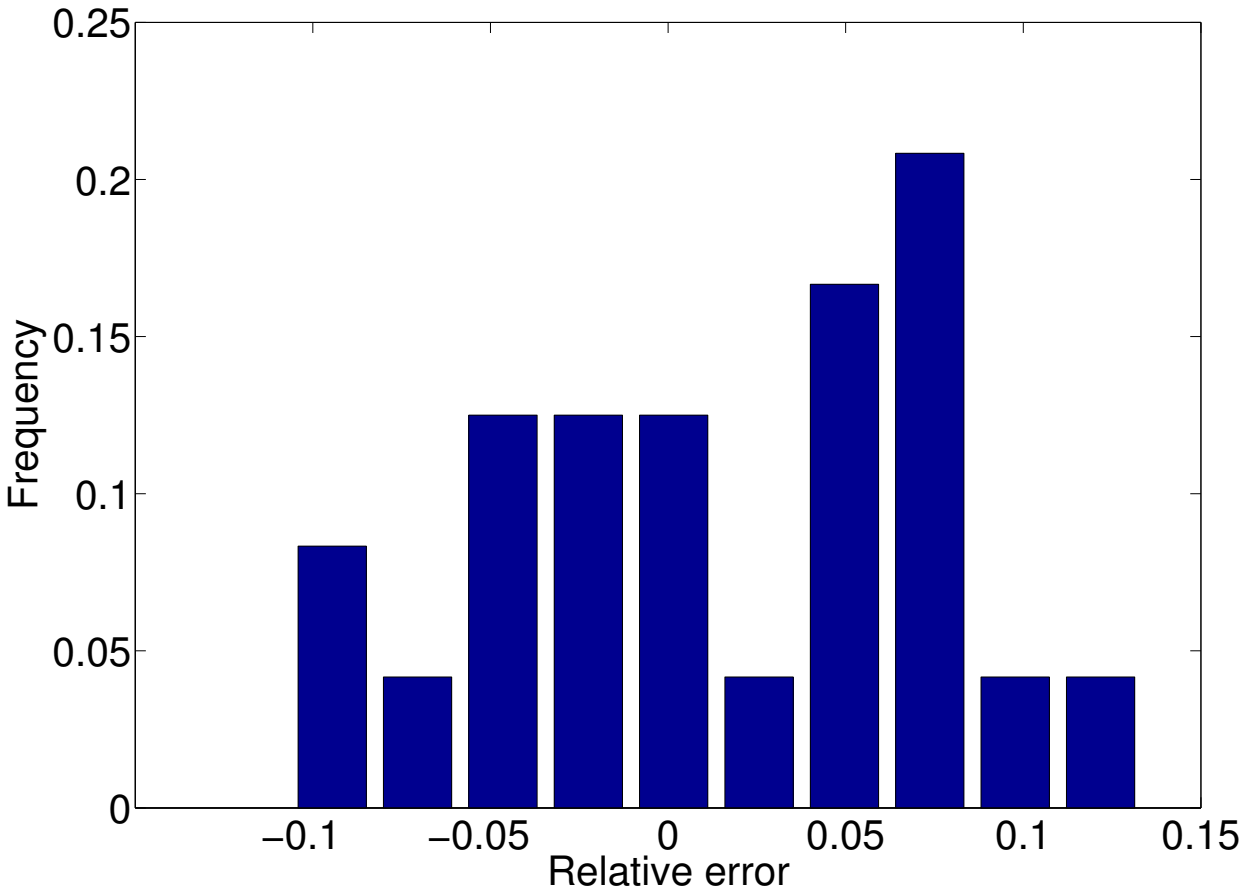

Figure 23: Histogram of relative errors in estimating the standard deviation of path delay

is less than 3% most of the time. We also observe that the median correlation coefficient is zero at 99% confidence level. Then, based on Formulae (2) and (3), the mean and standard deviation of the delay along a path P can be computed as follows:

$$\begin{aligned} \mu(d_P(t)) &= \sum_{i=0}^{n} \sum_{k=1}^{N_i(t)} m_i^k(t) \mu(d_{i,k}(t)) \\ \sigma(d_P(t)) &= \sqrt{\sum_{i=0}^{n} \sum_{k=1}^{N_i(t)} m_i^k(t) \sigma^2(d_{i,k}(t))} \end{aligned} \tag{4}$$

Observation 3. *Network queueing is relatively stable at short timescales.*

To leverage Formula (4) in estimating the mean and variance of the delay along path P in a distributed manner, each node v_i can compute the mean and variance of the delay from itself to the destination node v_{n+1} based on those of its next-hop along P. Denoting the mean and standard

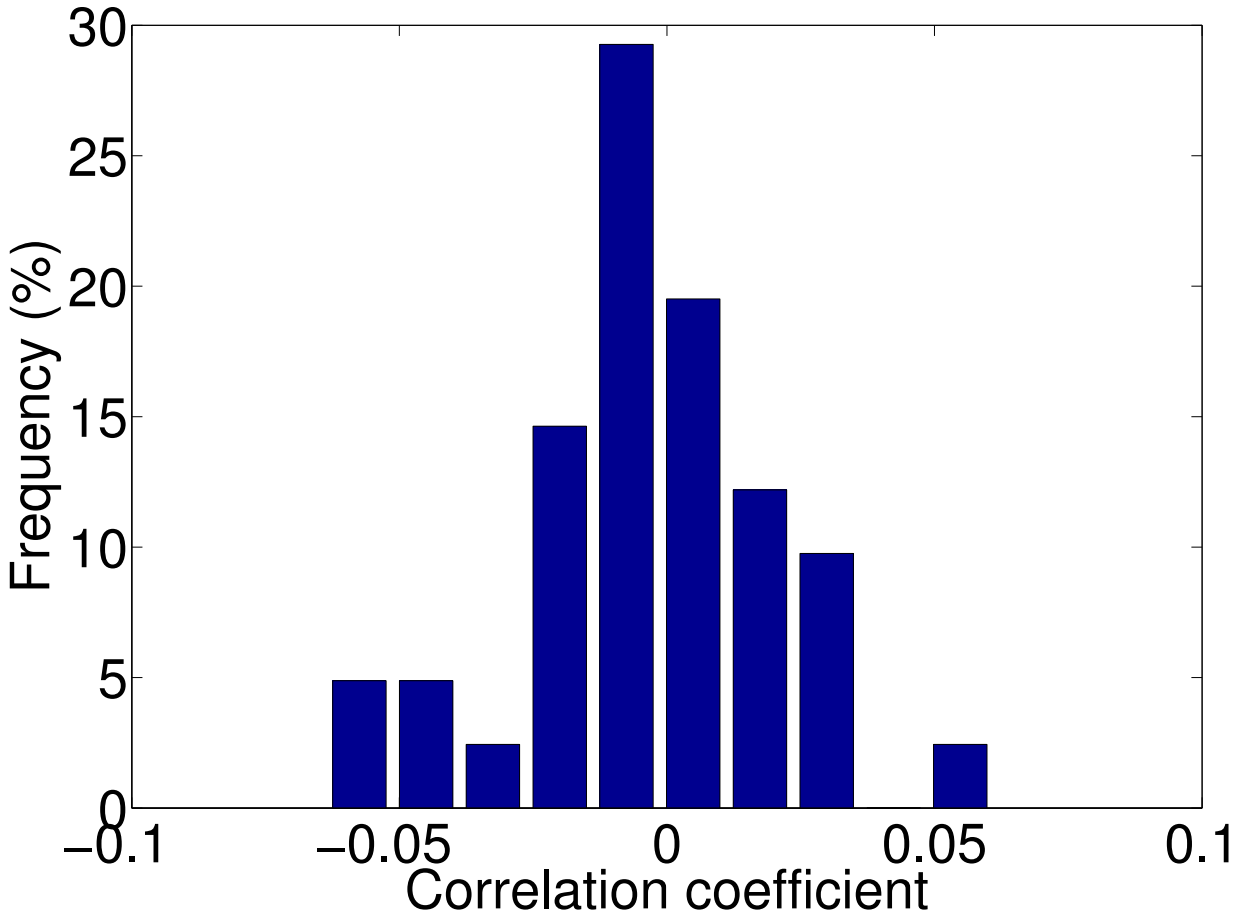

Figure 24: Histogram of the correlation coefficient between the packet-time across different outgoing links from the same node

deviation of the delay from v_i to v_{n+1} by $\mu(d_P^i(t))$ and $\sigma(d_P^i(t))$ respectively (i = 0 ... n), then we have

$$\begin{aligned} &\mu(d_P^i(t)) = \mu(d_P^{i+1}(t)) + \textstyle\sum_{k=1}^{N_i(t)} m_i^k(t)\mu(d_{i,k}(t)) \\ &\mu(d_P^{n+1}(t)) = 0 \end{aligned} \tag{5}$$

$$\begin{aligned} &\sigma(d_P^i(t)) = \sqrt{\sigma^2(d_P^{i+1}(t)) + \textstyle\sum_{k=1}^{N_i(t)} m_i^k(t)\sigma^2(d_{i,k}(t))} \\ &\sigma(d_P^{n+1}(t)) = 0 \end{aligned} \tag{6}$$

Formulae (7) and (8) can be implemented using a distance-vector-type routing algorithm. Due to information diffusion delay τ_i from v_{i+1} to v_i in routing, the implemented version of Formulae (7) and (8) are

$$\begin{aligned} &\mu(d_P^i(t)) = \mu(d_P^{i+1}(t-\tau_i)) + \textstyle\sum_{k=1}^{N_i(t)} m_i^k(t)\mu(d_{i,k}(t)) \\ &\mu(d_P^{n+1}(t)) = 0 \end{aligned} \tag{7}$$

$$\begin{aligned} &\sigma(d_P^i(t)) = \sqrt{\sigma^2(d_P^{i+1}(t-\tau_i)) + \textstyle\sum_{k=1}^{N_i(t)} m_i^k(t)\sigma^2(d_{i,k}(t))} \\ &\sigma(d_P^{n+1}(t)) = 0 \end{aligned} \tag{8}$$

For accurate estimation of $\mu(d_P^i(t))$ and $\sigma^2(d_P^i(t))$, we need to make τ_i as small as possible. This can be achieved by having v_{i+1} piggyback $\mu(d_P^{i+1}(t))$ and $\sigma^2(d_P^{i+1}(t))$ onto its data transmissions as well as control signaling (e.g., broadcast of routing beacons) so that v_i can overhear the values quickly. In WSC networks, sample data about physical behavior are usually generated periodically in a continuous manner, thus τ_i is at the same timescale of inter-packet arrival interval, which enables quick diffusion of path delay statistics.

We also observe that, even though network queueing varies significantly at a long timescale of hundreds of inter-packet intervals, it is much more stable at a short timescale of a few inter-packet intervals. In the NetEye medium traffic scenario and for the time lags of 1, 5, and 10 packet transmissions, for instance, Figures 25, 26, and 27 show the empirical cumulative distribution function (CDF) of link queueing level changes (i.e., changes in $m_i^k(.)$), node queueing level changes (i.e., changes in $m_i(.)$), and path queueing level changes (i.e., changes in $\sum_i m_i(.)$) respectively. We see that, at the timescale of a few inter-packet intervals, network queueing remains relatively stable,

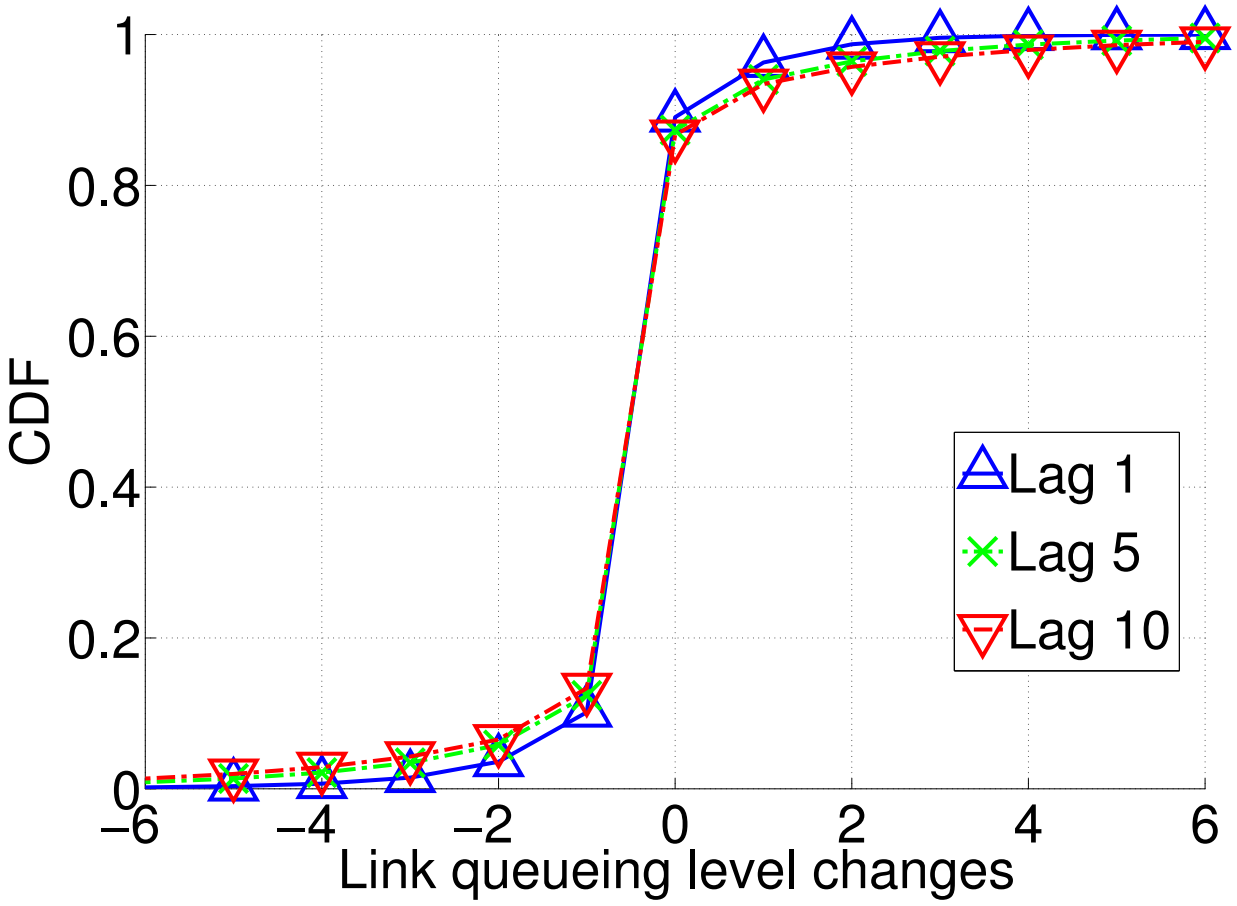

Figure 25: CDF of link queueing level changes

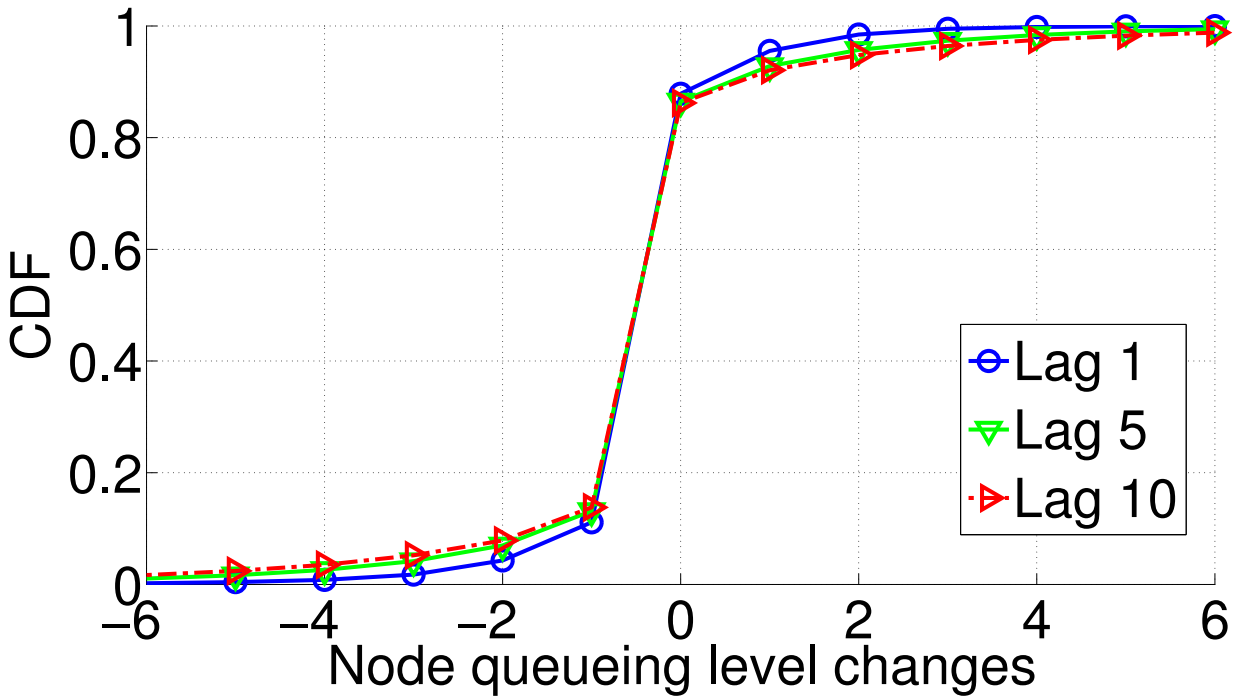

Figure 26: CDF of node queueing level changes

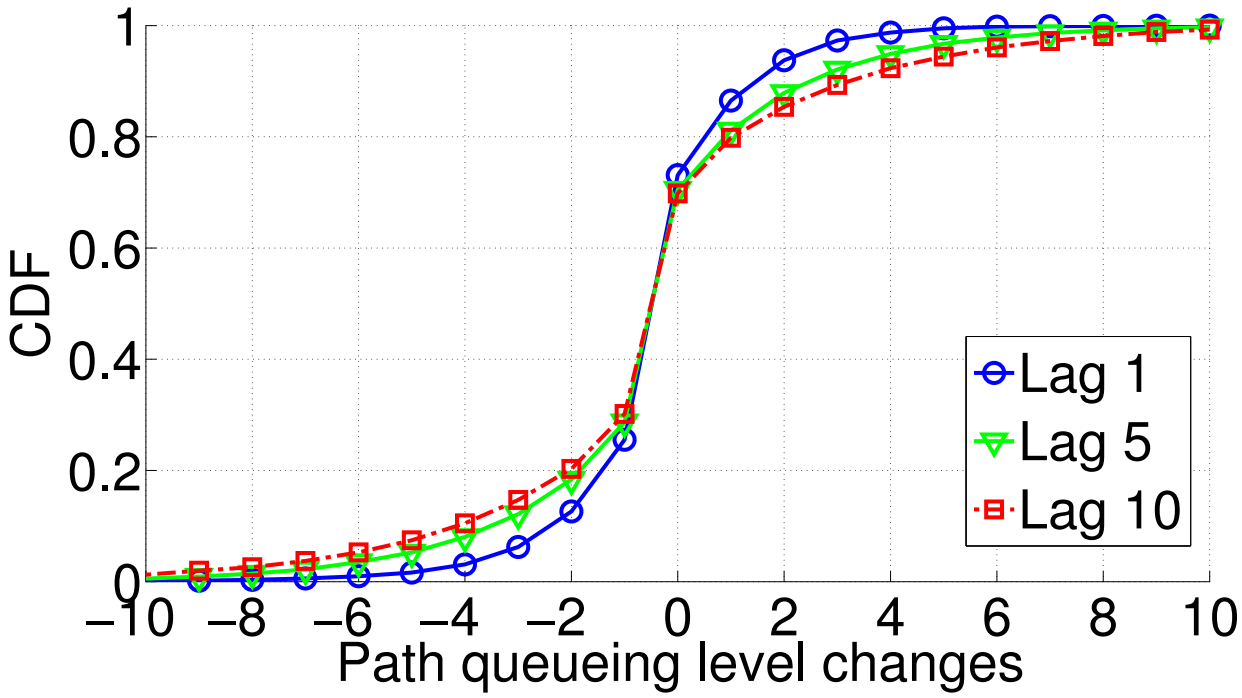

Figure 27: CDF of path queueing level changes

and, with more than 90% probability, the absolute changes in link, node, and path queueing levels are no more than 1, 1, and 3 respectively.[3] To ensure enough real-time capacity for each source node, routing hops tend to be limited (e.g., less than 10) in WSC networks [6]. This fact, together with the quick information diffusion and the relative stable network queueing at short timescales, enables the MTE method to accurately estimate the mean and variance of path delays in an agile manner.

Probabilistic path delay bound. With the mean and variance of path delay estimated via the MTE method, a node can derive the probabilistic path delay bounds using probability inequalities. To this end, we have

Proposition 1. *For a random variable X, if*

$$Pr\{X \geq f(x)\} \leq g(x), \tag{9}$$

then $Q_X^q = f(g^{-1}(1-q))$ is an upper bound on the q-quantile of X, where g^{-1} is the inverse function of g.

Proof. Let $g(x) = 1 - q$, then

$$Pr\{X \geq f(x)\} \leq 1 - q.$$

Thus,

$$Pr\{X \leq f(x)\} \geq q$$

Since $x = g^{-1}(1-q)$, we have

$$Pr\{X \leq f(g^{-1}(1-q))\} \geq q$$

Thus, $f(g^{-1}(1-q))$ is an upper bound on the q-quantile of X. □

For $\lambda > 0$ and a non-negative random variable X with mean μ and variance σ^2, two widely-

[3]The routing hop count in the NetEye medium traffic scenario with MTA is around 6 hops in general.

applicable probability inequalities [47] are the Markov Inequality

$$Pr\{X \geq \lambda\} \leq \frac{\mu(X)}{\lambda}, \tag{10}$$

and the one-tailed Chebyshev Inequality

$$Pr\{X - \mu \geq \lambda\sigma\} \leq \frac{1}{1+\lambda^2}. \tag{11}$$

Thus, we have

Corollary 1. *Using Markov Inequality,* $Q_X^q = \frac{\mu}{1-q}$.

Corollary 2. *Using one-tailed Chebyshev Inequality,* $Q_X^q = \mu + \sigma\sqrt{\frac{q}{1-q}}$.

For a typical five-hop path in NetEye, Figure 28 shows the ground truth and the estimated 90-percentile of the path delay when it is the sum of 5 and 40 packet-time random variables respectively. For comparison purpose, we present the probabilistic delay bounds estimated via the optimal-partition-minimum-delay (OPMD) method [38] or by assuming path delay is normally distributed [34]; we also present the maximum path delay. Given that Markov Inequality usually gives a looser bound than one-tailed Chebyshev Inequality, the probabilistic delay bound by Markov Inequality is greater than that by Chebyshev Inequality. Compared with the method of assuming path delay is normally distributed, the bound by Chebyshev Inequality is always greater than the actual 90-percentile path delay, whereas the former method underestimates the 90-percentile for the case of 40 packet-time random variables. Given that the OPMD method is rather conservative in estimating path delay bound, the bound by Chebyshev Inequality is also much less than the bound by the OPMD method, especially when path queueing increases. We also see that the maximum path delay is orders of magnitude greater than the bound by Chebyshev Inequality, thus using probabilistic delay bound instead of maximum path delay helps improve application-usable real-time capacity. Since the bound by Chebyshev Inequality upper-bounds and is close to the probabilistic

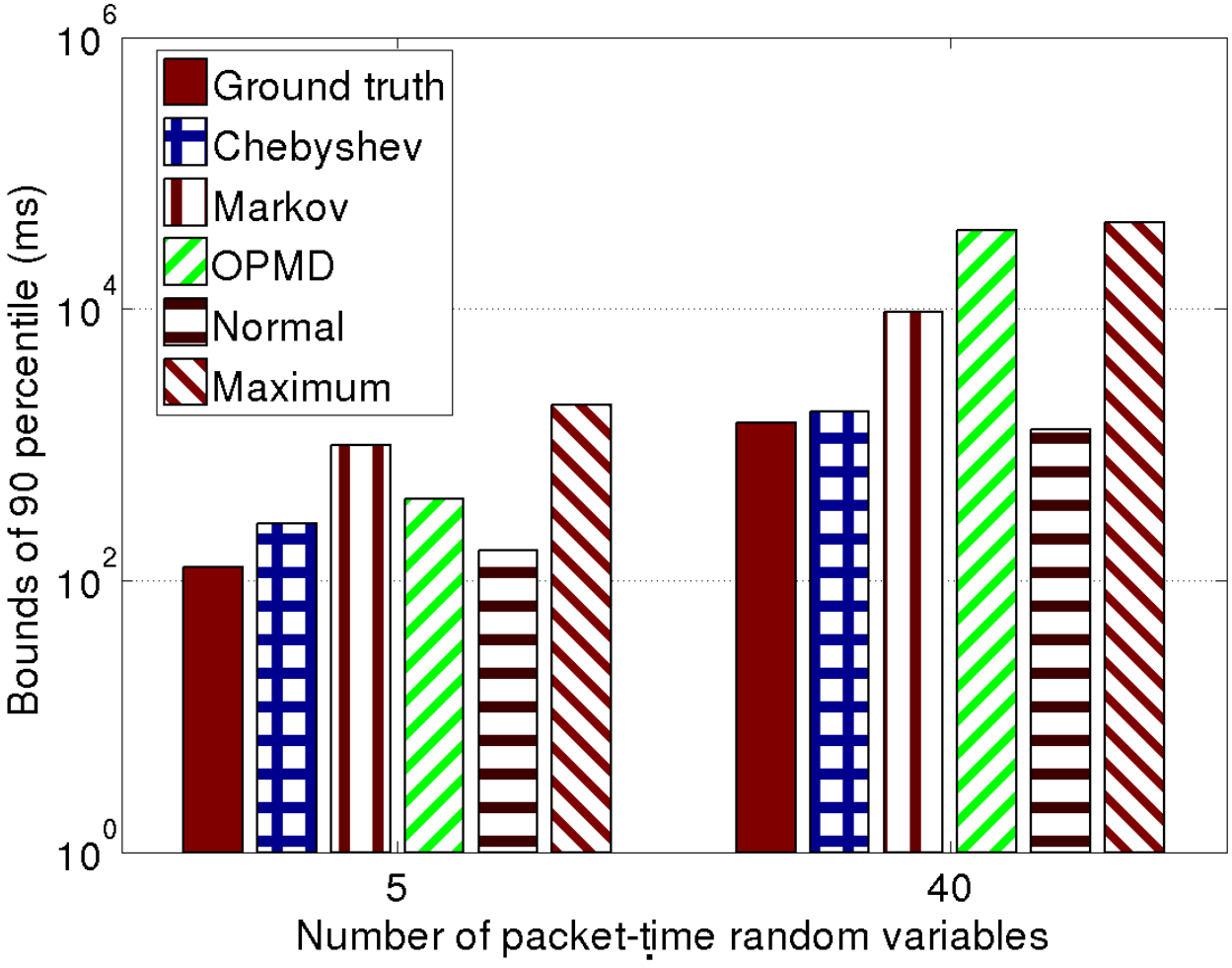

Figure 28: Bounds on 90-percentile of path delay

path delay, we use the probabilistic delay bound by Chebyshev Inequality in our protocol design.

From FCFS to EDF. In real-time scheduling, the earliest-deadline-first (EDF) algorithm is a commonly used algorithm, and it can ensure a smaller deadline than what is feasible with the first-come-first-serve (FCFS) algorithm. Our MTE method of computing probabilistic path delay bounds is derived by assuming that nodes use FCFS algorithm in intra-node transmission scheduling, but we observe that the delay bounds derived via our FCFS-based MTE method can also serve as the basis of selecting real-time packet forwarding paths even if the EDF algorithm is used. Formally,

Proposition 2. *If routed along a path whose probabilistic delay bound computed via the FCFS-based MTE method is less than the relative deadline of a packet, the packet will reach its destination before its deadline even if the EDF algorithm is used for intra-node transmission scheduling.*

Proof. We prove the proposition by contradiction. When a packet pkt_0 is forwarded where, each time a next-hop has to be chosen, the next-hop is chosen such that pkt_0's relative deadline is no less than the probabilistic path delay bound computed via the FCFS-based MTE method, the only way in which pkt_0 can miss its deadline is as follows: when pkt_0 is queued at some node v_i, another packet pkt_1 arrives at v_i at time t and is transmitted to a next-hop earlier than pkt_0 is transmitted. In this case, since pkt_1 arrives at v_i later than pkt_0, the relative deadline of pkt_1 at time t is no less than the probabilistic path delay bound $Q_i(t)$ from v_i to the destination assuming pkt_1 is transmitted after pkt_0. If the EDF algorithm schedules pkt_1 to be transmitted earlier than pkt_0, this implies that the relative deadline of pkt_0 at time t is greater than that of pkt_1 at t, thus being greater than the delay bound $Q_i(t)$. Note also that $Q_i(t)$ has included the time taken to transmit pkt_1 and pkt_0. Therefore, even if pkt_1 is transmitted earlier than pkt_0 (which is similar to switching their queue positions in a FIFO queue), pkt_0 can still be delivered to the destination before its deadline. Q.E.D. □

4 Multi-timescale adaptation for real-time routing

In what follows, we first present the multi-timescale adaptation (MTA) framework for real-time routing, then we address the challenges of implementing MTA in resource-constrained platforms.

4.1 MTA routing design

Overview. Real-time routing is subject to dynamics and uncertainties at multiple timescales. At a longer timescale, link properties such as ETX (i.e., expected-number-transmissions to successfully deliver a packet) vary as a result of changing environmental conditions (e.g., temperature); at a shorter timescale, data transmission delay varies on a per-packet basis, and bursty traffic may introduce sudden changes to network conditions. Robust system design usually requires adaptation to dynamics at the same or shorter timescales of the dynamics themselves. Yet we have found that, due to the highly-varying nature of path delays, routing using delay-based metrics can introduce large estimation errors and lead to routing instability as well as low performance [57]. To ensure long-term stability and optimality while addressing short-term dynamics at the same time, we propose a multi-timescale adaptation (MTA) framework for real-time routing as follows.

At lower frequencies, a directed-acyclic-graph (DAG) is maintained for data forwarding, and any path within the DAG is a candidate path for packet delivery. Given that link/path ETX reflects network throughput, data delivery reliability, and the overall trend of data delivery delay [15, 17], and that ETX-based routing structures tend to be stable even if ETX is dynamic [56], we propose to maintain the data forwarding DAG based on link and path ETX such that the DAG reflects long-term system optimality and changes relatively slowly compared with delay variation. More specifically, there is a directed edge from node v_i to v_i' in the DAG if and only if the minimum path ETX from v_i' to the destination v_0' is less than that from v_i to v_0'. The DAG defines, for each node v_i, a set of forwarder candidates $\mathcal{R}(v_i)$ where $v_i' \in \mathcal{R}(v_i)$ if and only if link $\langle v_i, v_i' \rangle$ belongs to the DAG.

At higher frequencies, the spatiotemporal flow of packets within the data forwarding DAG is

adaptively controlled to ensure reliable, real-time data delivery in the presence of short-timescale dynamics such as transient packet losses and per-packet variations of link delay. More specifically, each packet contains information about the remaining time to deadline, denoted by L, and the required real-time guarantee probability q. When the packet reaches a node v_i, v_i first finds the set of forwarder candidates within the DAG, denoted by $\mathcal{R}'(v_i, L, q)$, that can ensure the real-time requirements L and q; then v_i sets the next-hop node of the packet as the node of the smallest path ETX to v_0' among all the nodes in $\mathcal{R}'(v_i, L, q)$, and v_i puts the packet in the transmission queue. Queued packets are then scheduled for transmission using the earliest-deadline-first (EDF) algorithm.

Using the above approach, packets are always routed along the minimum-ETX path that satisfies the real-time data delivery requirement. In the presence of heavy traffic load that induces queueing, this real-time forwarding mechanism creates the water-filling effect as follows: packets are delivered to the minimum-ETX path until the path cannot ensure the required timeliness of data delivery (e.g., due to queueing), at which point the forthcoming packets are delivered to the path with the second-minimum-ETX, and so on; once the delay along paths of less ETX decreases (e.g., due to reduced queueing), more packets will be delivered to those paths to fill them up. This process repeats at the same timescale of packet arrival process, and it enables real-time data delivery while ensuring as small ETX in data delivery as possible. Unlike traditional delay-based routing that may lead to instability and low performance, this quick adaptation of spatial packet flow is enabled by 1) the MTE method of accurate, agile estimation of probabilistic path delay bound at the same timescale of the changes in path delay distribution and by 2) the overall stability of spatial packet flow along the data forwarding DAG.

System architecture. To implement the above MTA framework, we adopt a system architecture as shown in Figure 29.

Using MAC feedback for the status (e.g., number of retries and time duration) of transmitting data packets, each node uses data-driven approaches [24, 55] to estimate the ETX as well as the mean and variance of packet-time for its local links. Based on nodes' local link ETX, the data-

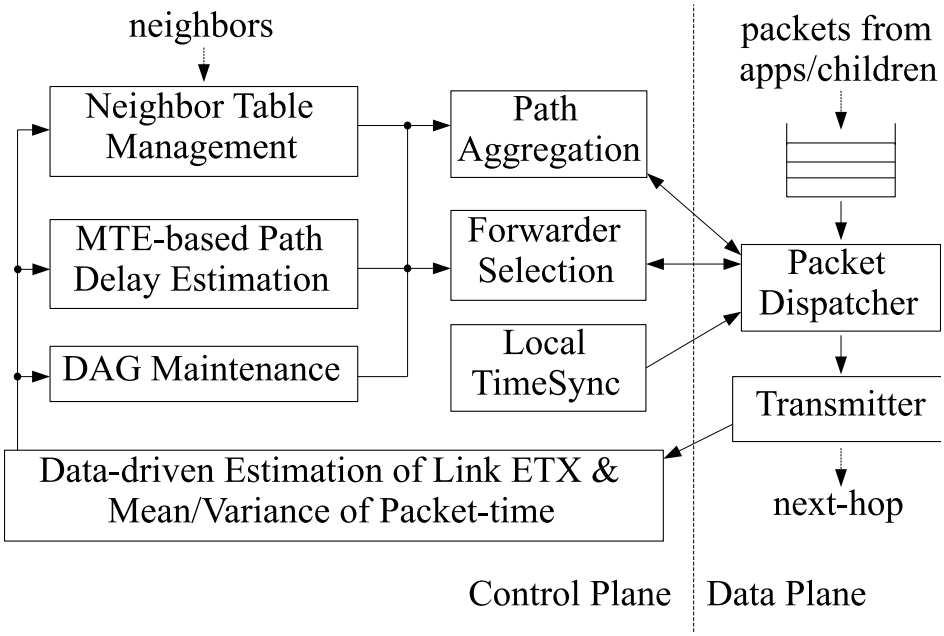

Figure 29: Architecture of MTA-based real-time routing

forwarding DAG can be established and maintained using distance-vector-routing-type diffusion computation.

Each node also maintains a neighbor table that stores information about its neighbors. In the case of limited memory, a node may only maintain information about a selected set of neighbors, e.g., those of small path ETX to the destination. For each neighbor in the neighbor table, the ETX as well as the mean and variance of the delay along different paths from the neighbor to the destination is maintained. Ideally, each node shall share information about all the paths from itself to the destination, but this is infeasible in practice due to limited memory size at nodes, overhead of exchanging path information with neighbors, and the exponentially increasing number of paths from a node to the destination as the hop distance to the destination increases. Therefore, each node v_i only shares with its neighbors a summary of its knowledge about the paths from itself to the destination, using the following path aggregation method:

1. For each path P from a neighbor to the destination v_0', as implicitly identified by an entry in the neighbor table, v_i computes the mean and variance of the delay from itself to v_0' through P using Formulae (7) and (8), and, based on Corollary 2 and application-required real-time guarantee probability q, compute the probabilistic delay bound for the path $\{v_i, P\}$ from v_i to v_0' through P;

2. Compute the set $\mathcal{P}$ of all valid paths from v_i to v_0', where a path $\{v_i, P\}$ is valid if there is no other path $\{v_i, P'\}$ that has both smaller ETX and smaller path delay bound (as computed above);

3. Order the paths in $\mathcal{P}$ in a non-decreasing order of the delay bounds they ensure, and select a subset of them as the path summary to share with neighbors; in our implementation, a node selects the path summary such that the delay bounds ensured by the selected paths cover the complete range of delay bounds ensured by the original set $\mathcal{P}$.

The computed path summary is piggybacked onto data packet transmissions by v_i, where each entry of the summary records the ETX as well as the mean and variance of the delay along the

corresponding path.

When a node receives a packet from a local application or a child, the packet dispatcher first uses a local time synchronization service [5] to compute the time elapsed since the packet was transmitted by its sender, which enables the dispatcher to derive the remaining time to deadline (a.k.a. relative deadline) for the packet. Based on the packet's relative deadline, the dispatcher puts the packet in a corresponding position in the local queue. Whenever the queue is not empty and the wireless channel becomes available, the dispatcher selects a packet to transmit using the EDF algorithm (a.k.a. least laxity first algorithm).

4.2 Implementation

In this section , we discuss challenges and our solutions in implementing MTA architecture (as shown in Figure 29) in TinyOS.

Computation overhead & task management. Unlike wired Internet routers, embedded WSC network nodes tend to be resource-constrained, yet the many control plane and data plane functions (e.g., link estimation, next-hop selection) need to be computed in real-time and on a per-packet basis to ensure accuracy and agility in estimation and routing. Therefore, the different computation tasks inside each node need to be managed well to ensure correct operation of the protocol. In fact, the computation tasks can consume so much CPU time that the entire protocol collapses in TelosB due to its failure of executing critical tasks. For instance, timers may fail to fire at specified time instants, and packet receptions may not be signaled to upper layers and thus lost. To address these issues, we split heavy-weight computations (e.g., computing the next-hop for a packet) into multiple subtasks such that each computation can be interrupted between its subtasks to ensure responsiveness of the system to events such as timer firing and packet reception. Our experience suggests that it suffices to have each subtask take no more than a few milliseconds of time to complete.

Be default, TinyOS employs a FIFO queue to store all pending tasks. It does not differentiate urgent tasks (i.e., data packet transmission) from normal tasks (i.e., beacon transmission) in terms

of real-time communication. To address this issue, we have implemented priority schemes for managing the tasks of the control and data planes to ensure real-time packet forwarding and control adaptation.

Global vs. local time synchronization. Global time synchronization is sufficient to provide a time reference across the entire network. Nonetheless, it comes at substantial cost of memory, computation, and bandwidth overhead, leaving even less resource for MTA in already resource-constrained devices. Even worse, it can take a long time for a global time synchronization protocol such as FTSP [41] to converge, during which the network cannot be used to deliver any real-time traffic.

Upon closer scrutiny, we discover that only neighboring nodes, not all nodes, have to agree on a common time as far as MTA/MTE is concerned, which enables estimating packet-time and tracking absolute/relative deadline when a packet is delivered from one node to another. Thus, we employ the packet-level time synchronization mechanism in TinyOS, which automatically converts a time value from the sender's local time to the receiver's local time. Only a few bytes are added into the header a data packet, and no additional control packets have to be sent, significantly reducing the overhead compared with global time synchronization.

Packet-level time synchronization requires timestamping packet reception. But the default radio stack for TelosB stores the incoming packet and its reception timestamp in two separate FIFO queues and then match them by taking head-of-queue elements from both queues. Occasional mismatches of a packet and its timestamp are observed due to disparate operations on the two queues. To address this issue, we store the reception timestamp of a received packet directly in its metadata instead of using separate FIFO queues.

5 Measurement evaluation

We evaluate the behavior of MTA through measurement study in the NetEye and Indriya testbeds. In what follows, we first present the methodology, then we study the design decisions of MTA and compare MTA with existing routing protocols.

5.1 Methodology

Protocols. We have implemented the MTA framework (including the MTE method) in TinyOS. To understand the design decisions of MTA, we comparatively study MTA with its following variants:

- *M-DS*: same as MTA but directly estimate path delay quantiles using non-parametric method P^2 [31] and path delay samples that are collected in a distributed manner;
- *M-DB*: same as MTA but estimates the mean and variance of path delay directly through path delay samples;
- *M-ST*: same as MTA but estimates the mean and variance of path delay as the sum of the mean and variance of the sojourn time at each node of the path, without decomposing the sojourn time into the individual packet-times;
- *M-MD*: Same as MTA but maintains the data forwarding DAG based on mean link/path delay instead of link/path ETX;
- *M-mDQ*: same as MTA but forwards packets to the next-hop candidate with the minimum path delay quantile instead of the one with minimum path ETX;
- *mDQ*: same as M-mDQ but does not use the data forwarding DAG of MTA for stability control;
- *M-FCFS*: same as MTA but uses FCFS instead of EDF for intra-node transmission scheduling.

Towards understanding the benefits of MTA, we also comparatively study MTA with the following existing protocols:

- *MCMP*: a multi-path QoS routing protocol where end-to-end QoS requirements on reliability and timeliness are uniformly divided into per-hop reliability and timeliness requirements, upon which a node chooses the minimum number of next-hops to satisfy the per-hop requirements in data delivery [29];

- *MM*: the geographic routing protocol MMSPEED [22] that routes and schedules packet transmissions based on nodes' distances to destinations, packet delivery deadlines, and mean link delays; MMSPEED also tries to improve packet delivery reliability by transmitting packets along multiple paths; (Note: we denote MMSPEED as MM for the readability of figures to be presented in the next section .)

- *MM-CD*: same as MMSPEED but, instead of using the mean link delay, uses a conservative estimate of link delay that equals the sum of the mean delay and three times the standard deviation of the delay;

- *SDRCS*: similar to MMSPEED but, instead of using geographic distance, uses data-forwarding hop-count as the measure of distance, where the hop-count is computed based on received-signal-strength (RSS) between nodes [51]; data forwarding is through receiver contention similar to that in opportunistic routing;

- *CTP* : an ETX-based non-real-time routing protocol in TinyOS [24].

(Note: for MM and MM-CD which use node locations, we configured each node with its correct location in our TinyOS programs.)

Traffic & real-time requirements. For studying the design decisions of MTA, we use the common case of the medium traffic scenario in NetEye; for comparing MTA with existing real-time routing protocols, we use all the periodic and event traffic scenarios discussed in Chapter 2. In our experiments, we use 90% as the required real-time guarantee probability by default, but

we have also experimented with the real-time guarantee probability of 99% and observed similar phenomena [39]. For differentiating the performance of different protocols, the deadline for each traffic scenario is chosen so that it is neither too stringent (that no protocol can support) nor too loose (that all protocols can support). In NetEye, the deadlines for light, medium, and heavy traffic are 250ms, 2 seconds, and 7.5 seconds respectively, and the deadline for event traffic is 10 seconds. In Indriya, the deadlines for periodic and event traffic are 2 seconds and 2.5 seconds respectively.

Metrics. For each combination of protocol, testbed, and traffic scenario, we run it for 10 times and evaluate protocol performance in terms of the following metrics:

- *Deadline success ratio (DSR)*: ratio of packets delivered to the sink before their deadlines;
- *Packet delivery ratio (PDR)*: ratio of packets delivered to the sink;
- *Number of transmissions per packet delivered (NTX)*: total number of transmissions, including retransmissions, divided by the number of unique packets delivered to the sink.

To understand protocol behavior in more detail, we also analyze the different causes for a packet to miss its deadline:

- *Overflow*: packet discarded due to node queue overflow;
- *Transmission failure*: a packet not delivered to the next hop even after the maximum number of retransmissions at a node;
- *Rejection*: no candidate path can ensure the required real-time delivery guarantee (i.e., deadline and probability) of a packet;
- *Expiration*: deadline expired before the packet reaches the sink, whether or not the packet is delivered to the sink.

5.2 Design decisions of MTA

For periodic medium traffic in NetEye and for protocol MTA and its variants, Figures 30, 31, 32, and 33 show the deadline success ratio, packet delivery ratio, number of transmissions per packet delivered, and packet delivery status respectively. We see that MTA consistently enables the

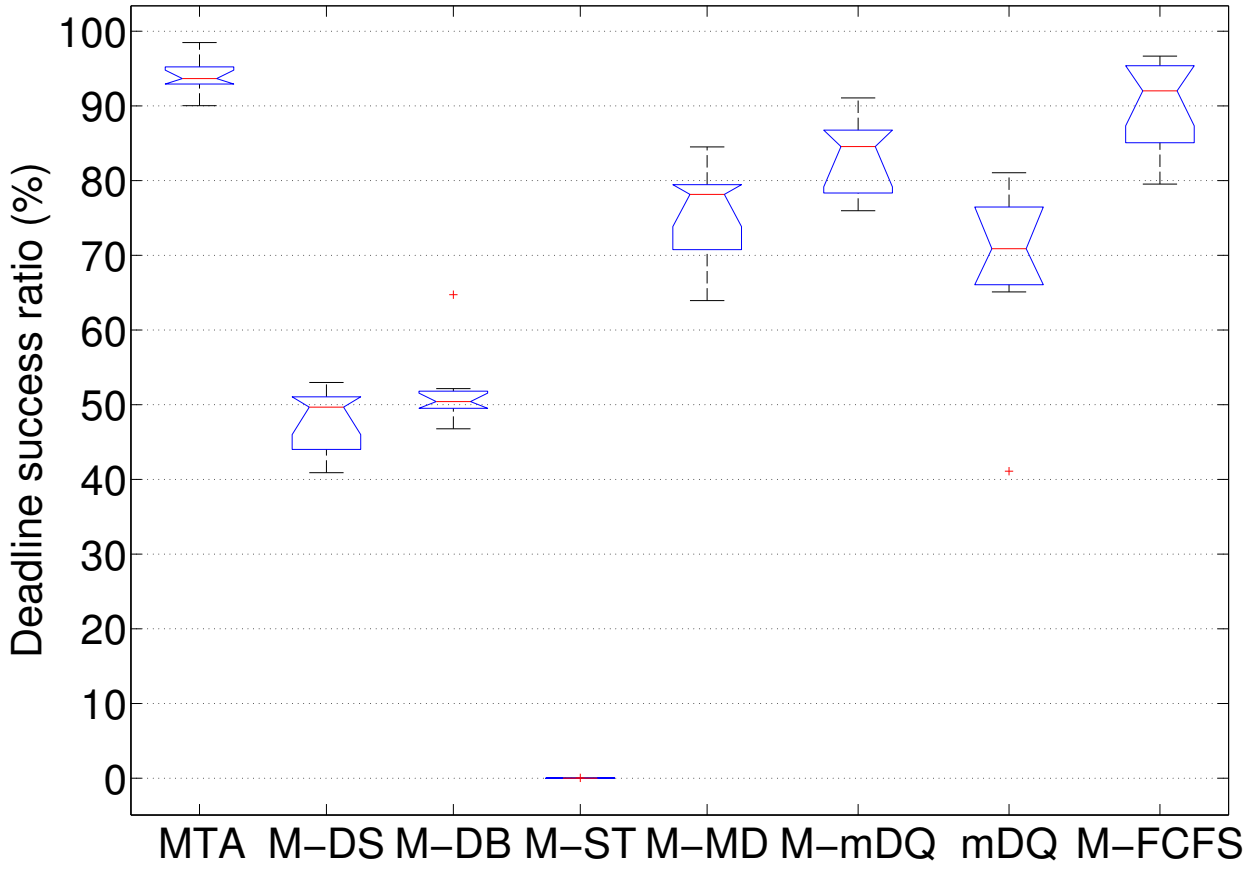

Figure 30: Deadline success ratio: MTA and variants

highest deadline success ratio and packet delivery ratio as well as the lowest transmission cost.

Compared with MTE as used in MTA, M-DS, M-DB, and M-ST all underestimate path delay quantiles such that packets are routed along paths that cannot ensure the required data delivery deadline, which makes packets rejected in the network or their deadlines expire before reaching the sink node (as shown in Figure 33). For a typical four-hop path in NetEye, for instance, Figure 34 shows the empirical cumulative distribution function (CDF) of the relative error in estimating the 90 percentile of path delay using MTA, M-DS, M-DB, and M-ST respectively, where the relative

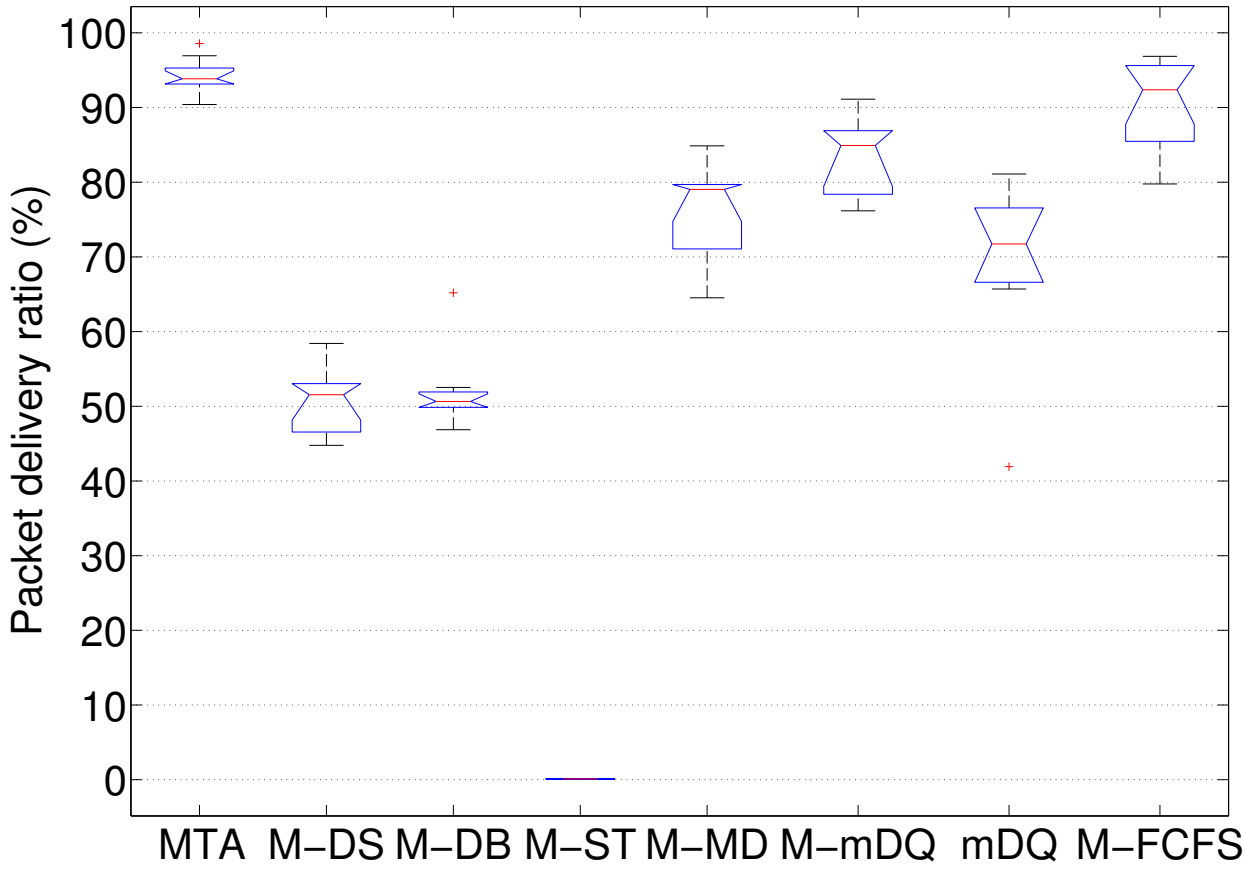

Figure 31: Packet delivery ratio: MTA and variants

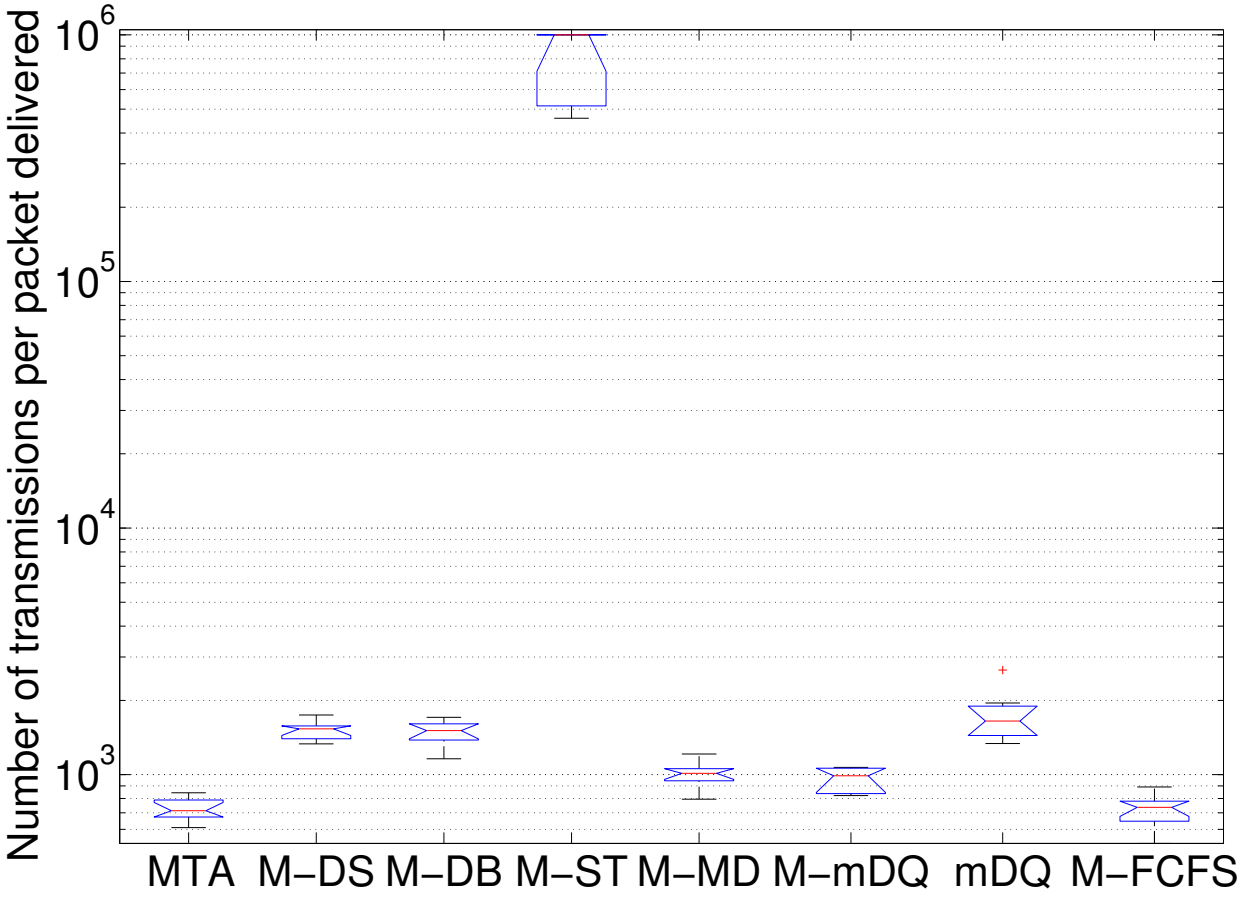

Figure 32: Number of transmissions per packet delivered: MTA and variants

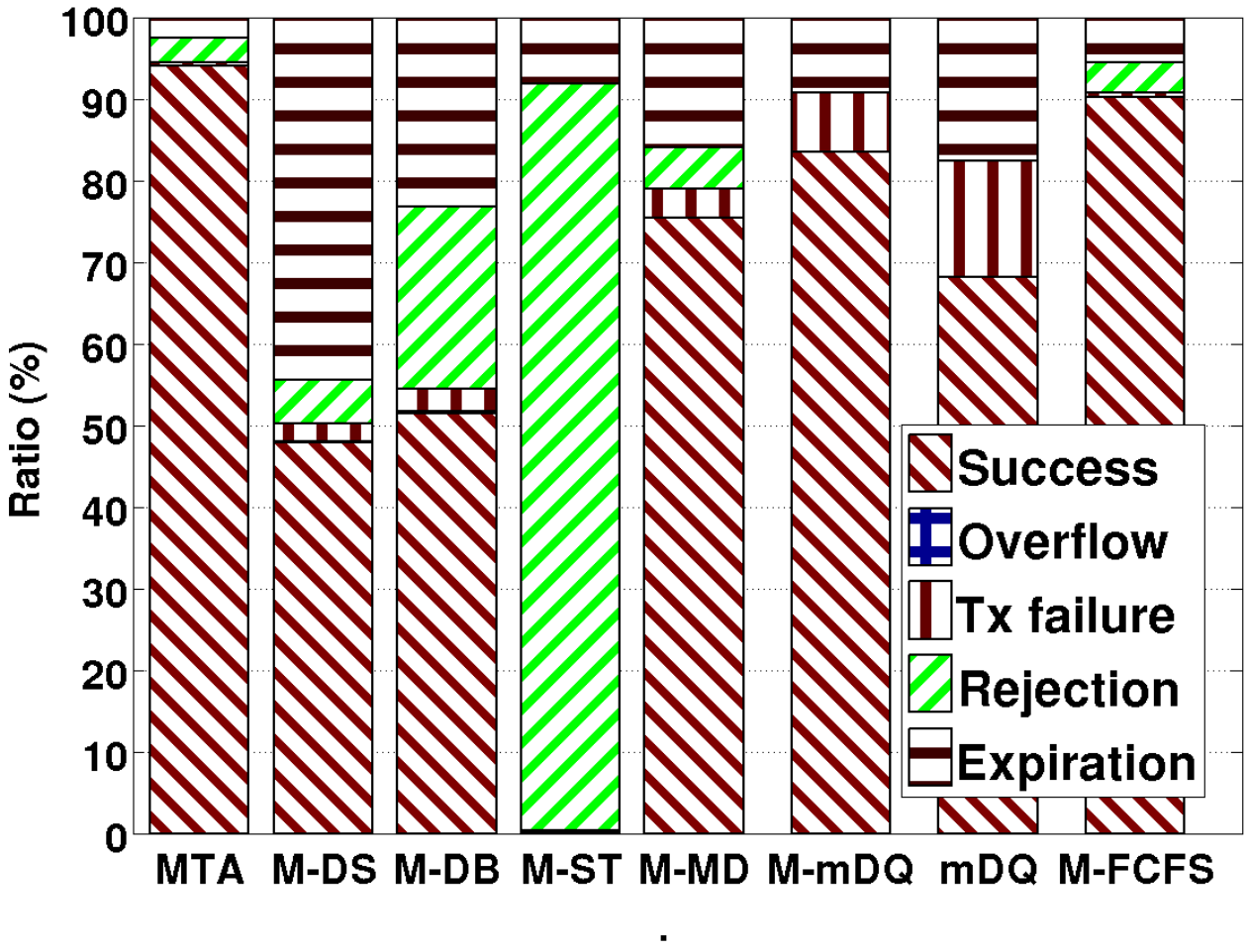

Figure 33: Packet delivery status: MTA and variants

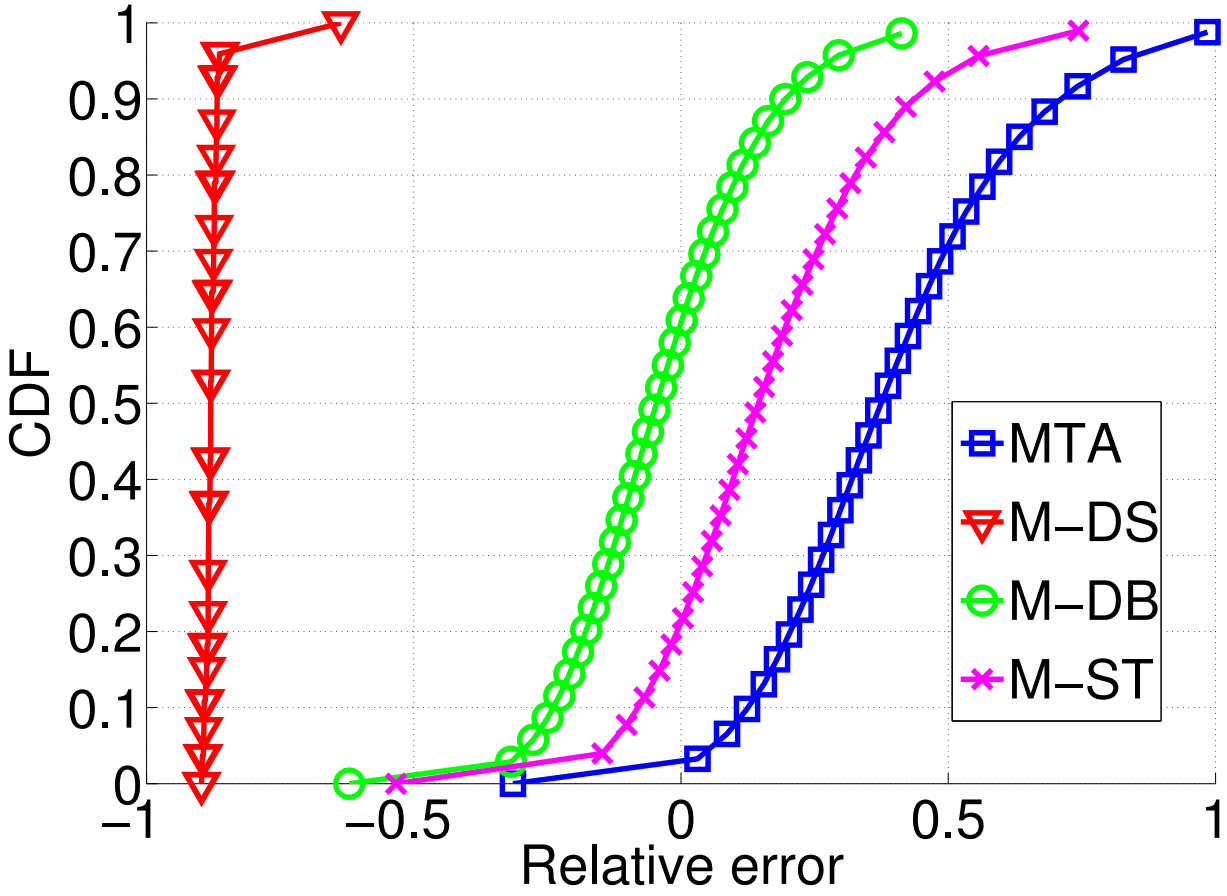

Figure 34: CDF of the relative error in estimating the 90 percentile of path delay

error is defined as the estimated percentile minus the actual percentile and then divided by the actual percentile. We see that M-DS, M-DB, and M-ST underestimate the 90 percentile for 100%, 60%, and 20% of the time respectively. M-DS and M-DB tend to underestimate because, when sampling path delay in a distributed manner, a sample of the delay along a link can be used multiple times due to diffusion delay; this tends to reduce the variability of the collected path delay samples, thus causing the underestimation of path delay variance which in turn leads to the underestimation of delay quantiles. M-ST tends to underestimate because the sojourn time at different nodes are positively correlated due to positive correlation in packet queueing at nodes; since the variance of the sum of the sojourn time at different nodes of a path equals to the sum of their individual variance plus their pair-wise covariance, path delay variance tends to be underestimated in M-ST, which in turn leads to the underestimation of path delay quantiles. In contrast, MTA only underestimates for 2% of time thanks to the MTE method. The reason why MTA does not completely avoid underestimation is because, just as in any possible estimation method, there are inherent errors (e.g., due to the EWMA estimator) in the MTE method that leads to small errors in estimating the mean and standard deviation of path delay which in turn can offset the conservativeness of Chebyshev inequality.

It is known that ETX-based routing enables higher delivery ratio and lower transmission cost than delay-based routing [57], thus MTA enables higher delivery ratio and lower transmission cost than M-MD, M-mDQ and mDQ. M-mDQ and mDQ use paths with minimum path delay quantiles; this introduces more route changes than in MTA, leading to larger errors in estimating path delay quantiles and thus reduces deadline success ratio. Even though M-MD, M-mDQ, and mDQ do not try to minimize path ETX as M-DS, M-DB, and M-ST do, they outperform M-DS, M-DB, and M-ST which do not use MTE as the path delay estimation method; this shows the importance of accurate, agile estimation of path delay quantiles via MTE.

The median deadline success ratio in M-FCFS is higher than the required real-time guarantee probability of 90%, but there are a few cases when the deadline success ratio is slightly below 90%. This is because, just as any estimation method, the MTE method is not perfect; it has to use

estimators such as exponentially-weighted-moving-average which has inherent estimation errors, and the delay in diffusing network state (e.g., mean and variance of path delay from a node to the sink) introduces slight estimation errors in MTE. The fact that MTA always ensures a deadline success ratio of at least 90% shows that the temporal packet flow control enabled by EDF helps ameliorate the impact of the minor imperfections in MTE. Thus an interesting direction to explore is the joint optimization of spatial and temporal packet flow in real-time networking, and we will study this in more detail in our future work.

5.3 Different routing protocols

NetEye. For periodic medium traffic in NetEye, Figures 35, 36, 37, and 38 show the deadline success ratio, packet delivery ratio, transmission cost, and packet delivery status respectively.

We see that MTA always ensures the required real-time delivery performance, and the median deadline success ratio in MTA is 76%, 89%, 86%, 11%, and 38% higher than that in MCMP, MM, MM-CD, SDRCS, CTP respectively. The median number of transmissions per packet delivered in MTA is less than that in MCMP, MM, MM-CD, SDRCS, CTP by a factor of 3.7, 9.7, 6.5, 1.2, and 1.2 respectively.

One reason why the deadline success ratio in MCMP, MM, and MM-CD is very low is because a large fraction of packets are lost due to queue overflow as can be seen in Figure 38. They all try to use multiple paths to ensure data delivery reliability: at the sources, multiple copies of a packet can be sent; at the next hop, each of these copies can be multiplied again, and so on. This multipath routing mechanism can lead to exponentially increasing number of copies of a packet, thus causing severe queue overflow and large transmission cost as shown in Figures 38 and 37 respectively. SDRCS does not have this problem because it does not use multipath routing, instead it uses a data forwarding mechanism similar to opportunistic routing.

Another reason why MCMP, MM, MM-CD, and SDRCS do not perform well is because, by evenly dividing end-to-end QoS requirements into per-hop requirements, they implicitly assume that network conditions are uniform across the network which is usually not the case. Among these

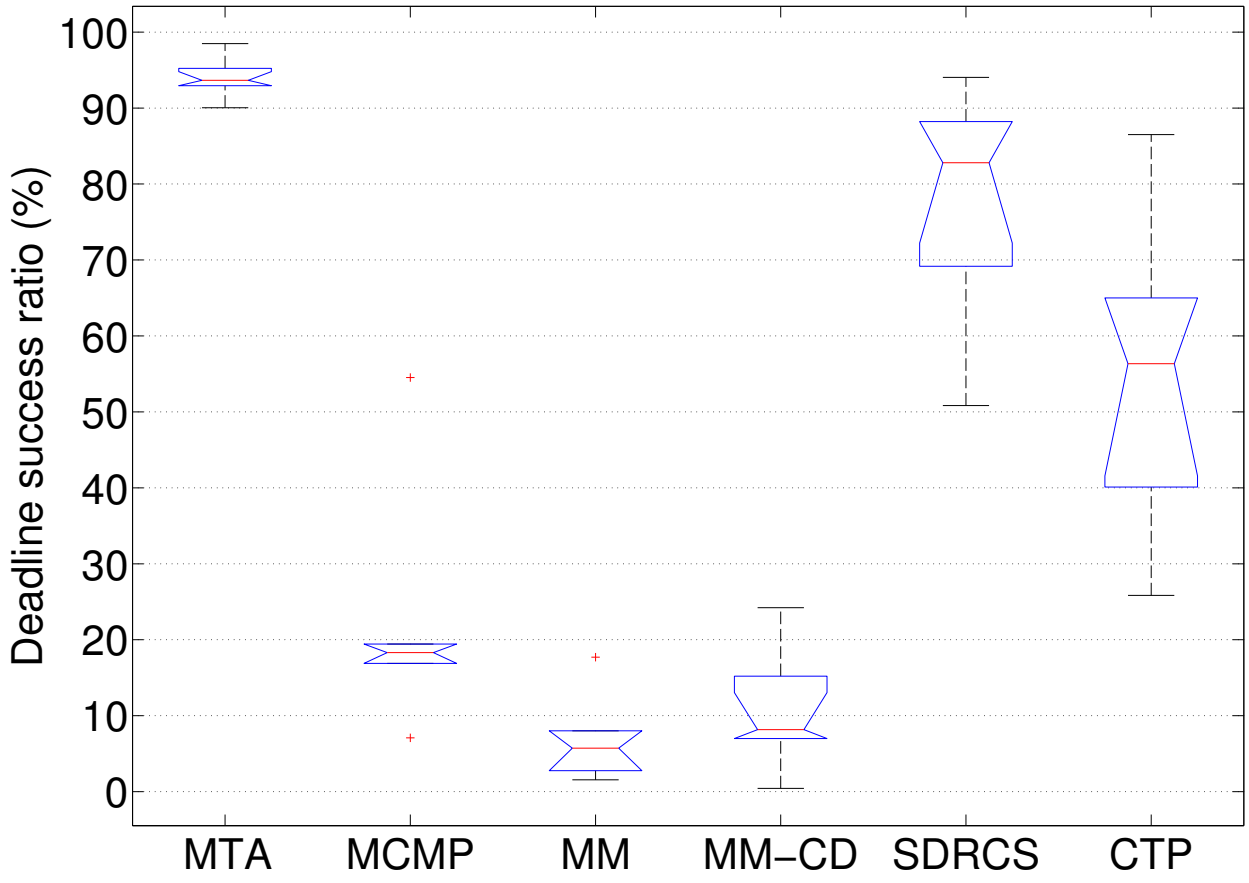

Figure 35: Deadline success ratio: MTA and existing protocols in the NetEye medium traffic scenario

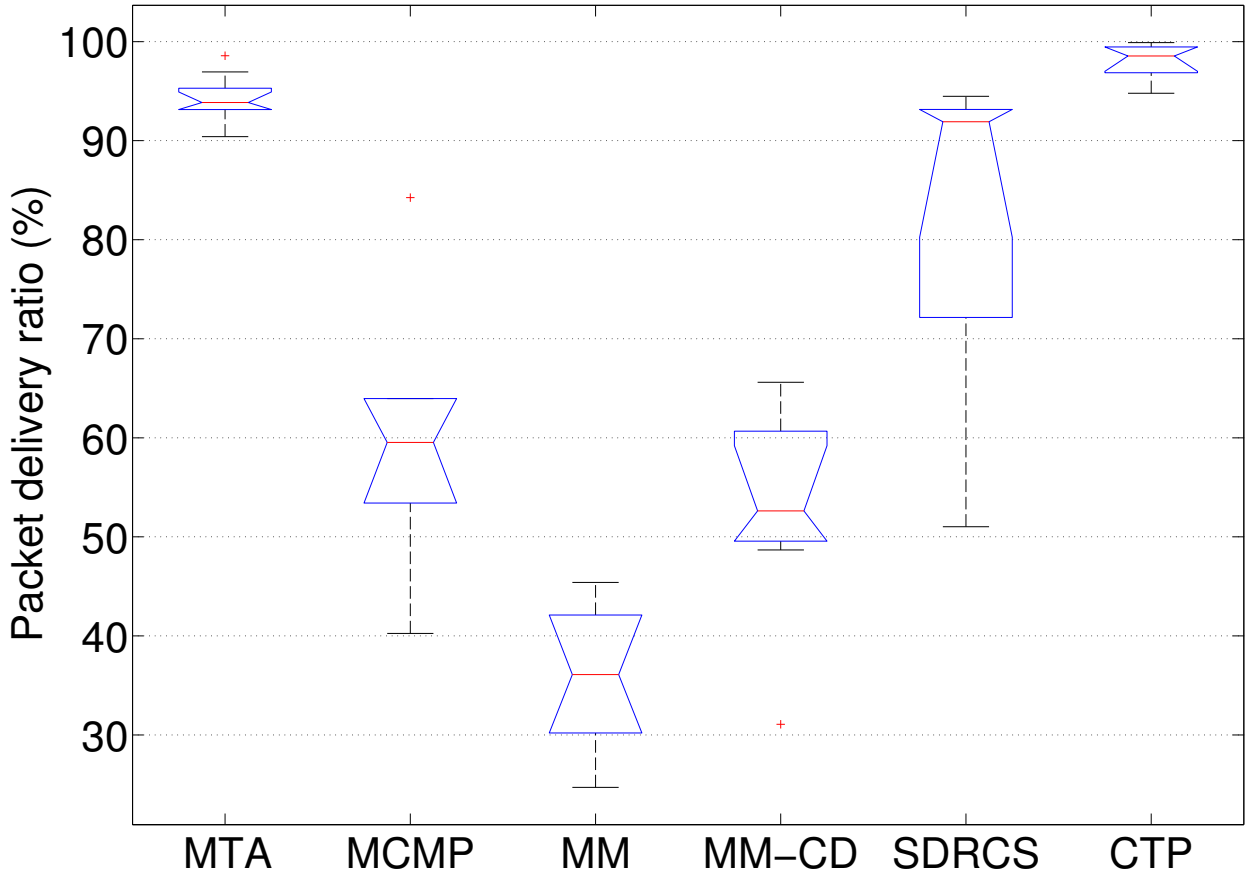

Figure 36: Packet delivery ratio: MTA and existing protocols in the NetEye medium traffic scenario

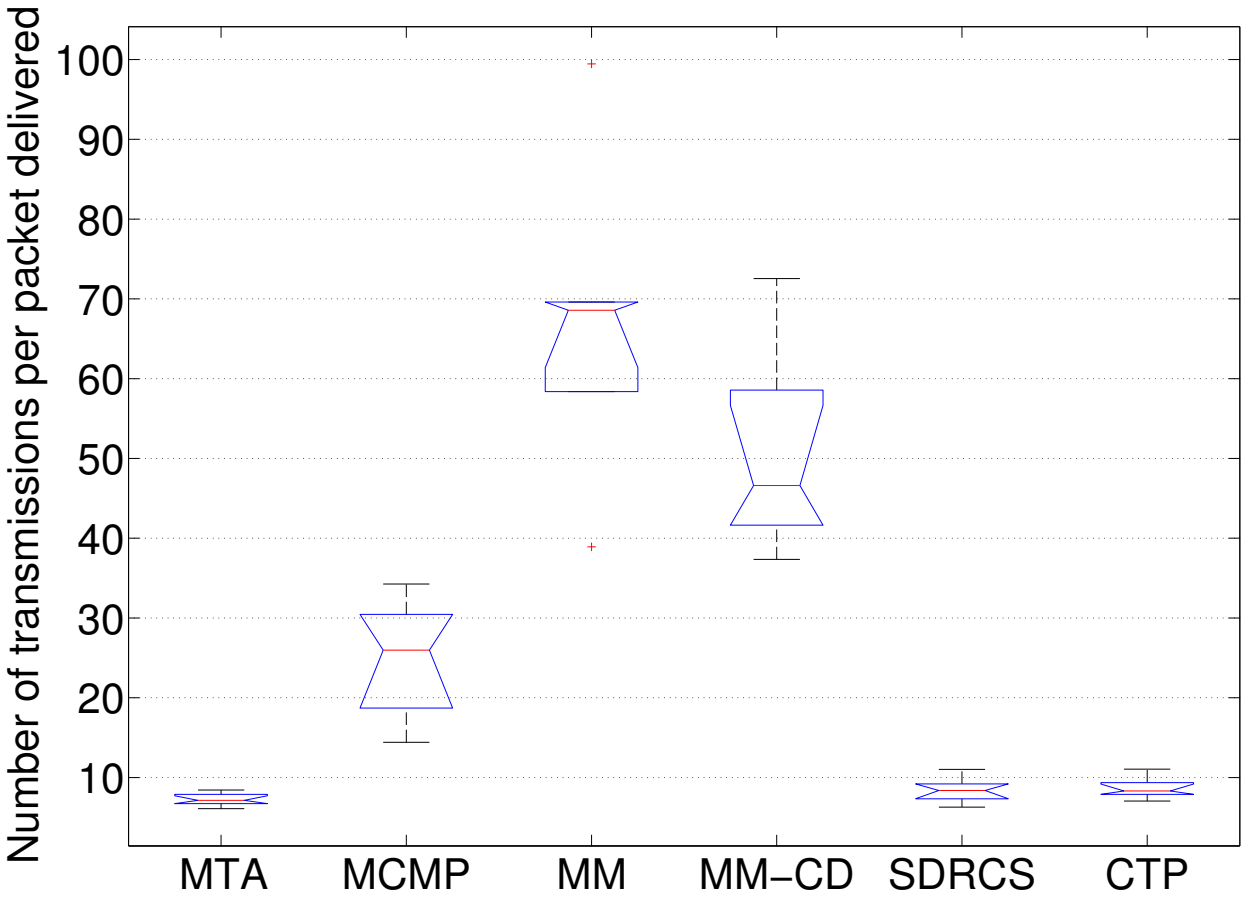

Figure 37: Number of transmissions per packet delivered: MTA and existing protocols in the NetEye medium traffic scenario

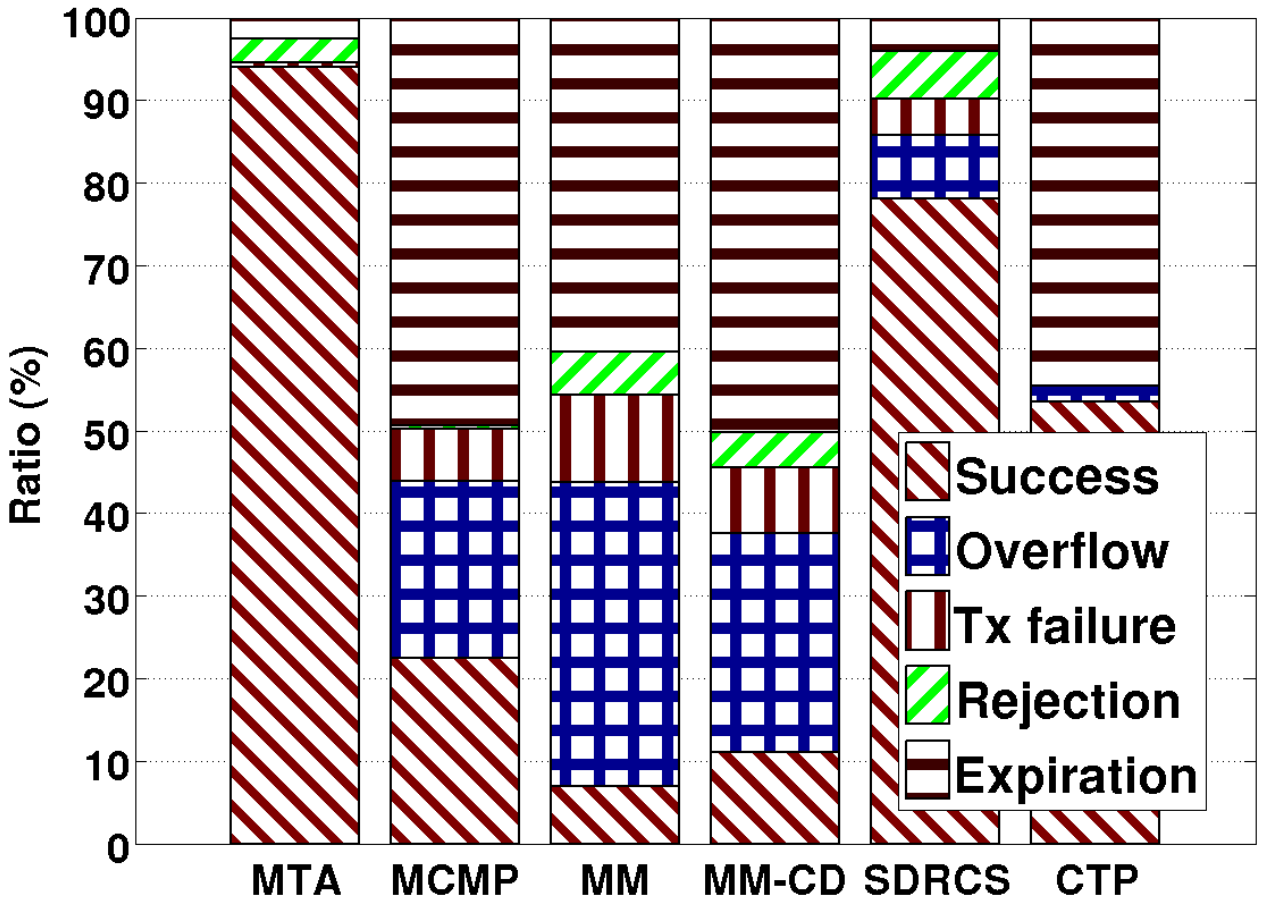

Figure 38: Packet delivery status: MTA and existing protocols in the NetEye medium traffic scenario

protocols, the negative impact of this assumption is relatively less severe in SDRCS because it uses signal strength as the basis of measuring forwarding distances and signal strength is a better metric for measuring wireless link quality than geographic distance. MTA does not have this problem because MTE enables accurate, agile estimation of end-to-end delay quantiles without assuming uniform network conditions. A third reason for the low performance of MM, MM-CD, and SDRCS is because they only consider mean delays instead of the probabilistic distributions of delays.

Compared with MTA, CTP has higher packet delivery ratio, but CTP only enables a median deadline success ratio of 56% which is lower than the real-time guarantee probability of 90% and much lower than the 93% probability guarantee by MTA. This is because CTP only considers path ETX in routing, and it is delay-unaware. Even if a low-ETX path is experiencing large delay due to queueing, CTP still uses the path, thus leading to large data delivery delay and deadline miss. Through accurate, agile estimation of path delay via MTE, in contrast, MTA can switch to a less congested path whenever it detects the inability of the current low-ETX path to deliver packets before their deadlines, creating the water-filling effect as we have discussed in Chapter 4. For instance, Figure 39 shows the histogram of the ETX of the paths taken by all the packets. We see that MTA tends to use paths of lower-ETX with higher probability, even though the minimum-ETX path will not always be used with the highest probability (e.g., when the capacity of the minimum-ETX path is reduced due to the shared path segments with other paths).

For periodic light traffic in NetEye, Figures 40, 41, 42, and 43 show the deadline success ratio, packet delivery ratio, transmission cost, and packet delivery status respectively. The results are similar to those of periodic medium traffic except that CTP's performance is now indistinguishable from MTA's while it is worse than MTA's in medium traffic. Both protocols yield close to 100% packet delivery and deadline success and similar transmission costs. This is because, in light traffic, there is almost no queueing. Hence, a path's ETX is a good indicator of its delay, and MTA tends to choose minimal ETX path as CTP does, resulting in similar performance.

For periodic heavy traffic in NetEye, Figures 44, 45, 46, and 47 show the deadline success ratio, packet delivery ratio, transmission cost, and packet delivery status respectively. Again, the

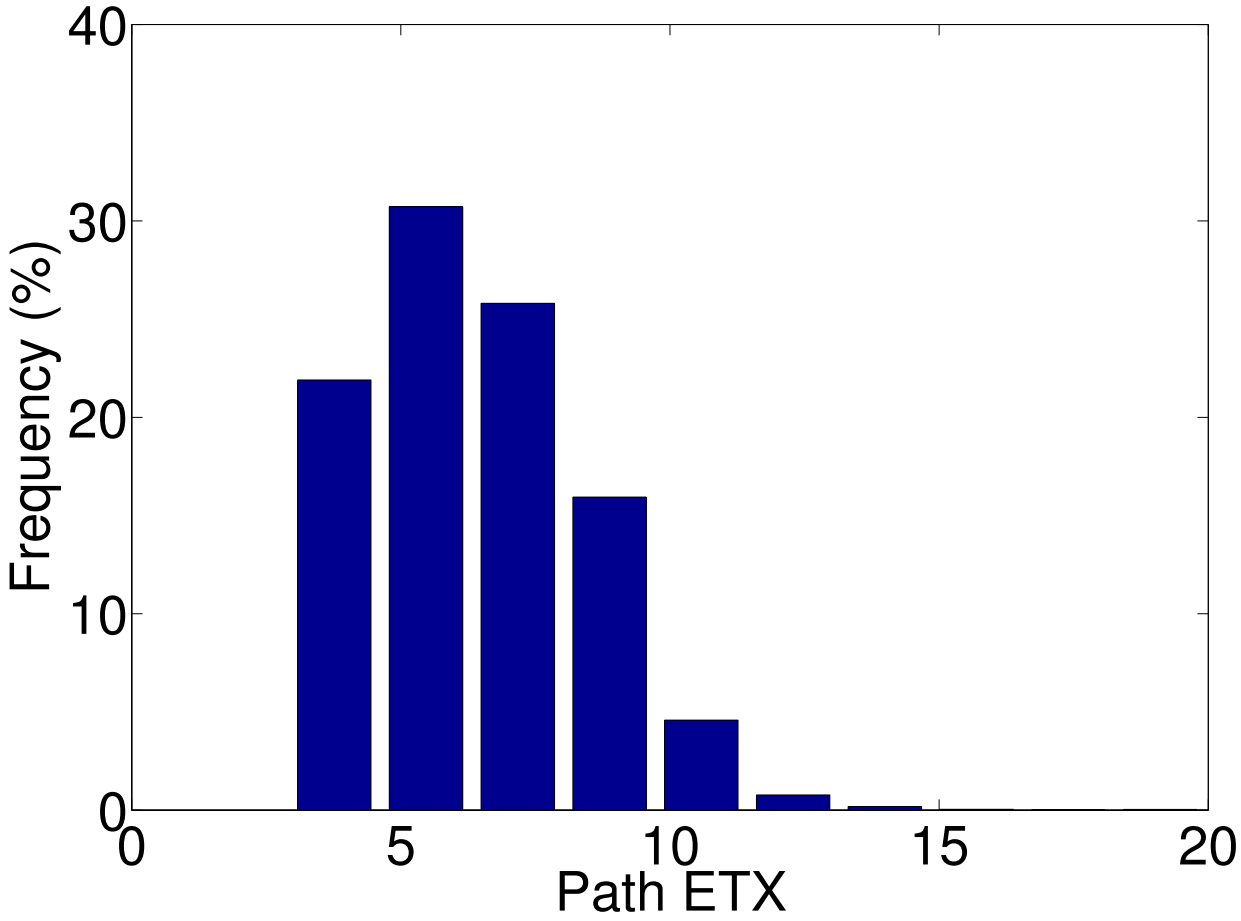

Figure 39: Histogram of the ETX of the paths taken by all the packets

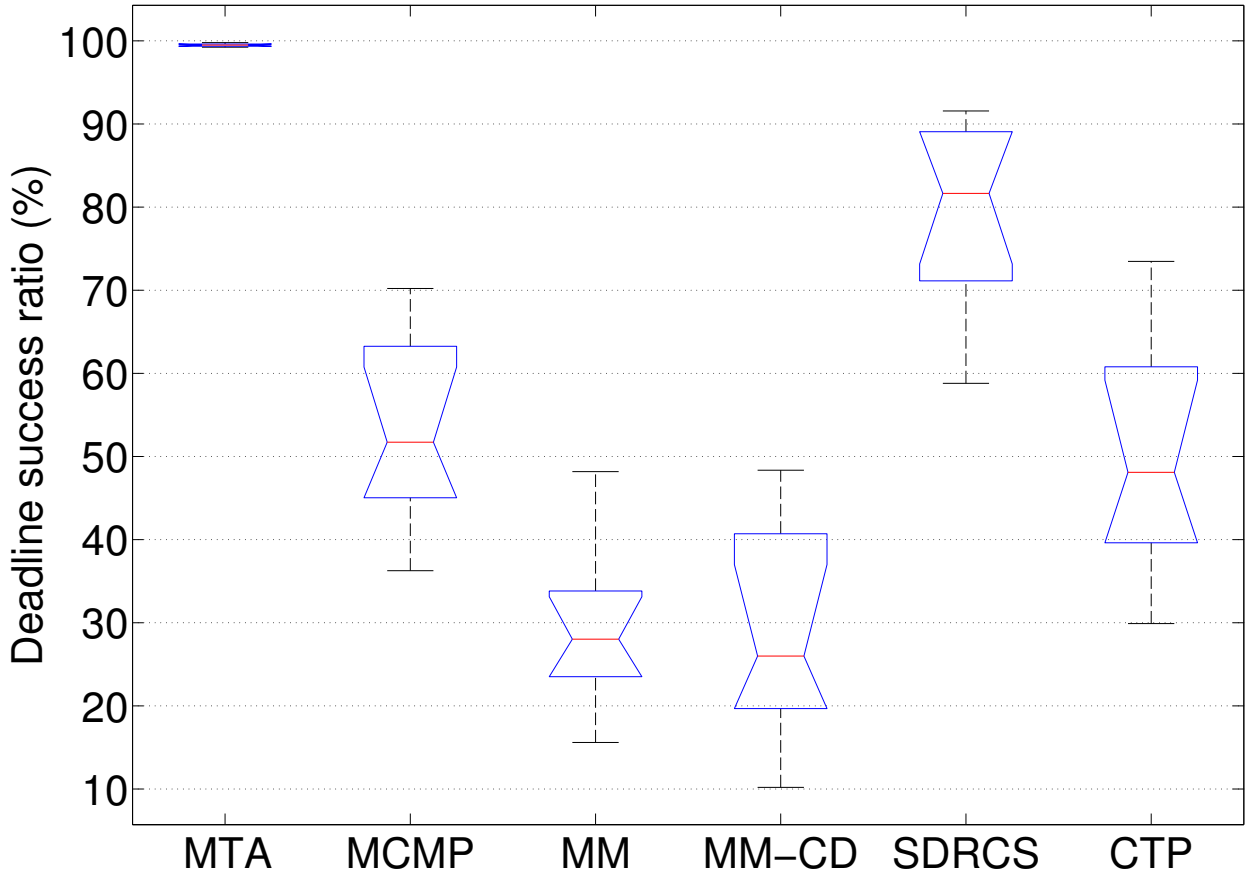

Figure 40: Deadline success ratio: MTA and existing protocols in the NetEye light traffic scenario

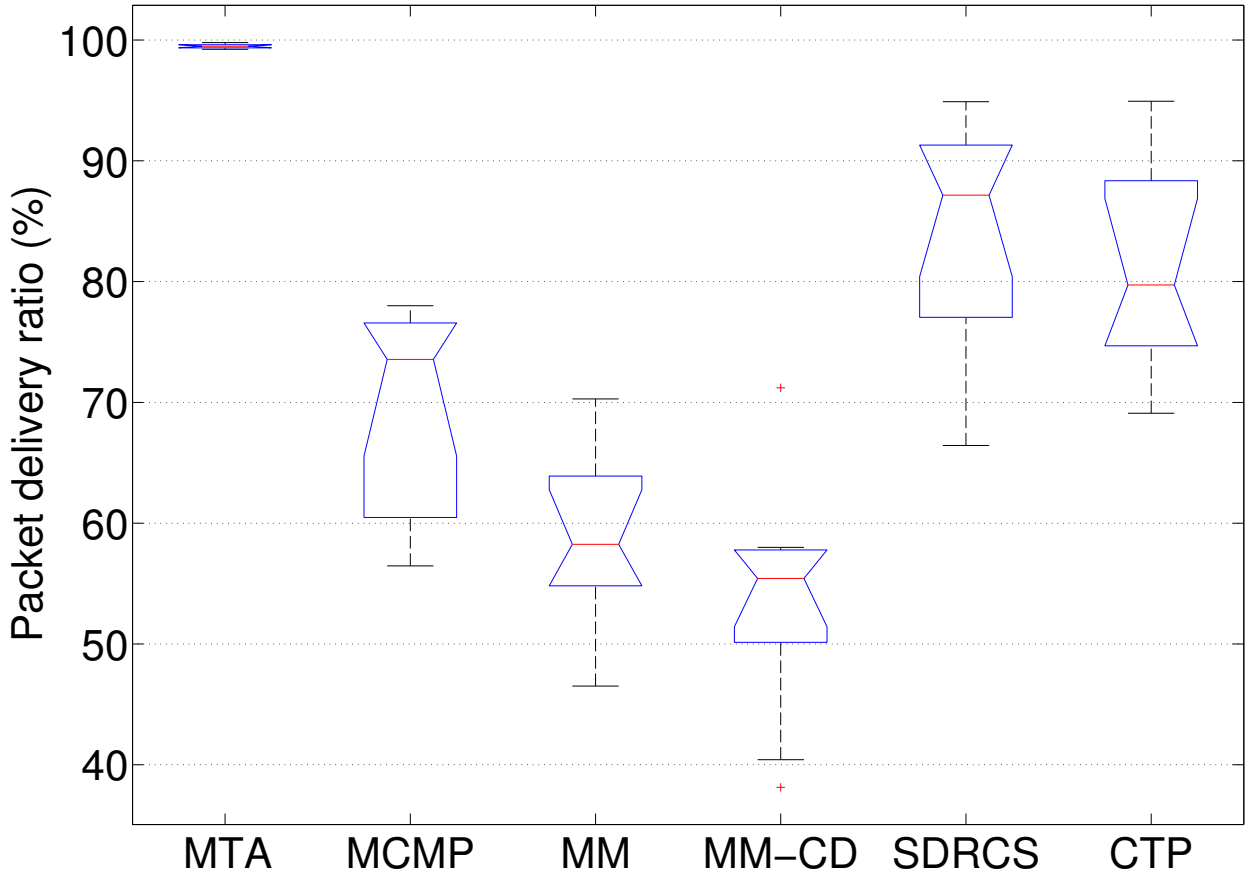

Figure 41: Packet delivery ratio: MTA and existing protocols in the NetEye light traffic scenario

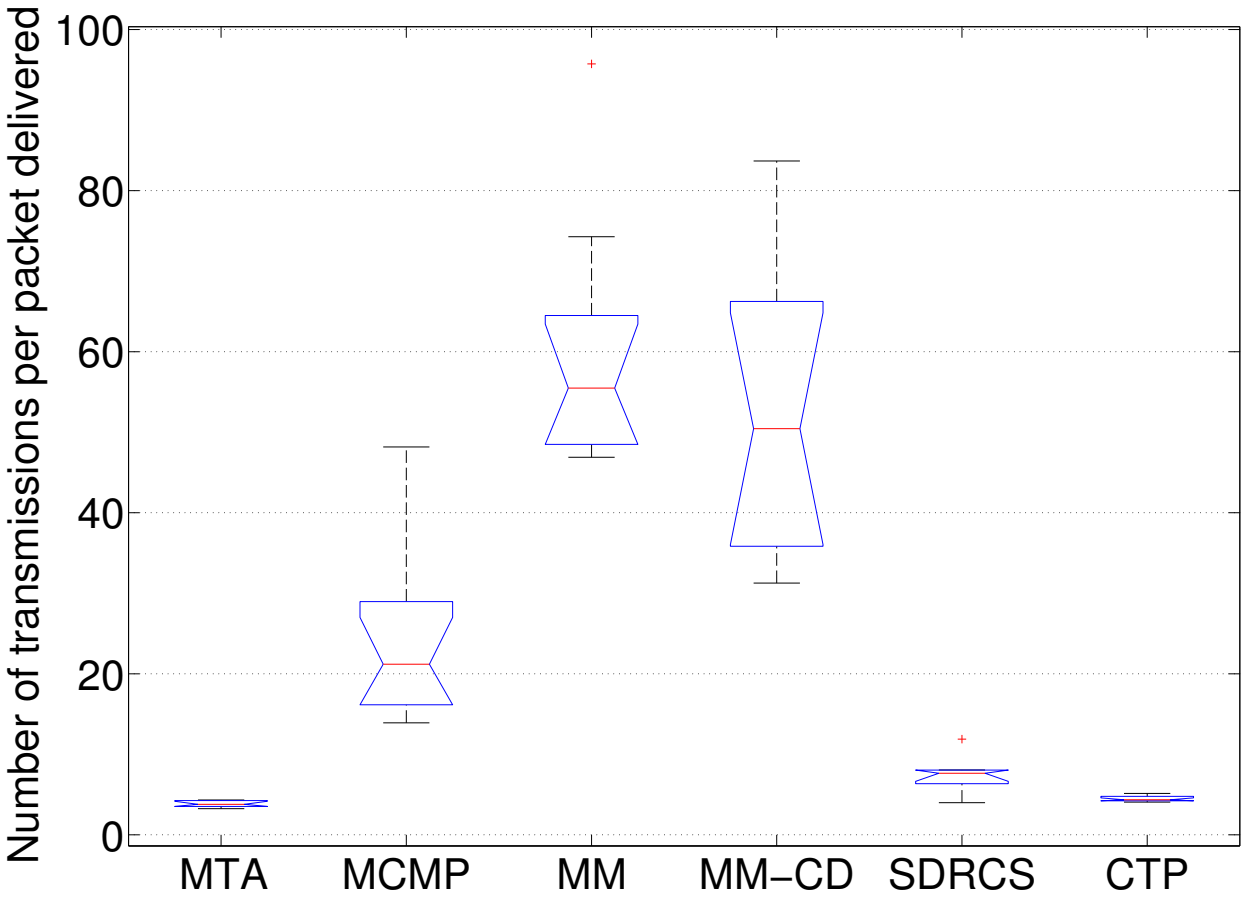

Figure 42: Number of transmissions per packet delivered: MTA and existing protocols in the NetEye light traffic scenario

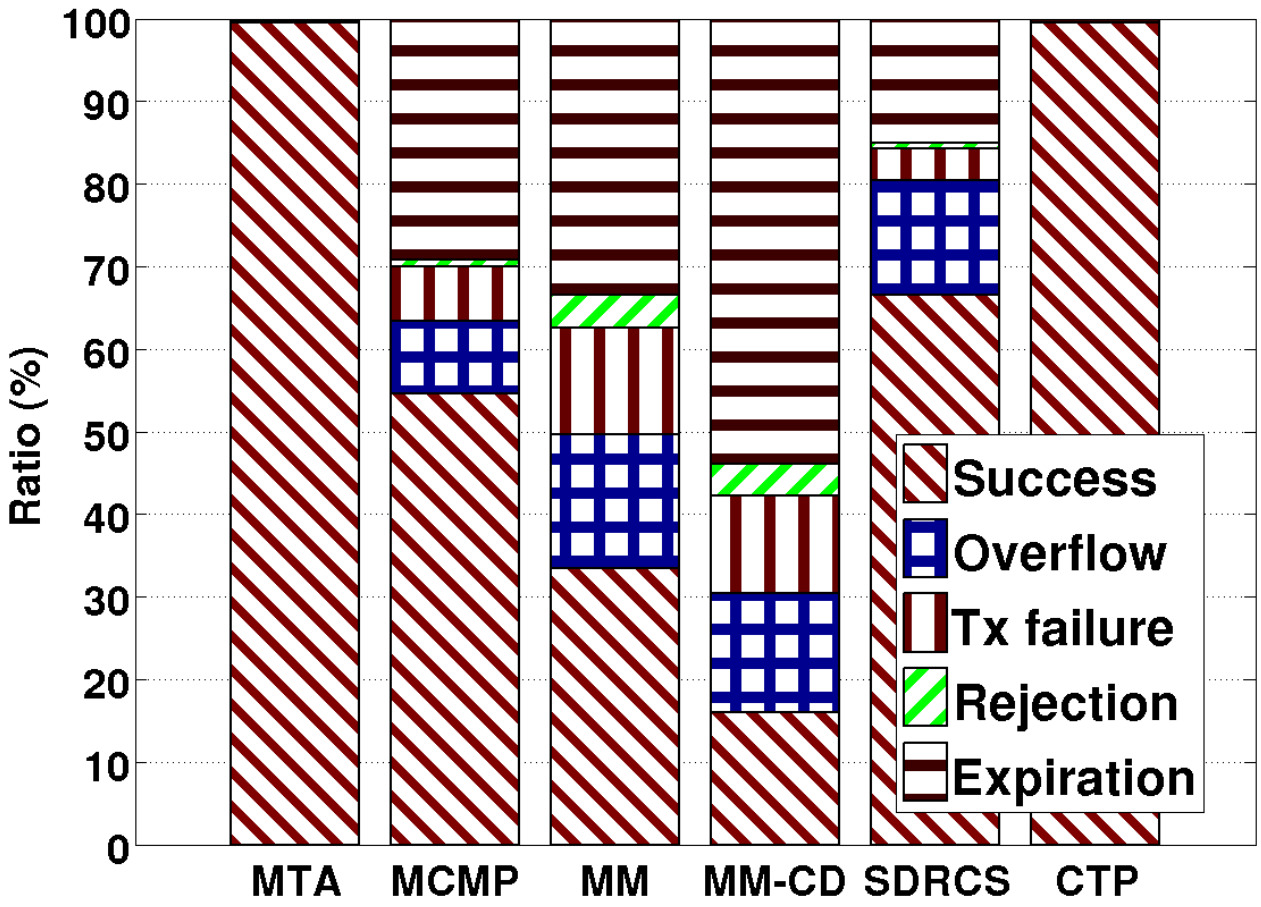

Figure 43: Packet delivery status: MTA and existing protocols in the NetEye light traffic scenario

results are similar to those in medium traffic, and MTA consistently guarantees deadline success ratio over the required 90%. In CTP, however, more packets are lost due to queue overflow in the presence of heavy traffic, thus leading to a median packet delivery ratio of around 80% (compared with its packet delivery ratio of around 100% in medium traffic).

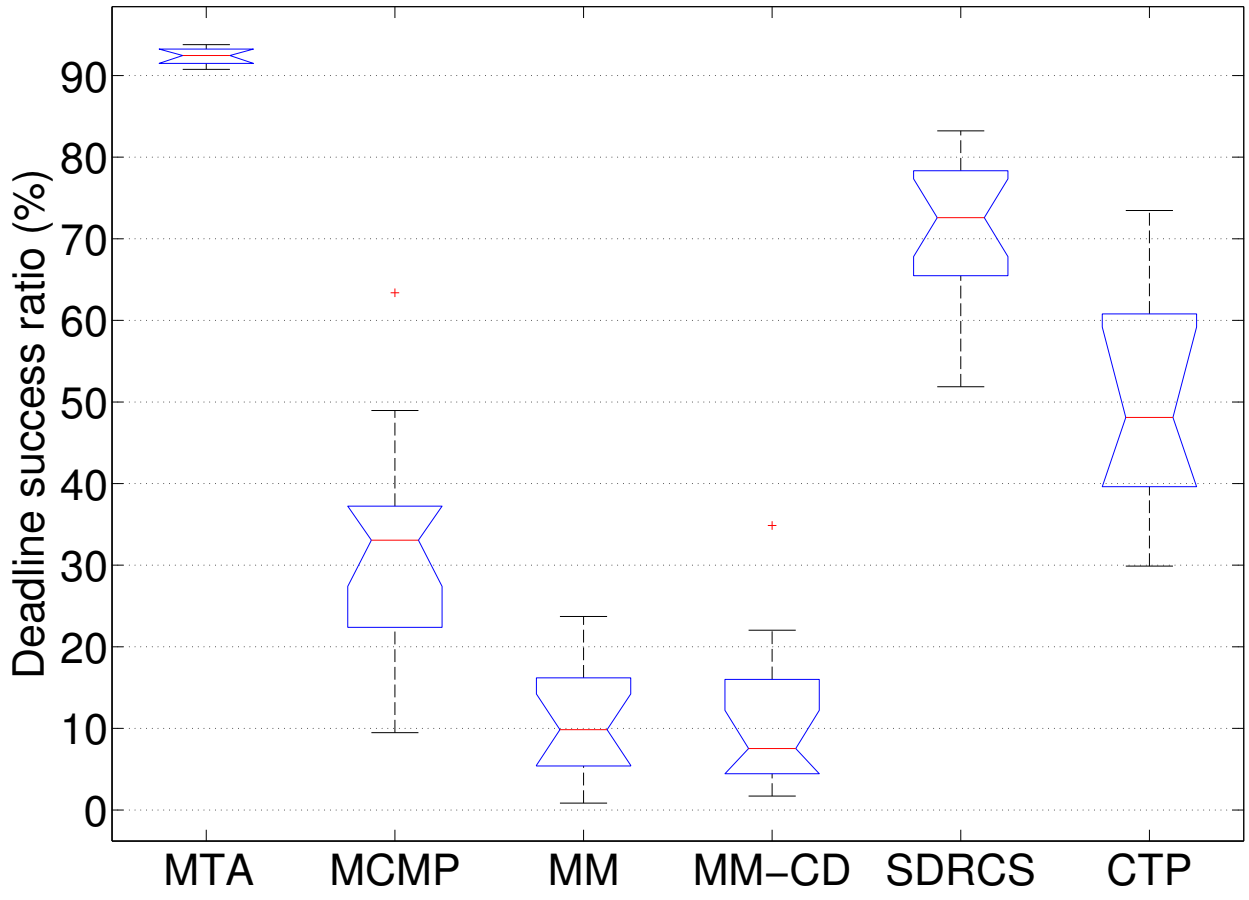

Figure 44: Deadline success ratio: MTA and existing protocols in the NetEye heavy traffic scenario

For event traffic in NetEye, Figures 48, 49, 50, and 51 show the deadline success ratio, packet delivery ratio, transmission cost, and packet delivery status respectively. The comparative results closely resembles those in medium traffic, and MTA ensures a deadline success ratio greater than the required 90%.

All the previous traffic set the required real-time probability as 90%. To demonstrate that MTA works for arbitrarily required probability guarantee, we raise the probability to 99% and set the corresponding deadline as 7.5 seconds. For the same periodic medium traffic in NetEye,

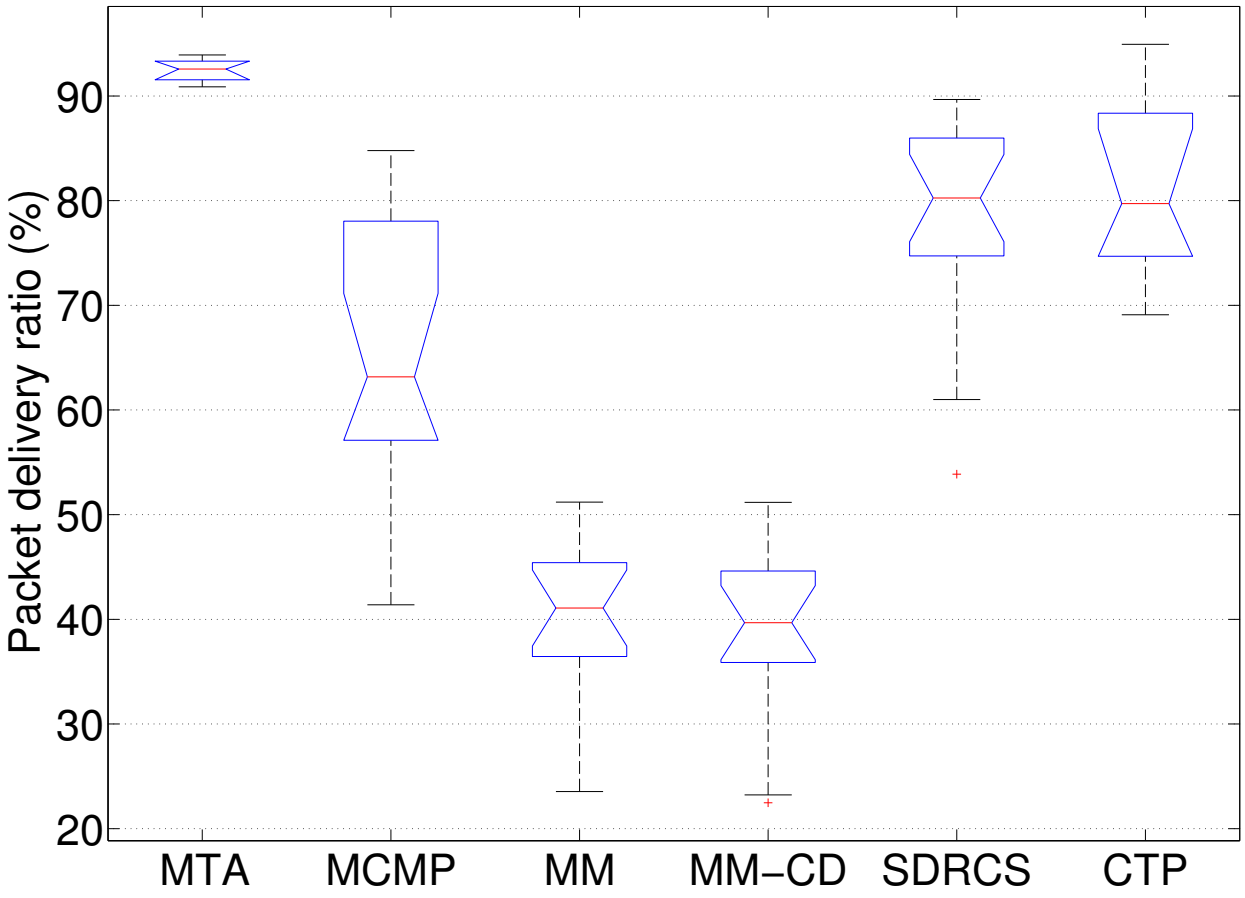

Figure 45: Packet delivery ratio: MTA and existing protocols in the NetEye heavy traffic scenario

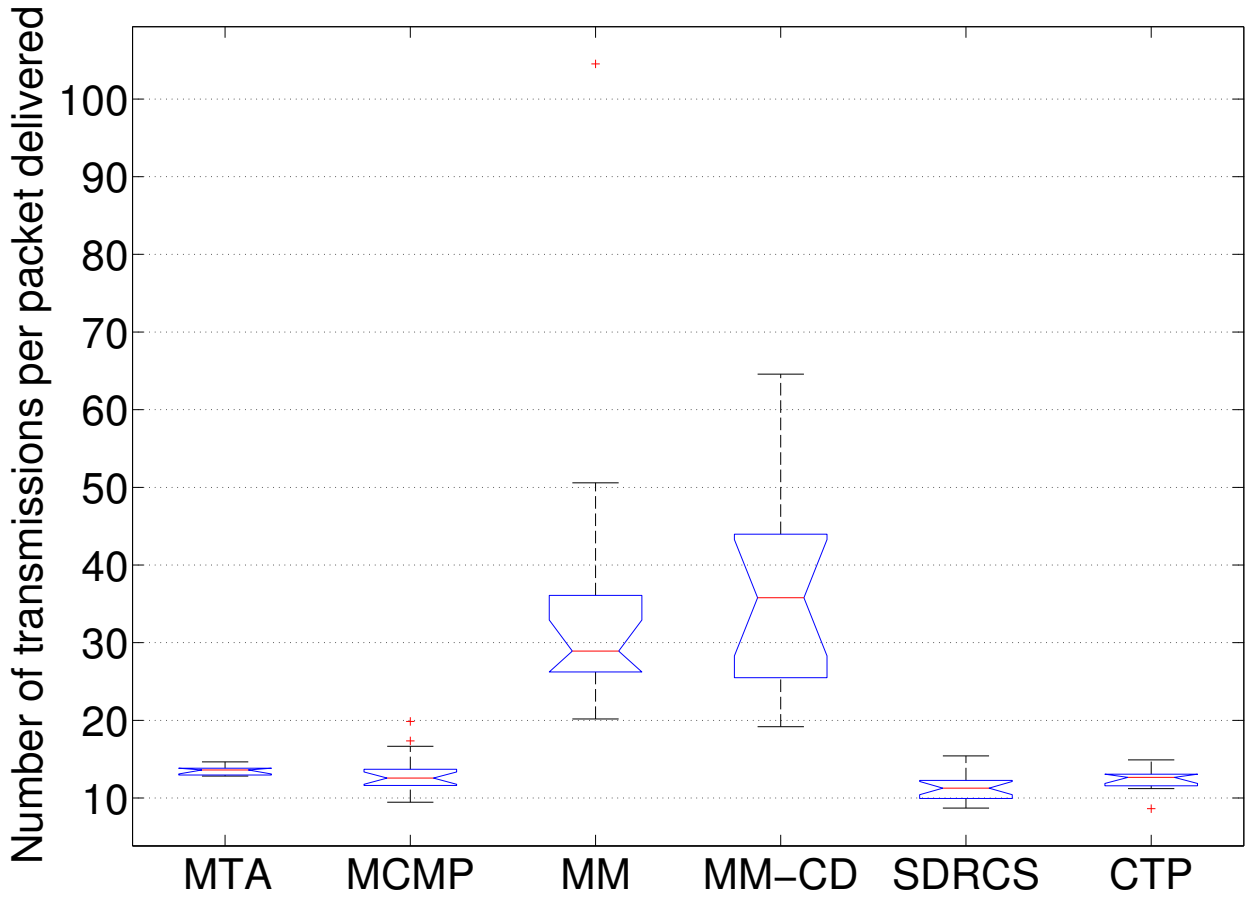

Figure 46: Number of transmissions per packet delivered: MTA and existing protocols in the NetEye heavy traffic scenario

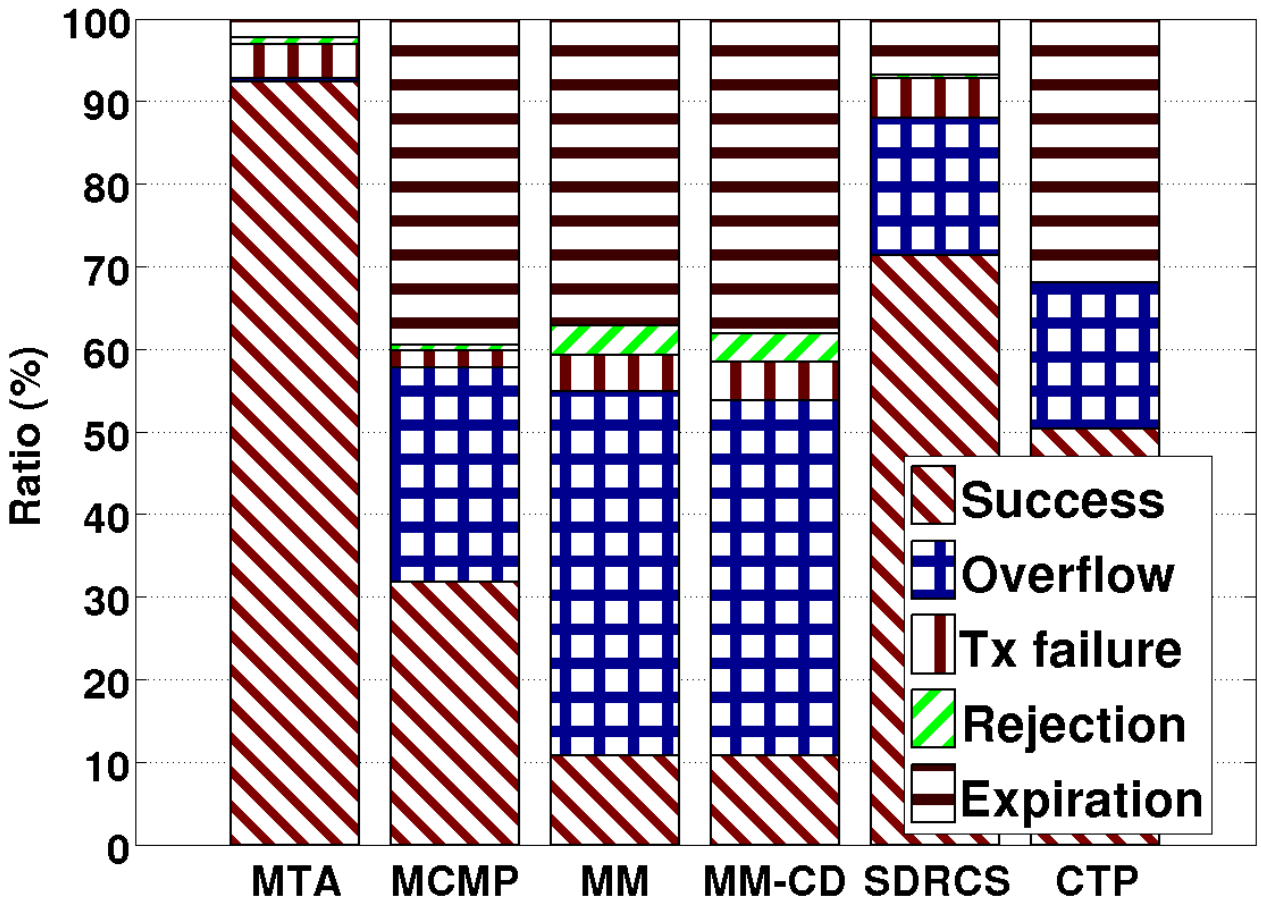

Figure 47: Packet delivery status: MTA and existing protocols in the NetEye heavy traffic scenario

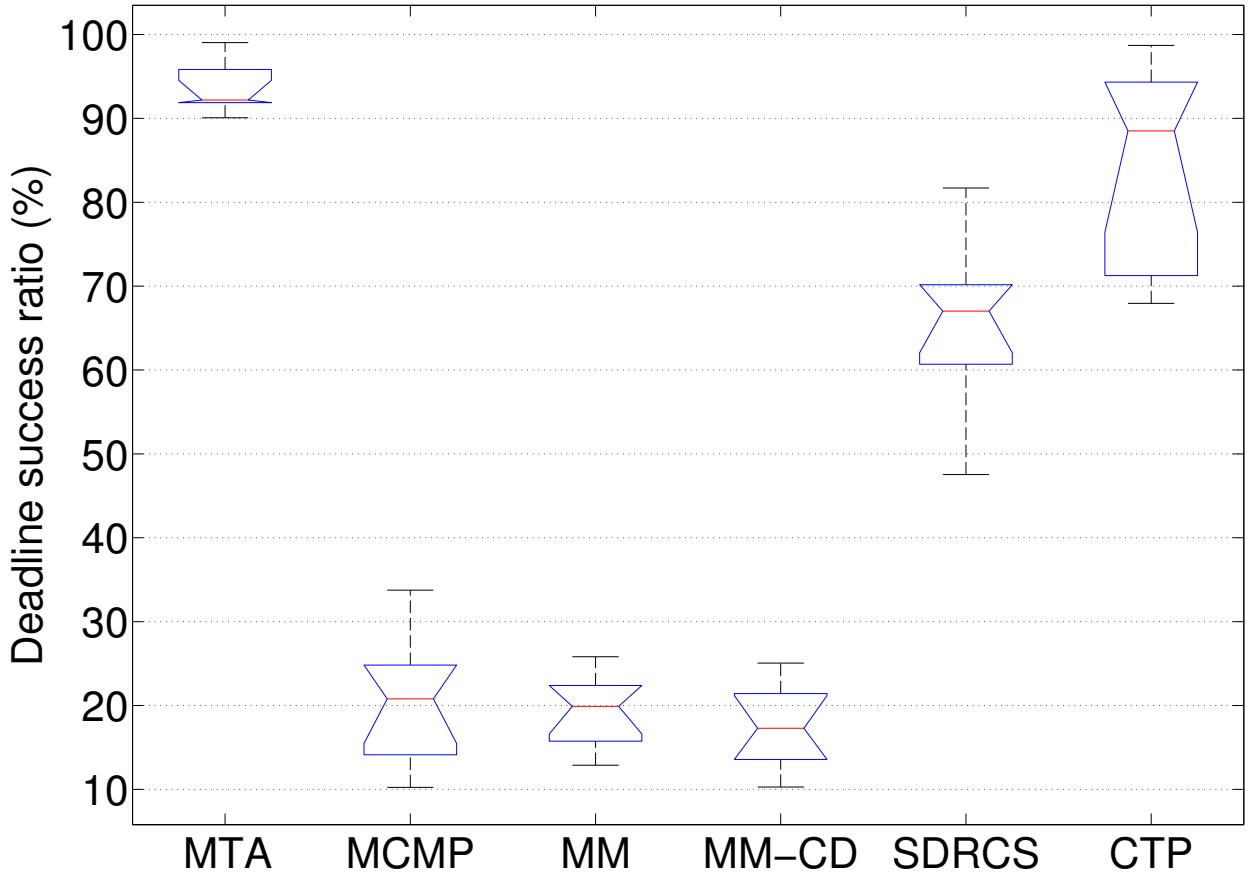

Figure 48: Deadline success ratio: MTA and existing protocols in the NetEye event traffic scenario

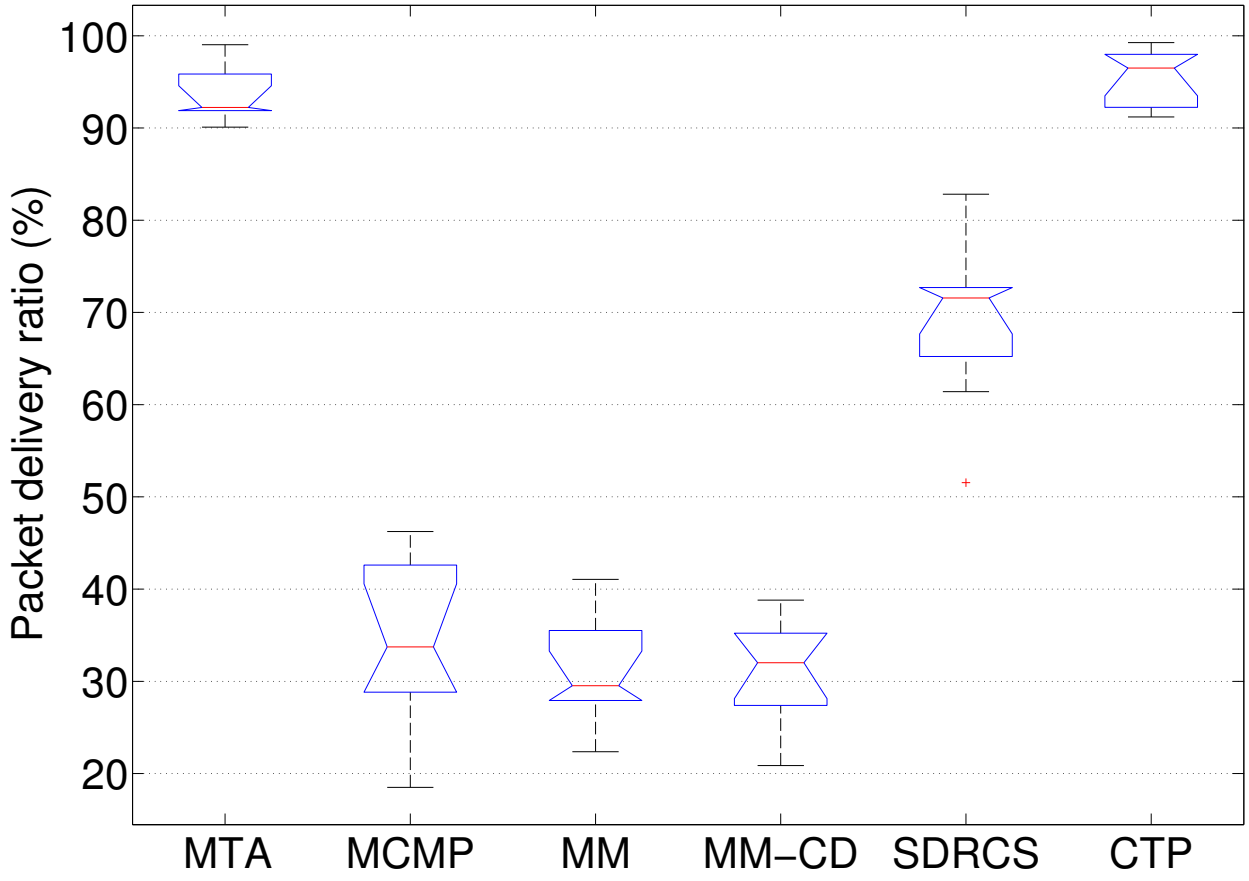

Figure 49: Packet delivery ratio: MTA and existing protocols in the NetEye event traffic scenario

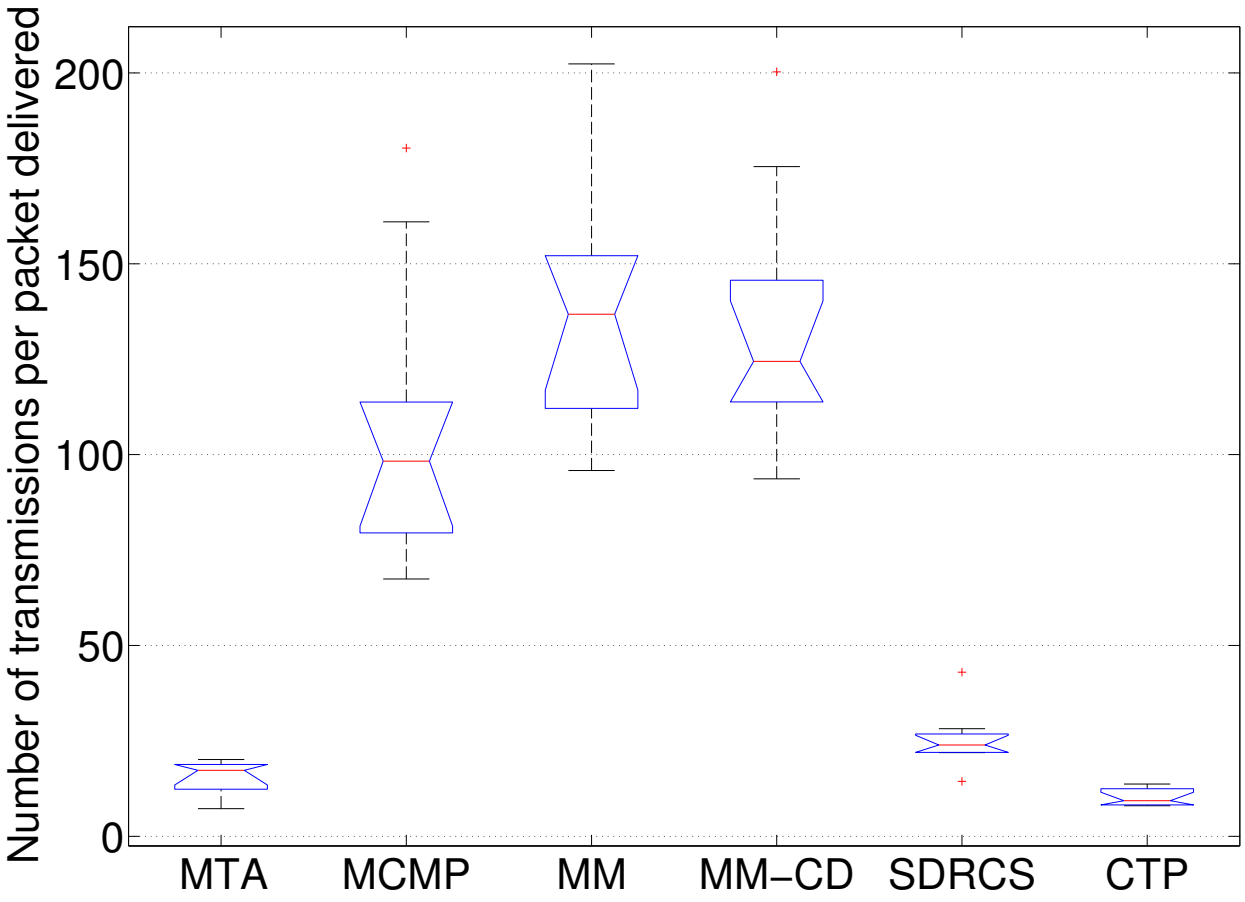

Figure 50: Number of transmissions per packet delivered: MTA and existing protocols in the NetEye event traffic scenario

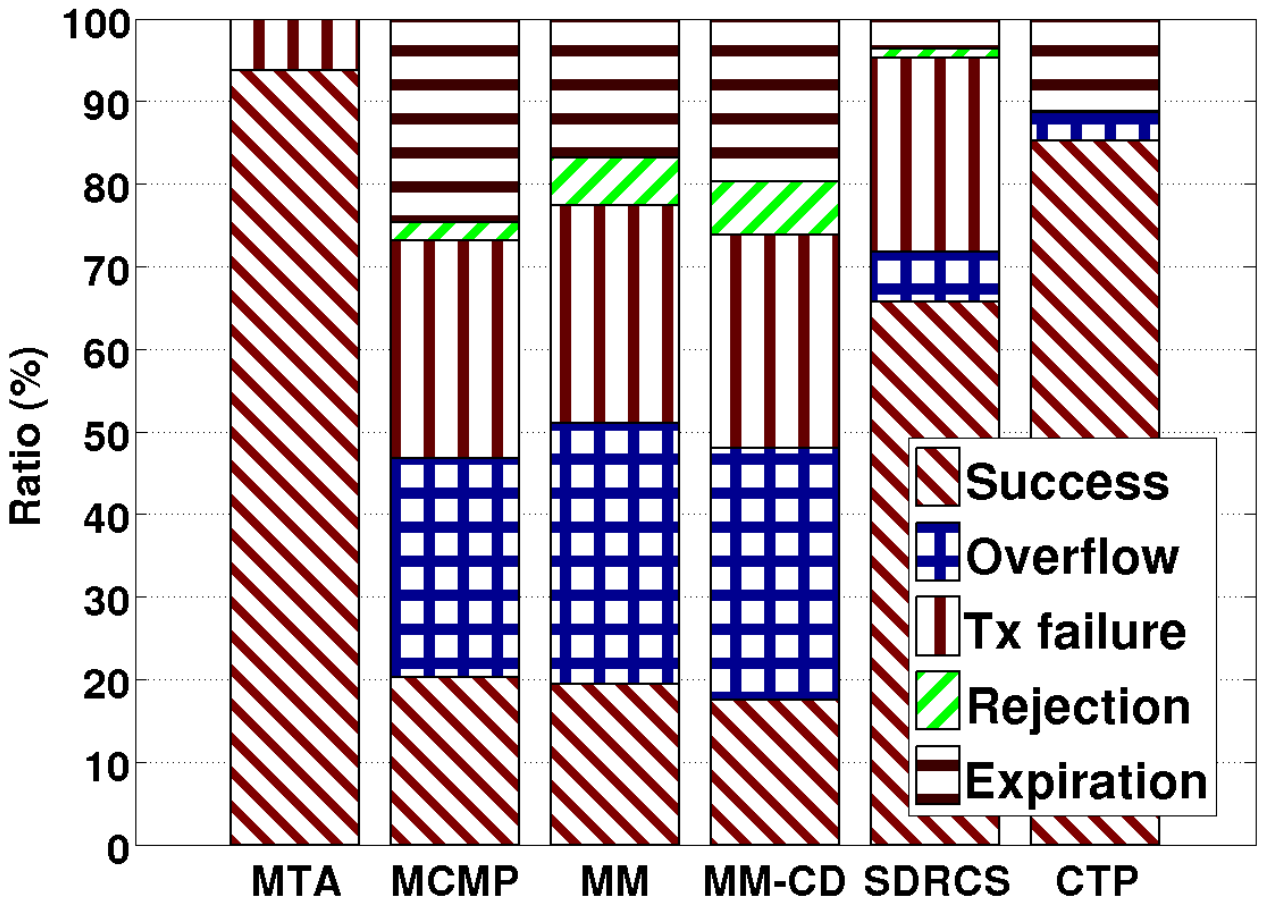

Figure 51: Packet delivery status: MTA and existing protocols in the NetEye event traffic scenario

Figures 52, 53, 54, and 55 show the deadline success ratio, packet delivery ratio, transmission cost, and packet delivery status respectively. Even the required probability guarantee of 99% is more stringent than 90%, Figure 52 shows that MTA still consistently meets the deadline. All the relative performance comparisons are similar to those of periodic medium traffic with required probability 90%.

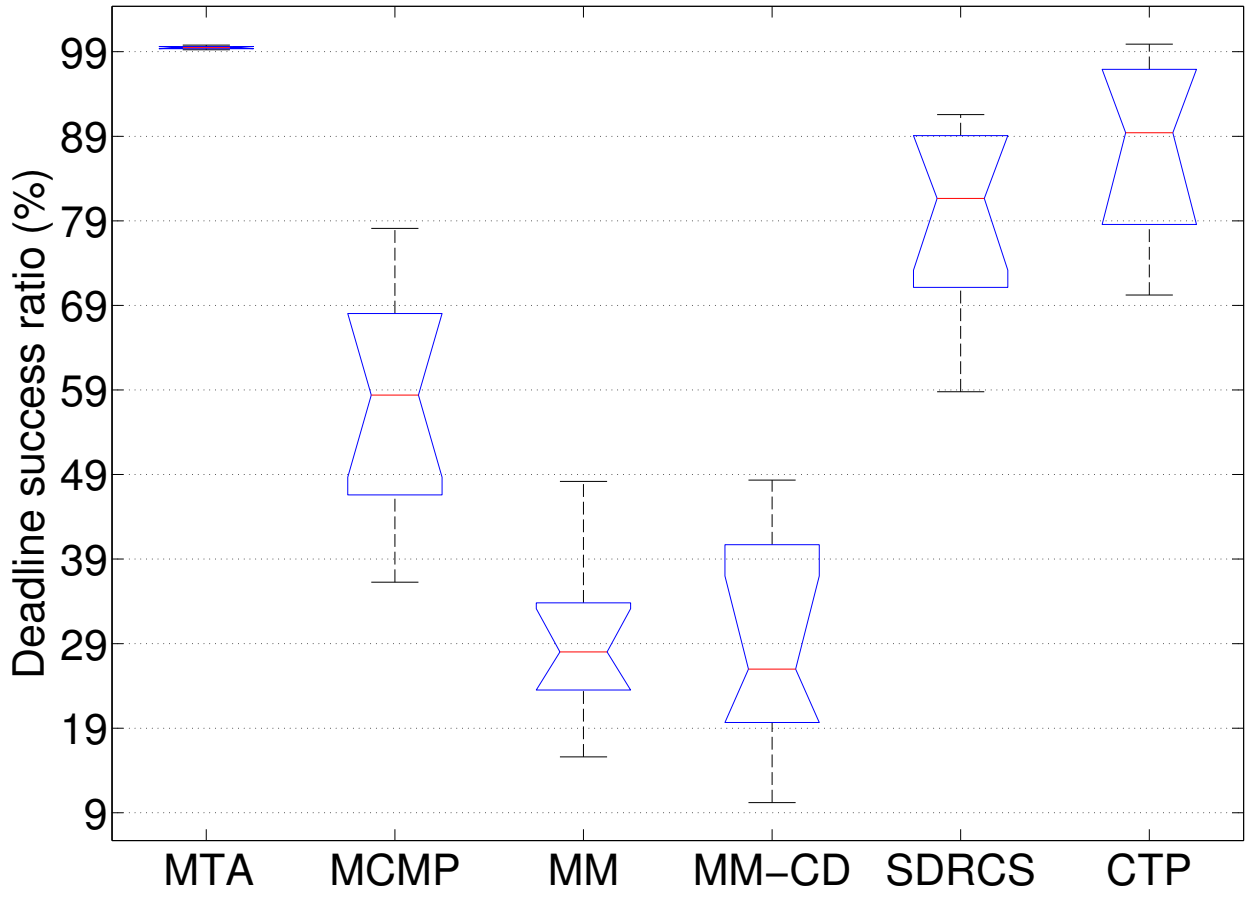

Figure 52: Deadline success ratio: MTA and existing protocols of under medium traffic (real-time probability 99%) in NetEye

Indriya. Figures 56, 57, 58, and 59 show the deadline success ratio, packet delivery ratio, transmission cost, and packet delivery status for periodic traffic in Indriya respectively. The overall relative behavior between protocols is similar to that in NetEye, but the performance of MM, MM-CD, and SDRCS become much worse compared with MTA. MTA still ensures real-time data delivery, but MM and MM-CD can hardly deliver any packet to the sink, let alone delivering

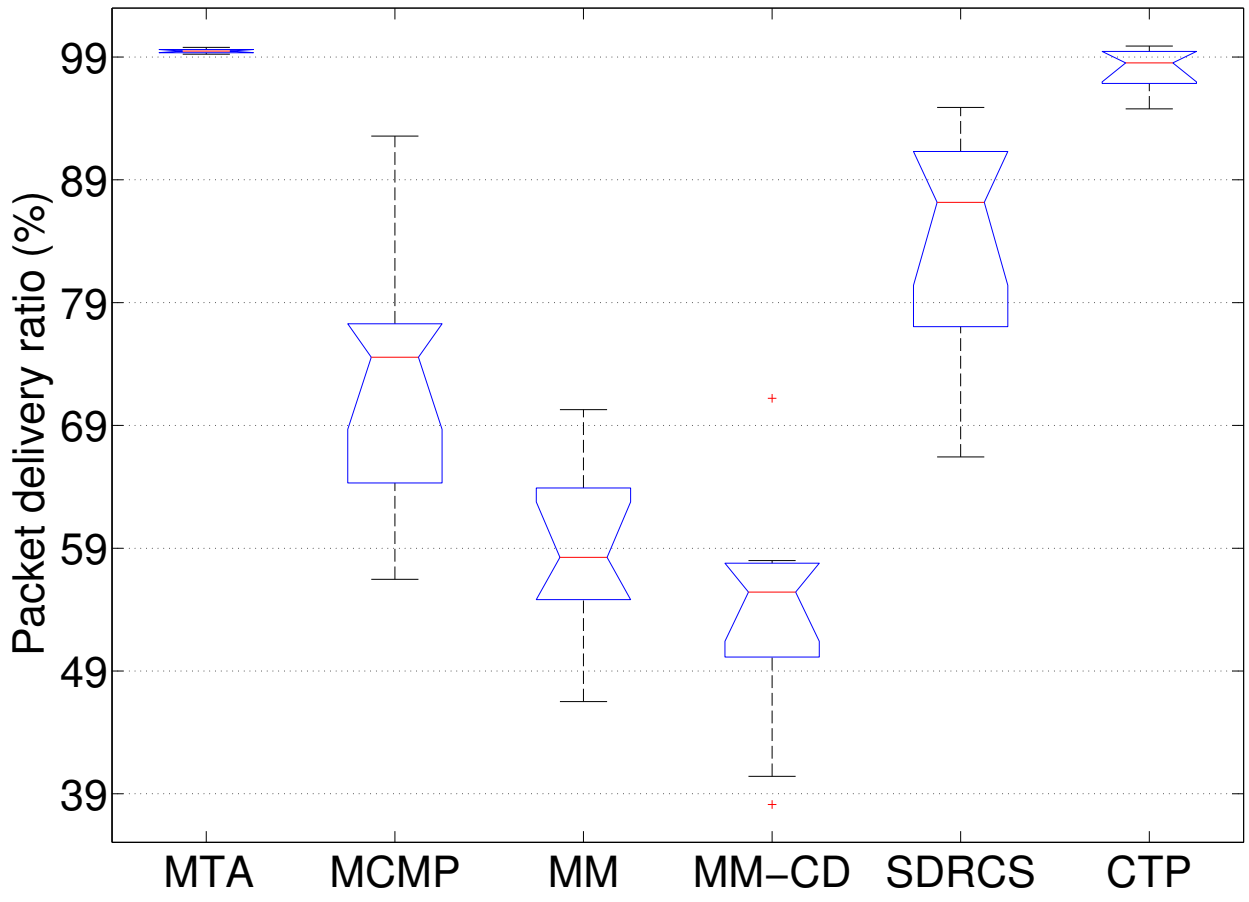

Figure 53: Packet delivery ratio: MTA and existing protocols under medium traffic (real-time probability 99%) in NetEye

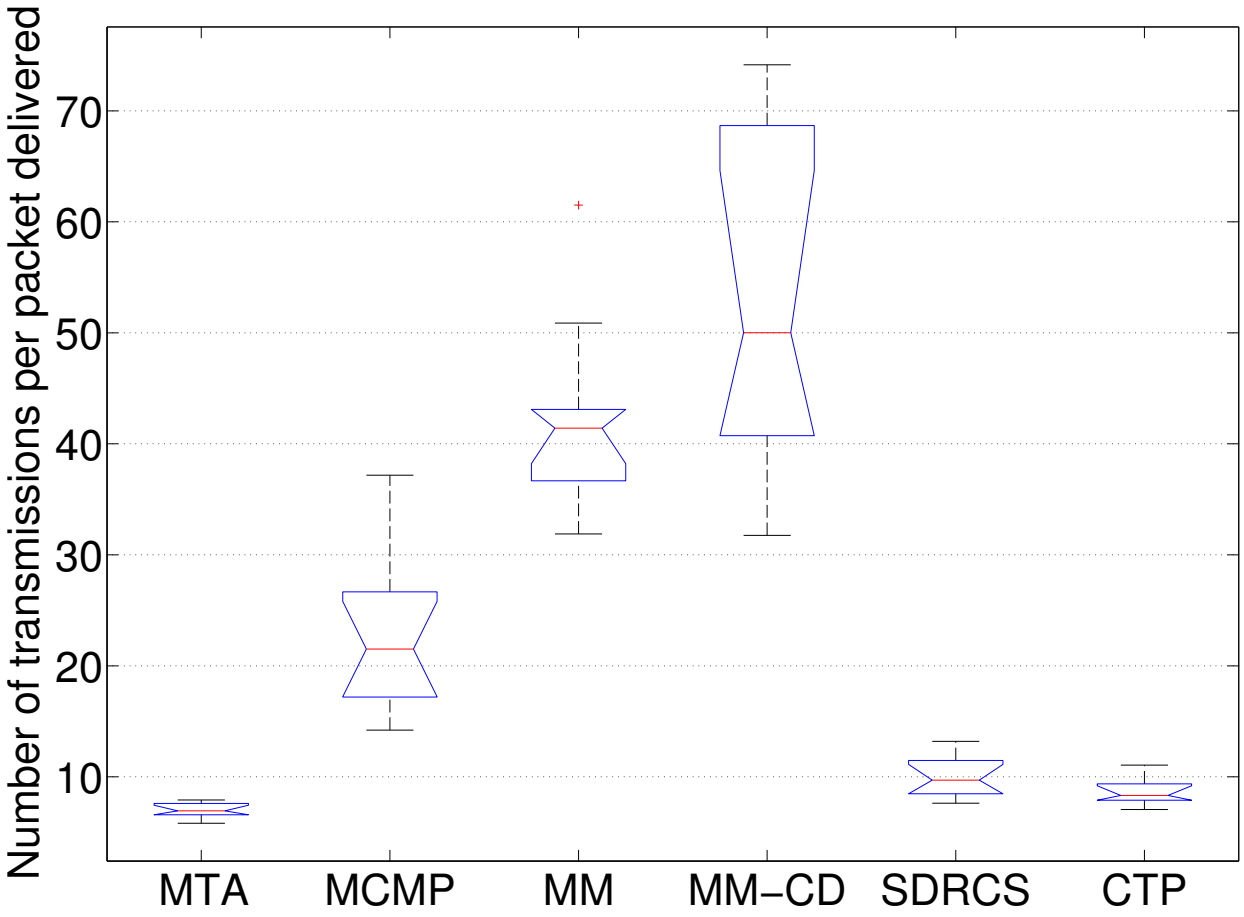

Figure 54: Number of transmissions per packet delivered: MTA and existing protocols under medium traffic (real-time probability 99%) in NetEye

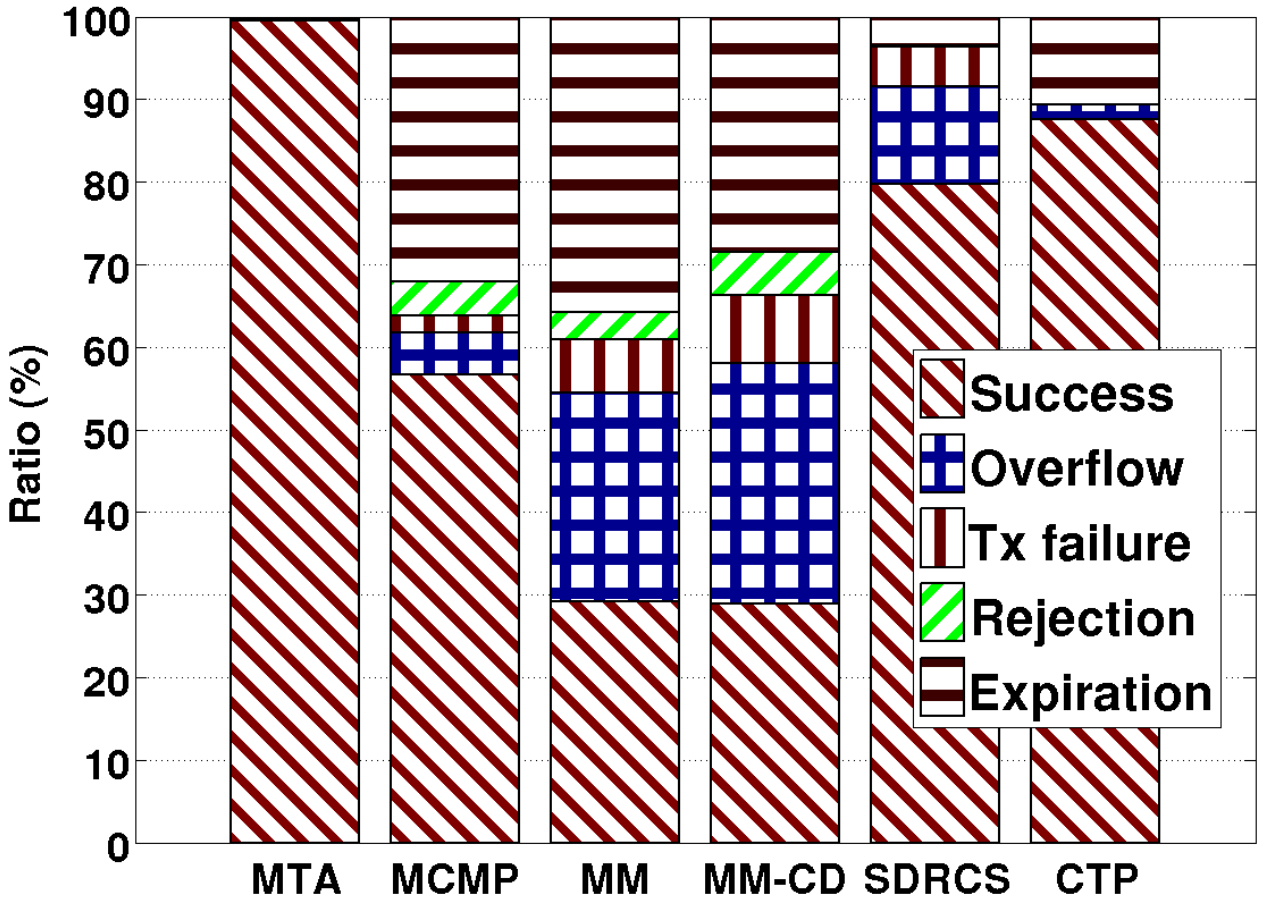

Figure 55: Packet delivery status: MTA and existing protocols under medium traffic (real-time probability 99%) in NetEye

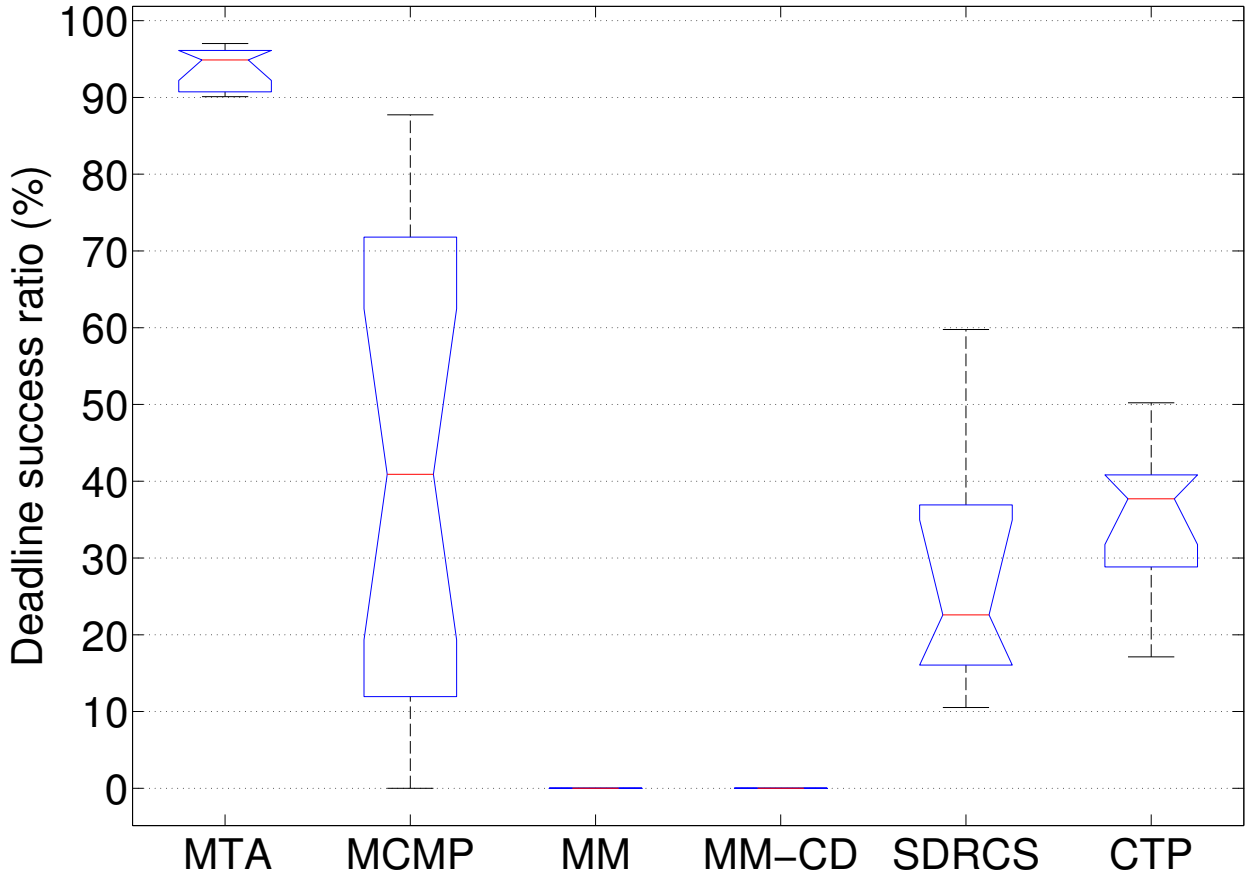

Figure 56: Deadline success ratio: MTA and existing protocols in Indriya

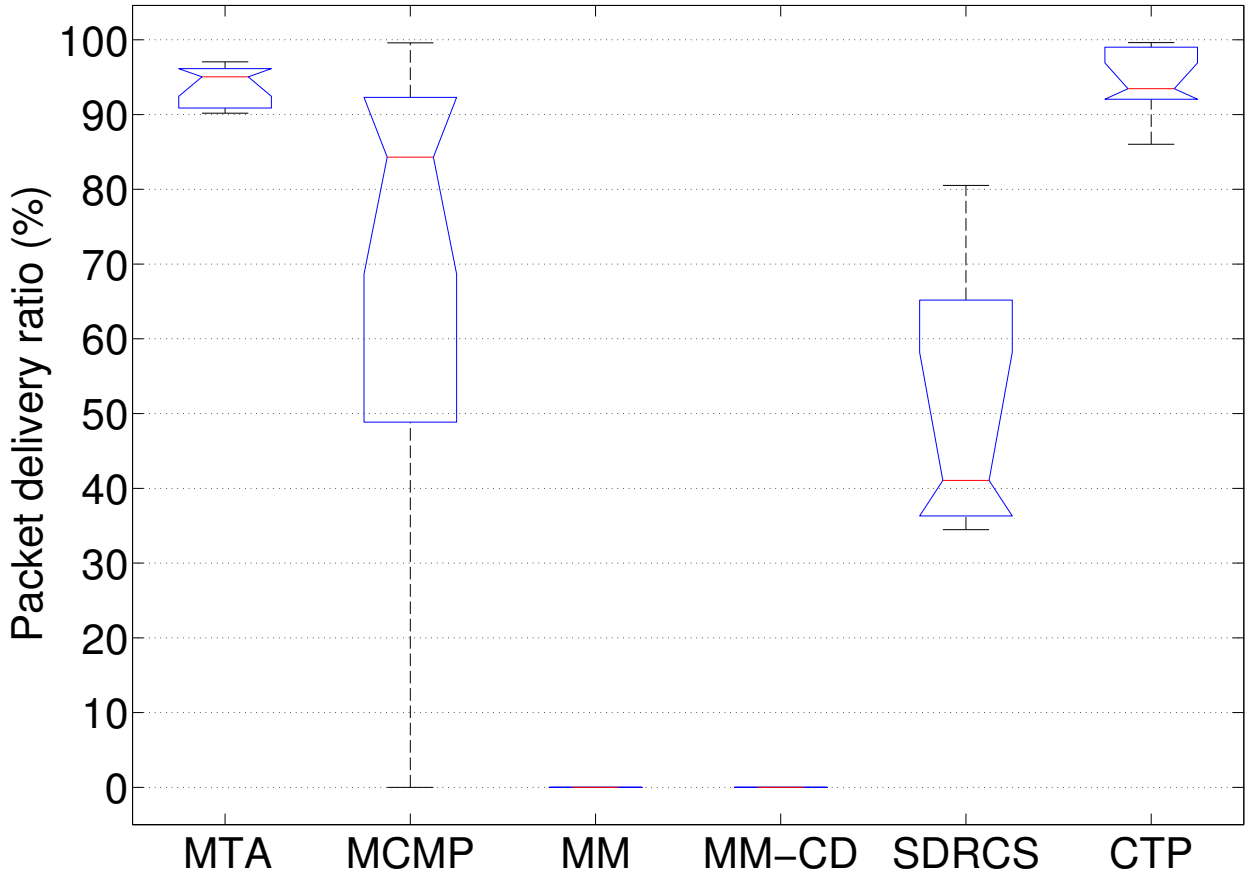

Figure 57: Packet delivery ratio: MTA and existing protocols in Indriya

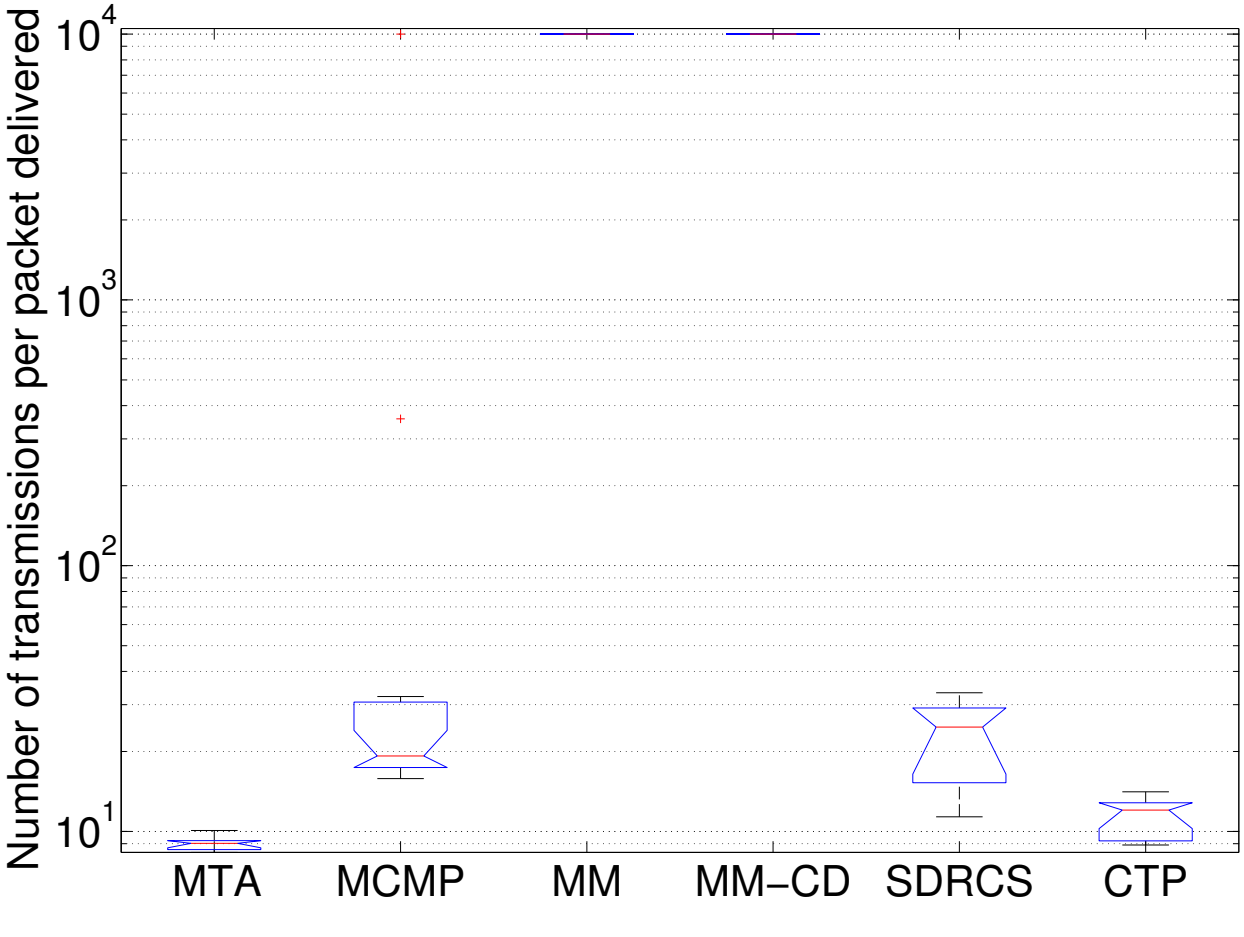

Figure 58: Number of transmissions per packet delivered: MTA and existing protocols in Indriya

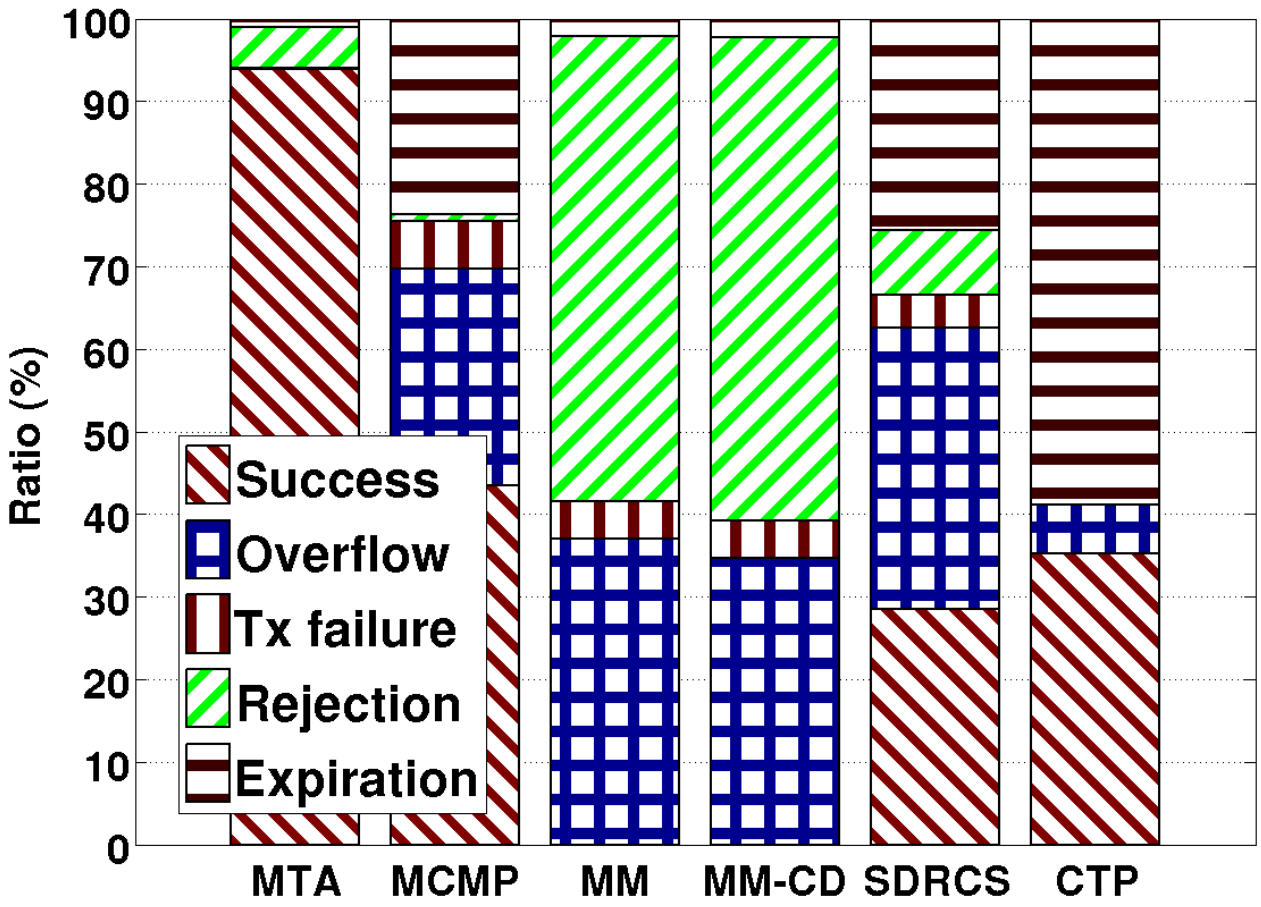

Figure 59: Packet delivery status: MTA and existing protocols in Indriya

packets in time; the deadline success ratio in SDRCS is also more than 72% less than that in MTA. One major cause for this is that MM, MM-CD, and SDRCS implicitly assume uniform network conditions while the degree of heterogeneity in Indriya is significant and higher than that in NetEye. As a result of the uniformity assumption, for instance, about 60% and 50% of packets are rejected in MM and MM-CD respectively as shown in Figure 59. Given a packet, more specifically, its deadline and the distance from its source to the sink determines the required forwarding speed for this packet. For the packet to reach the sink, every single hop it traverses has to provide a speed no less than the required speed. That is, if any intermediate hop cannot meet the required speed, a packet is rejected. For MM and MM-CD, it only takes about 3 or 4 hops to reach the sink from the sources in NetEye, while it takes about 7 or 8 hops in Indriya, making packets in Indriya more likely to be rejected. Note that there is significant queue overflow in SDRCS because SDRCS happen to use low reliability links in Indriya which reduce the network throughput and thus increase the queueing and queue overflow.

Figures 60, 61, 62, and 63 show the deadline success ratio, packet delivery ratio, transmission cost, and packet delivery status for event traffic in Indriya respectively. The results are similar to those of periodic traffic with some minor differences. Firstly, MM and MM-CD can deliver some packets in time instead of almost none as in periodic traffic, though their deadline catch ratio is less than 10%. Secondly, the ratio of overflows tend to decrease significantly compared to periodic traffic except for MTA and CTP. Both can be primarily attributed to the reduction of traffic load.

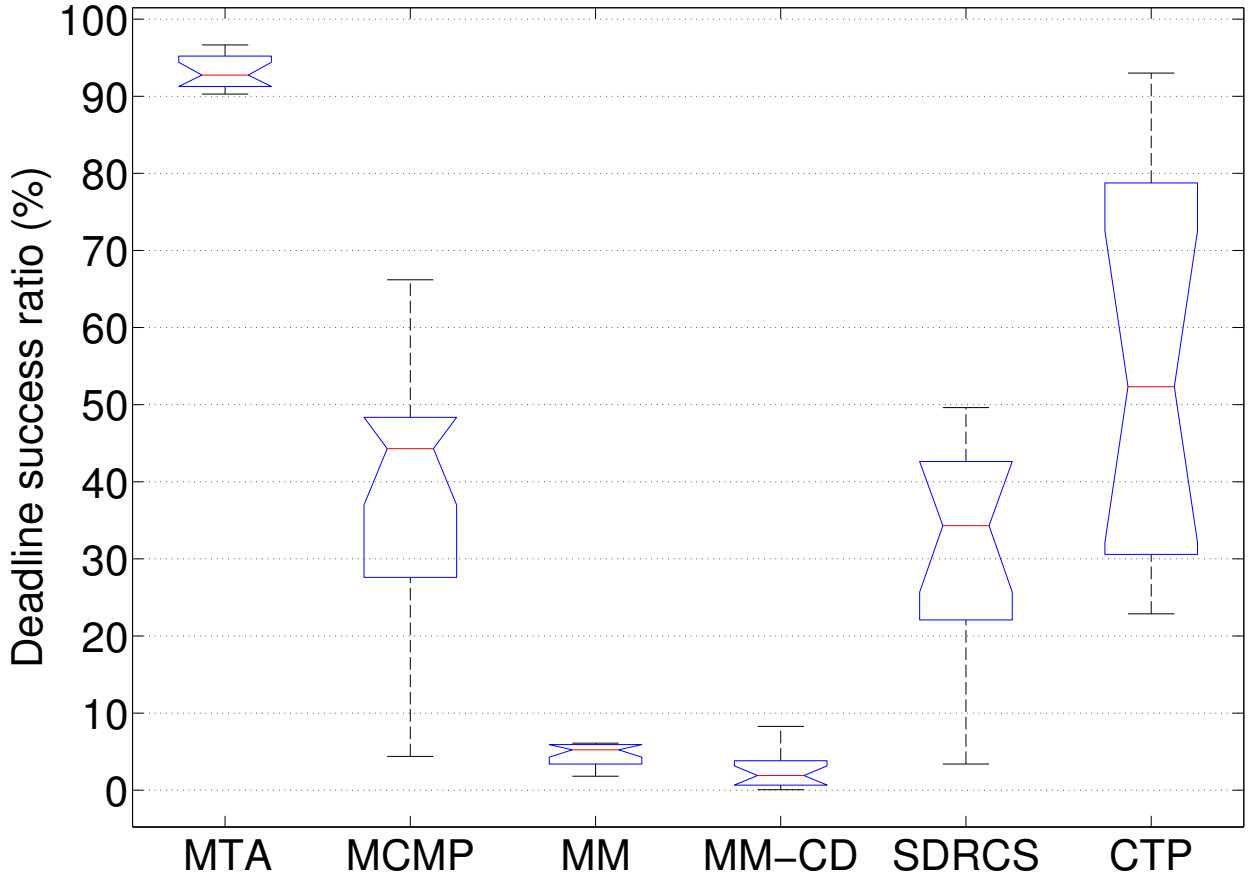

Figure 60: Deadline success ratio: MTA and existing protocols under event traffic in Indriya

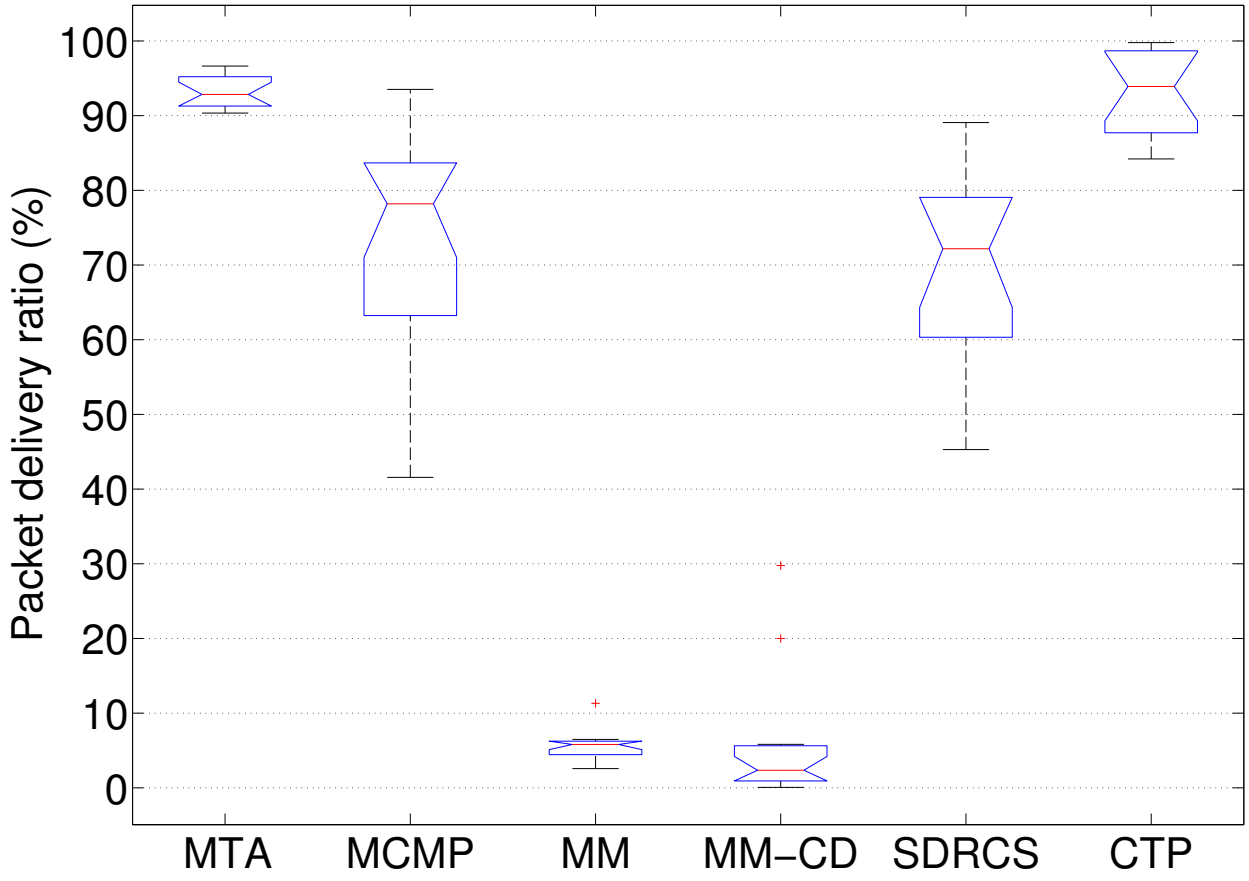

Figure 61: Packet delivery ratio: MTA and existing protocols under event traffic in Indriya

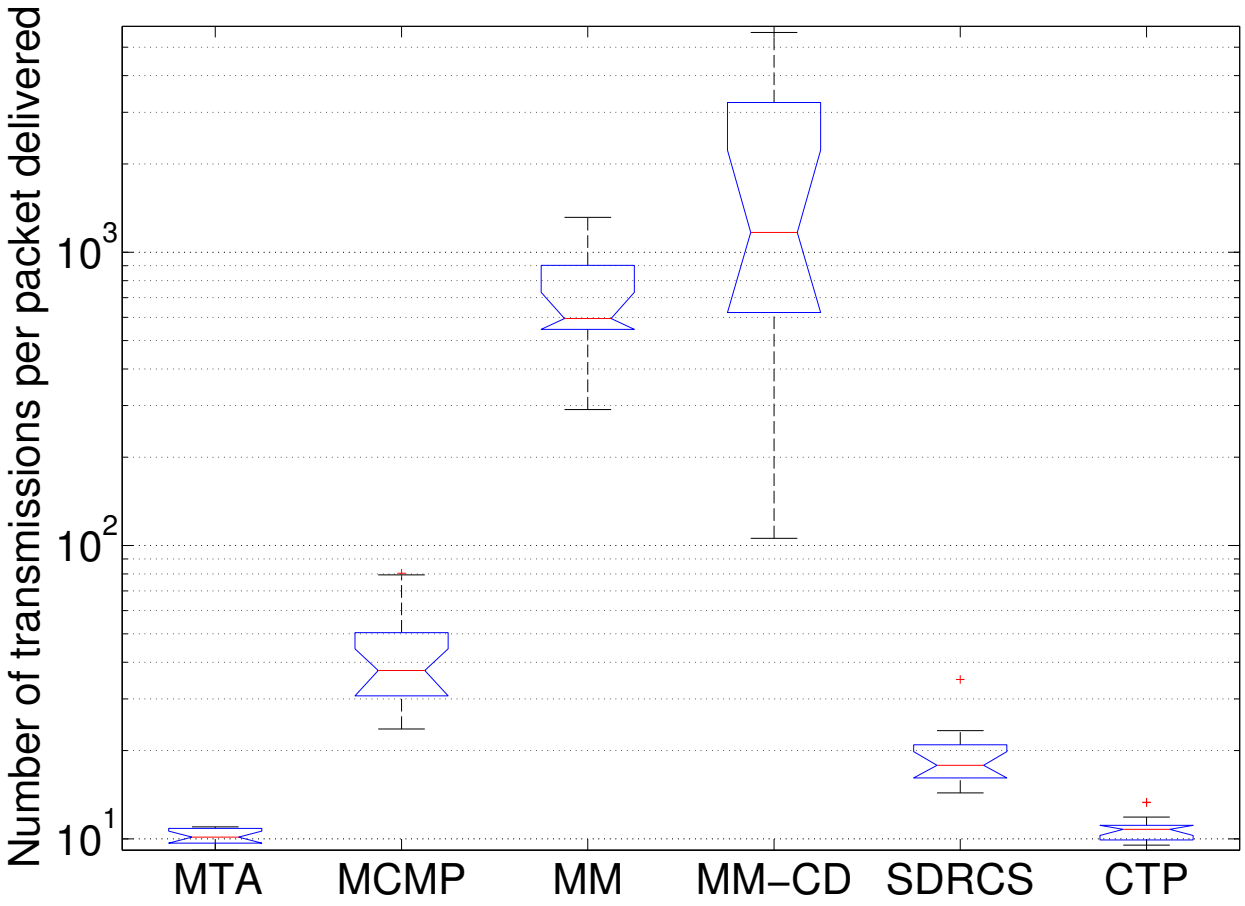

Figure 62: Number of transmissions per packet delivered: MTA and existing protocols under event traffic in Indriya

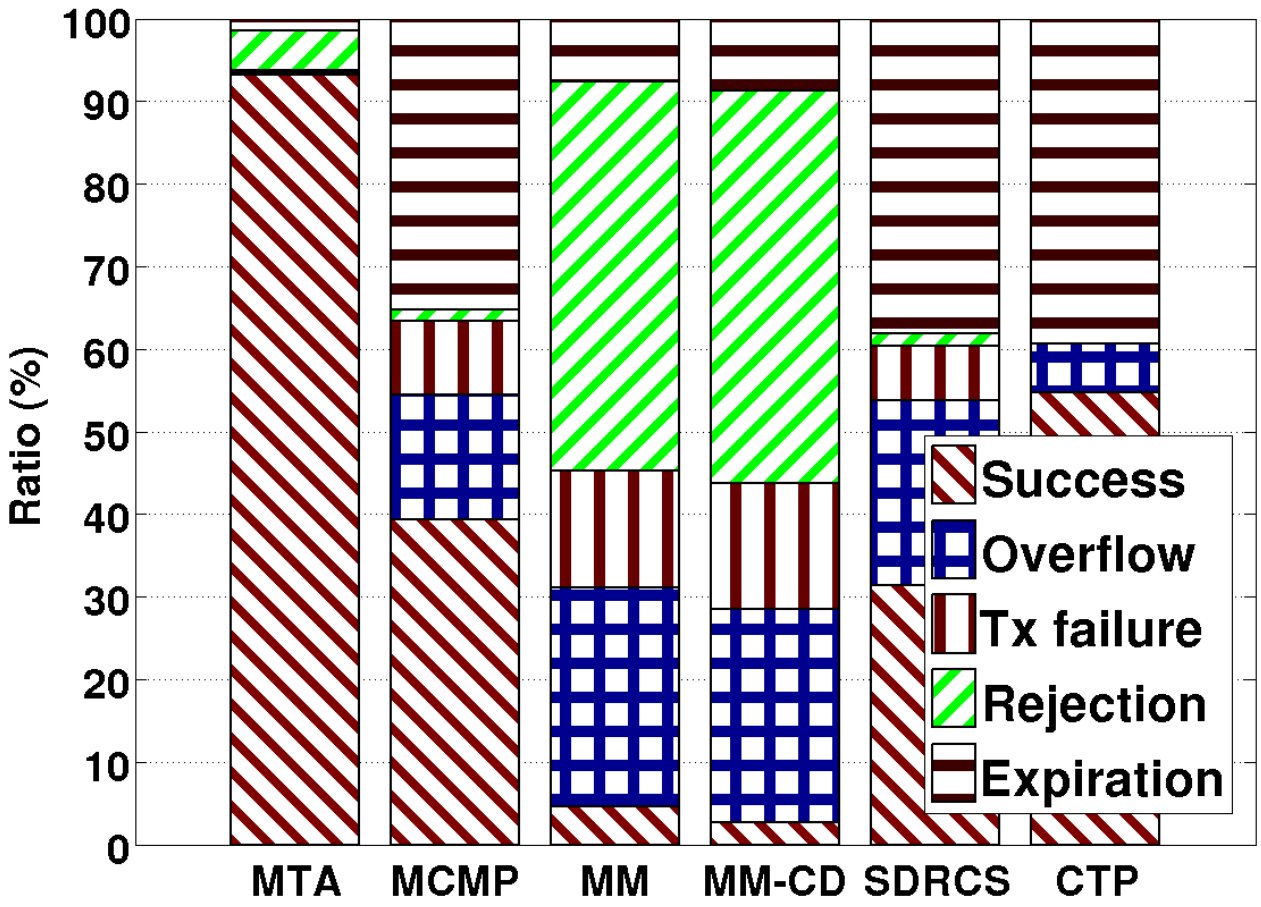

Figure 63: Packet delivery status: MTA and existing protocols under event traffic in Indriya)

6 Discussion: sparse networks

To evaluate the effect of network connectivity on the performance of MTA, we also run it and other routing protocols in sparse networks in NetEye and Indriya.

- *NetEye*: In the 15 by 7 grid, we randomly select 50% of the nodes. Figure 65 shows the number of neighbors each node has in the resulting topology, contrasting Figure 64 for randomly selecting 80% of the grid. Obviously, the former with median of 9 has a lower density than the latter with median of 12. We choose the farthest four nodes, namely node 61, 62, 79, and 80 as sources, and still choose node 15 as the sink. Every source generates a packet every 100 ms with deadline 4 seconds and required probability 90%.

  Figures 68 shows the deadline success ratio in the sparse network. Again, MTA significantly outperforms other protocols by constantly meeting deadline catch ratio requirement. It also maintains better packet delivery ratio and transmission cost than other protocols except CTP, whose performance is comparable to MTA's as shown in Figure 69 and 70. Figure 71 shows the packet delivery status. MCMP, MMSPEED, MMSPEED and SDRCS experience more queueing overflows because the network's capacity is reduced when the node density decreases. The fact that MTA and CTP do not suffer from queueing overflow can be attributed to their ability to identify good paths to compensate for reduced capacity.

- *Indriya*: We reduce the transmission power level from 11 to 6. Figure 66 and 67 show the number of neighbors each node has in the resulting topology for power level 11 and 6, respectively. The latter with median of 5 is clearly sparser than the former with median of 8.5. We choose node 1, 2, 3, and 4 as sources and still node 100 as the sink. Every source generates a packet every 500 ms with deadline 5 seconds and required probability 90%.

  Figures 72, 73, 74, and 75 show the deadline success ratio, packet delivery ratio, transmission cost, and packet delivery status respectively. All the relative performance comparisons are similar to those of power level 11.

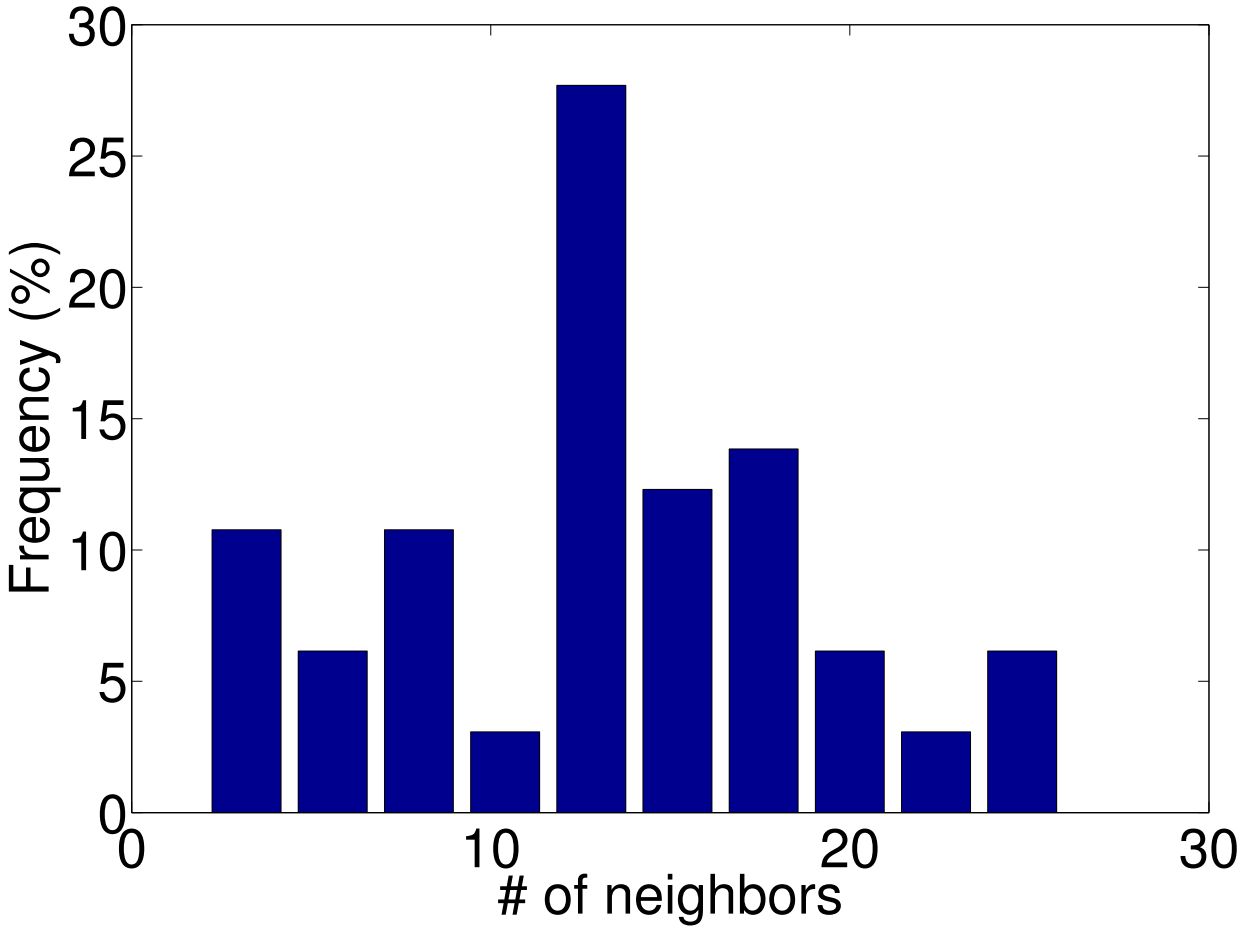

Figure 64: Histogram of the number of neighbors for each node in random 80% NetEye

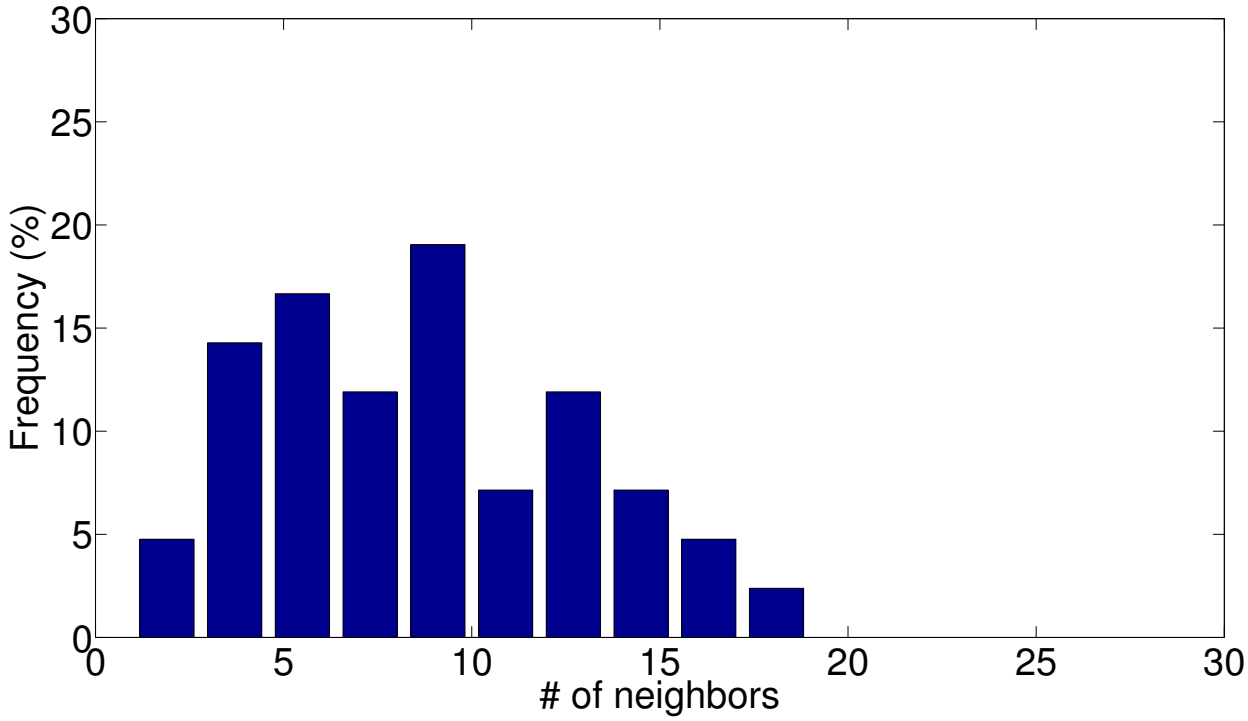

Figure 65: Histogram of the number of neighbors for each node in random 50% NetEye

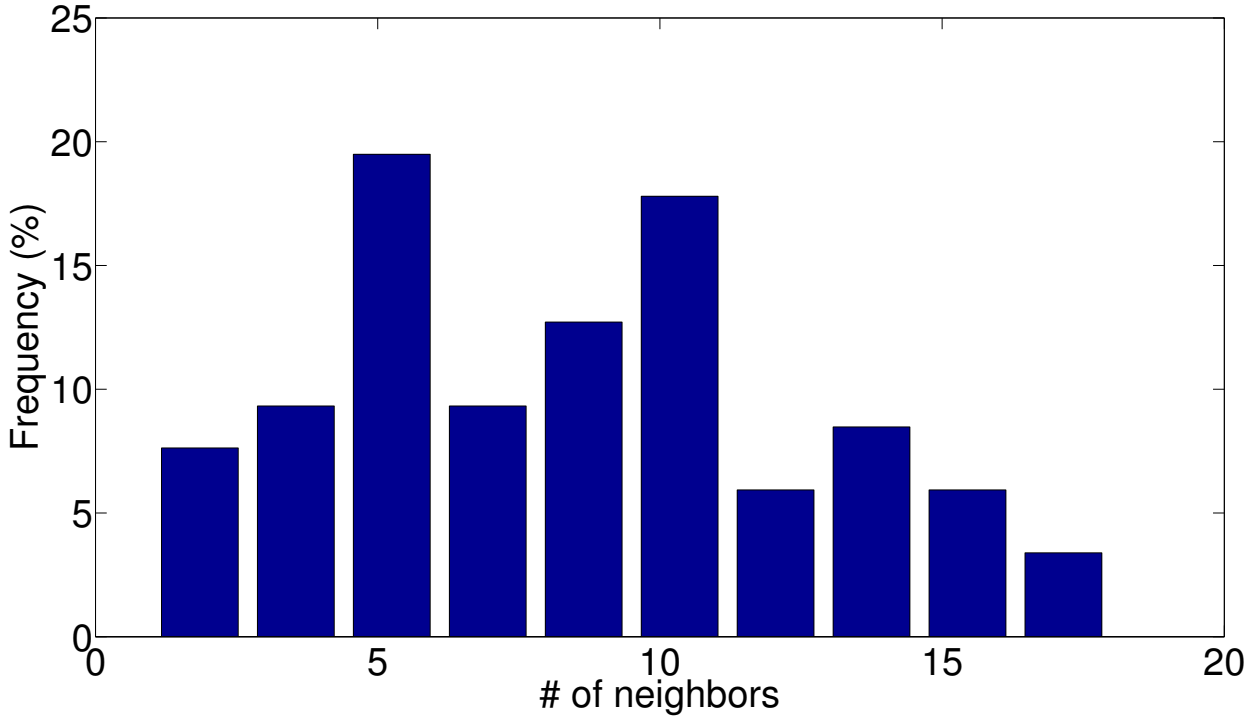

Figure 66: Histogram of the number of neighbors for each node in Indriya using power level 11

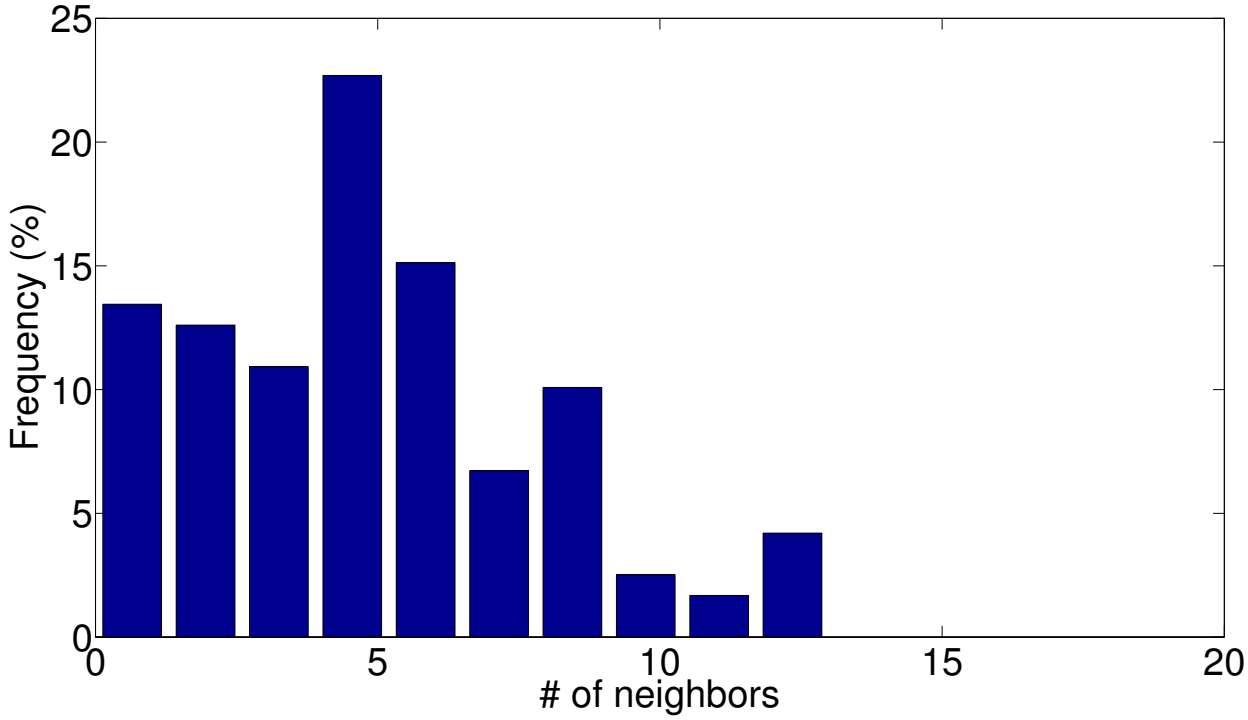

Figure 67: Histogram of the number of neighbors for each node in Indriya using power level 6

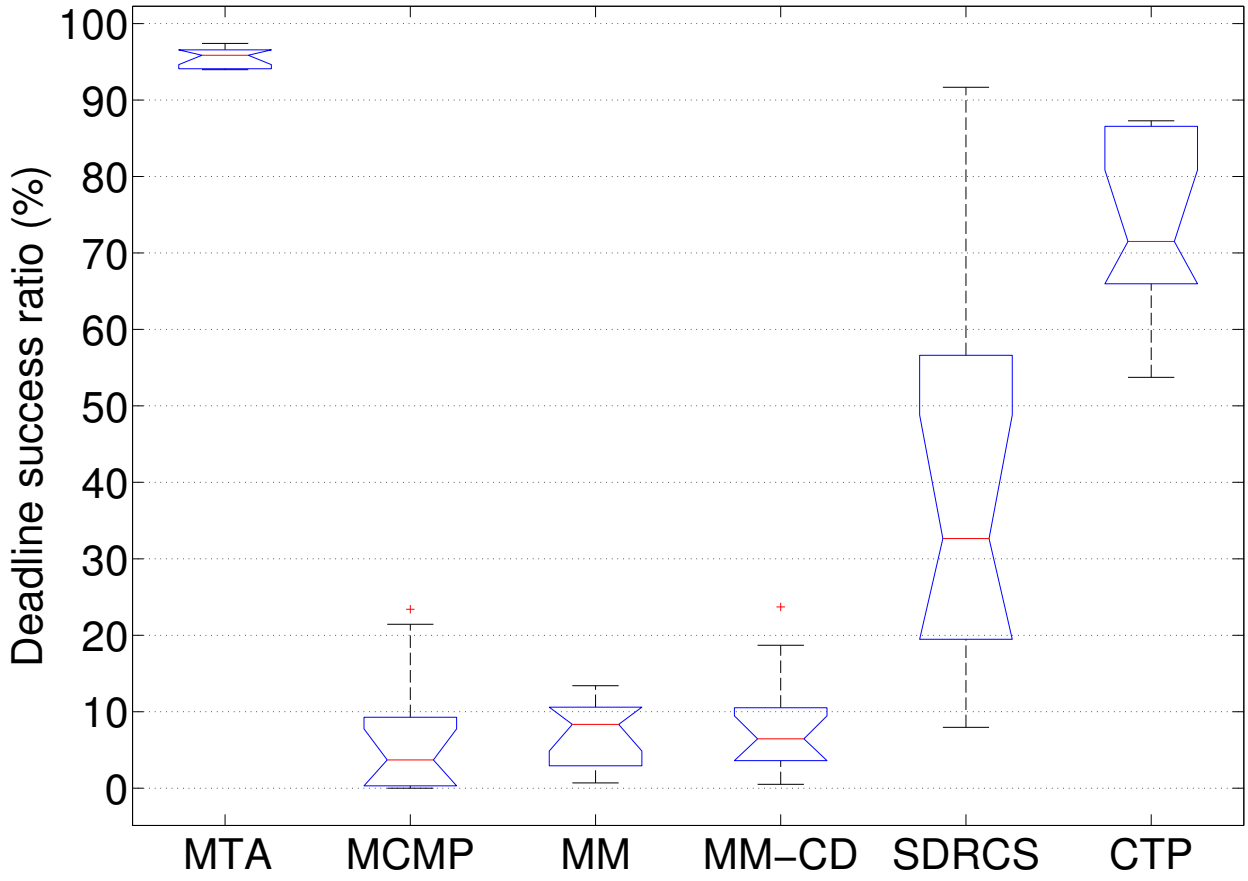

Figure 68: Deadline success ratio of different protocols in sparse NetEye

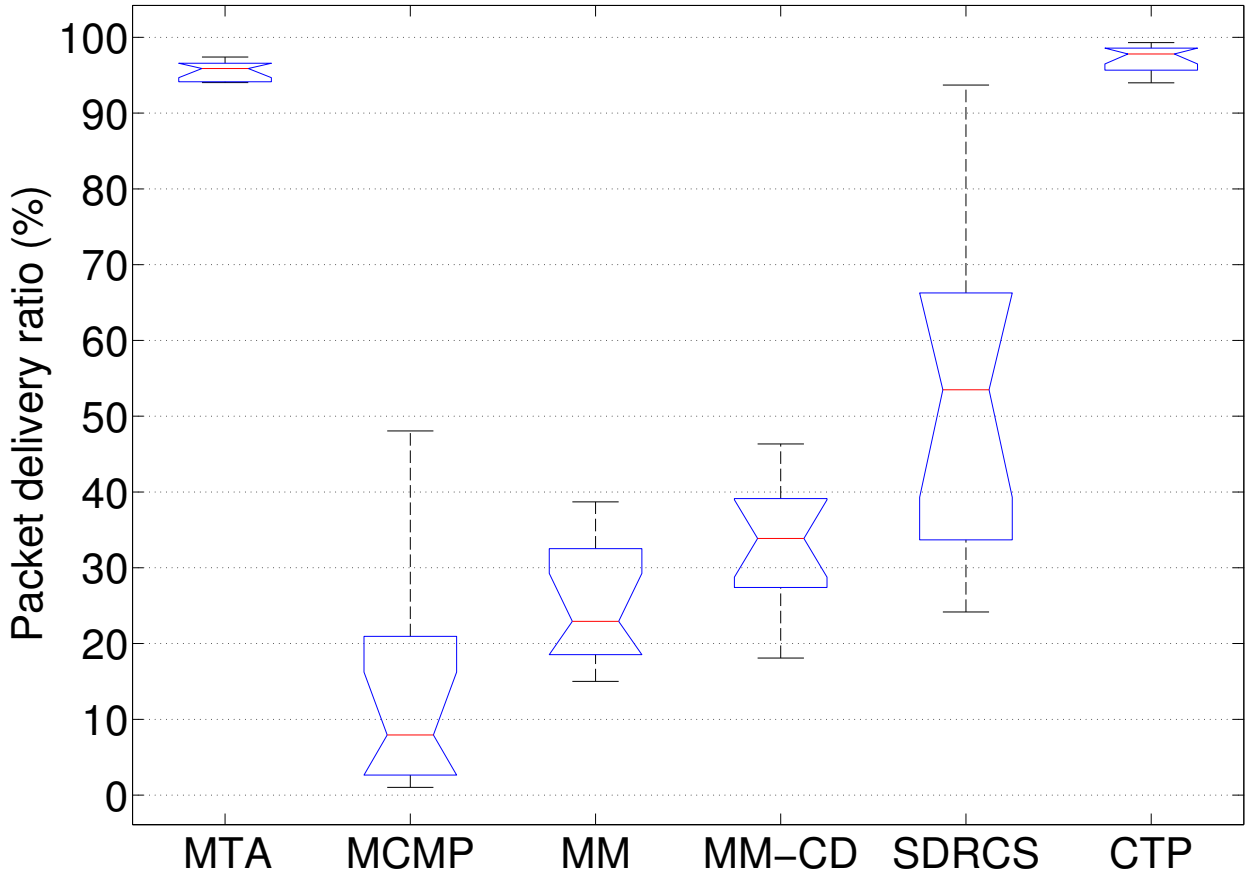

Figure 69: Packet delivery ratio of different protocols in sparse NetEye

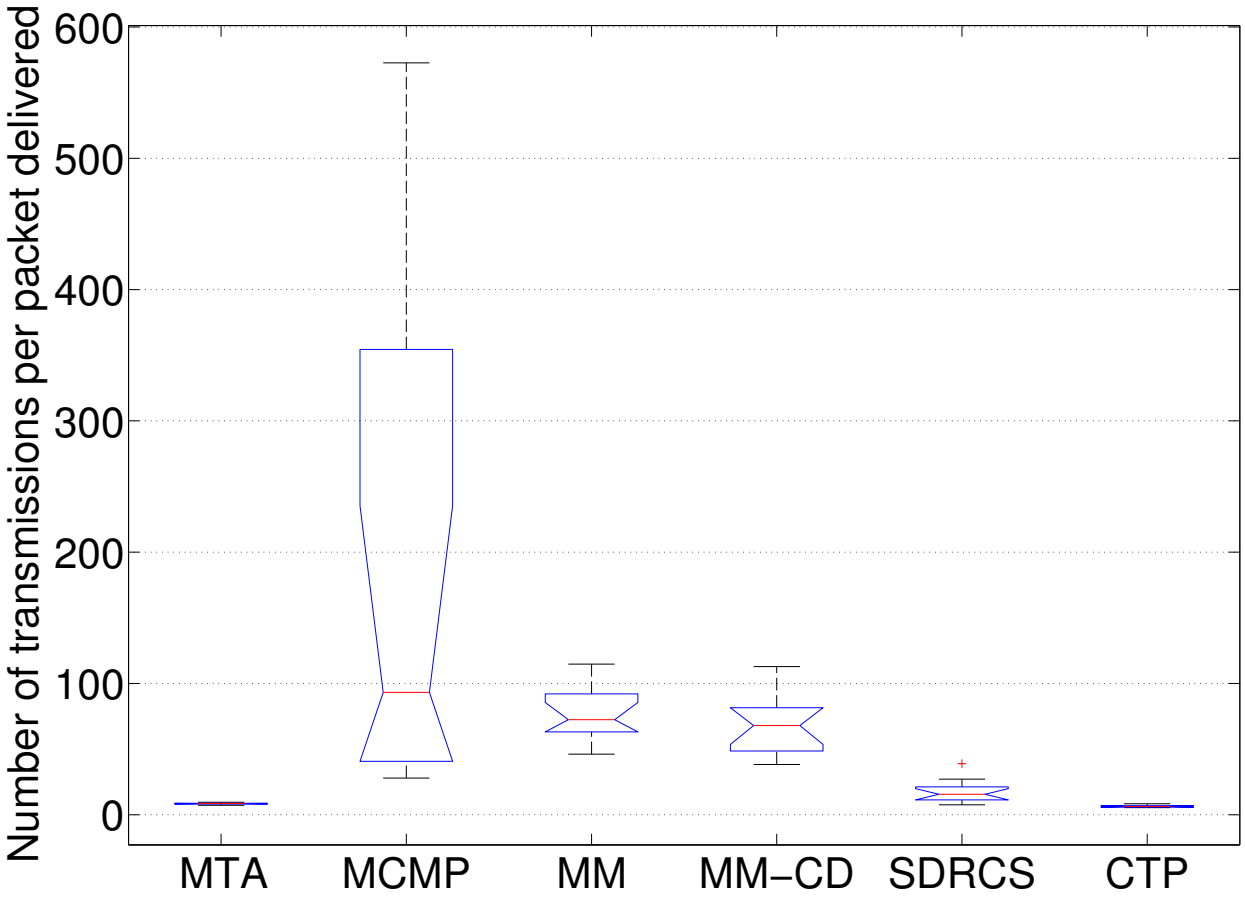

Figure 70: NTX of different protocols in sparse NetEye

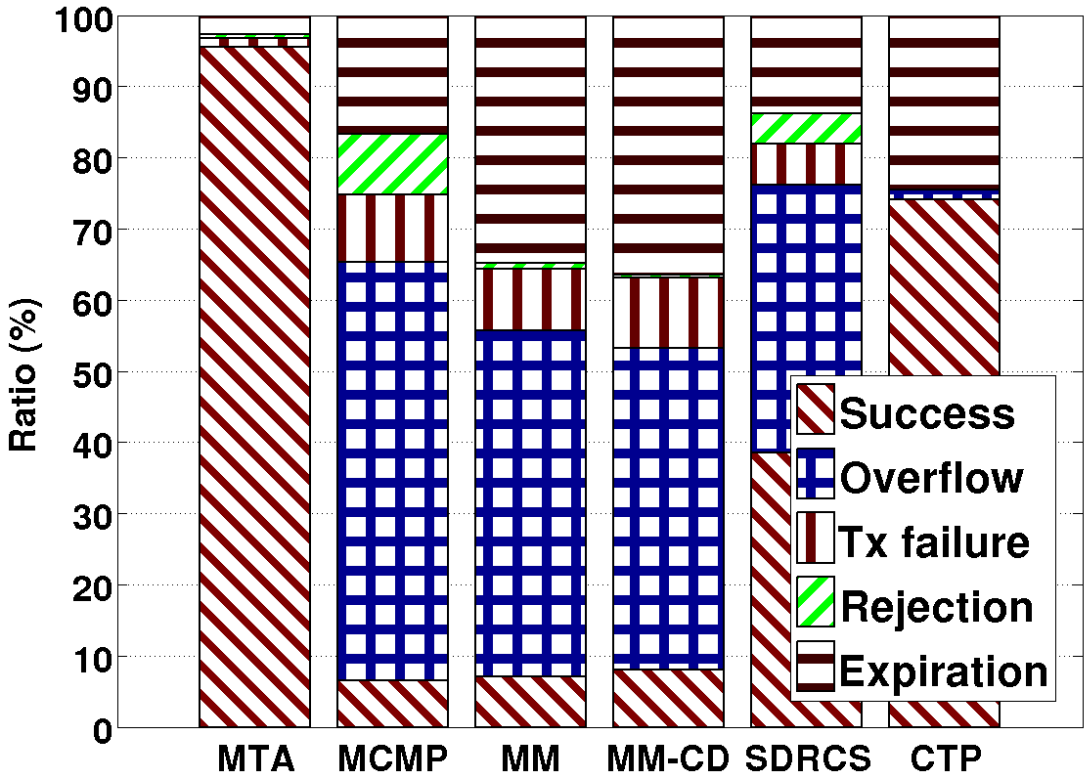

Figure 71: Packet outcomes of different protocols in sparse NetEye

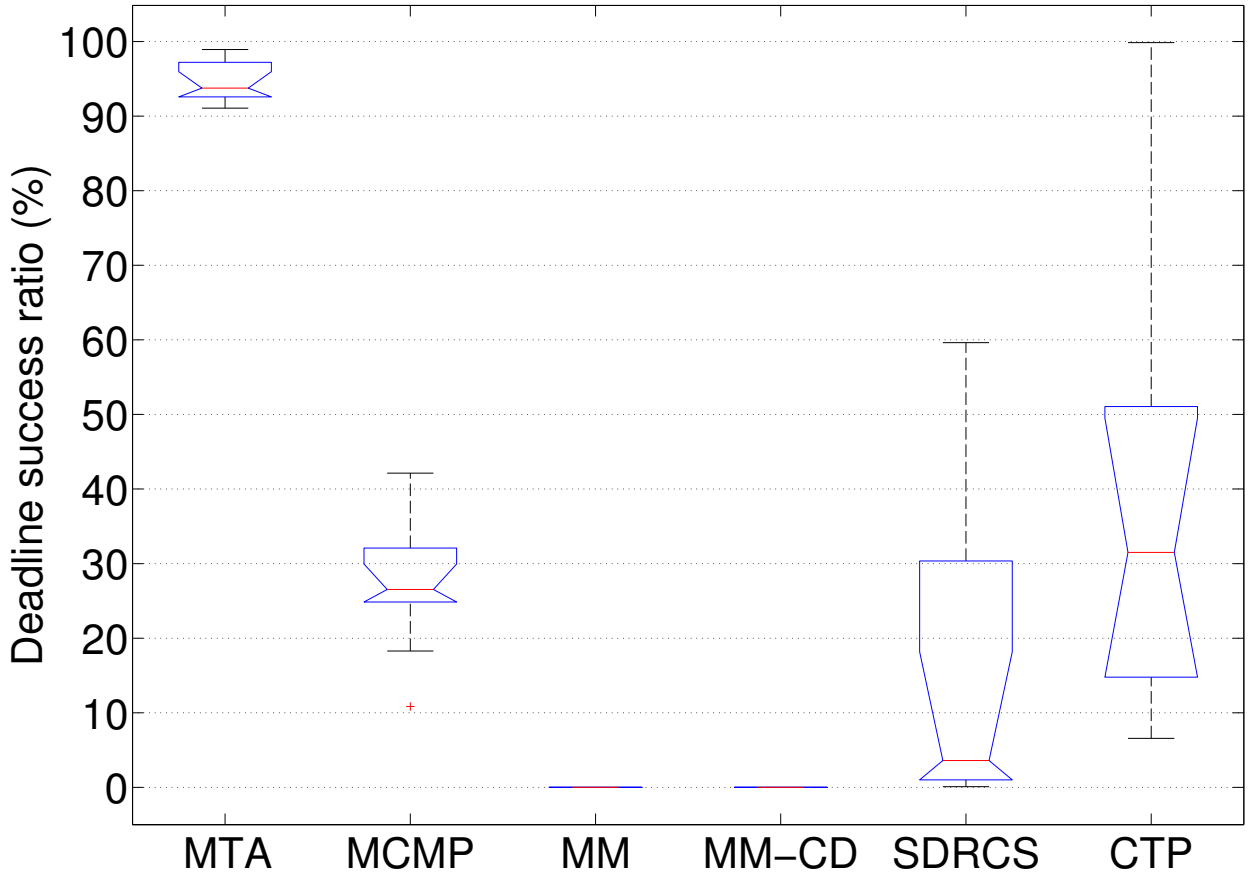

Figure 72: Deadline success ratio of different protocols in sparse Indriya

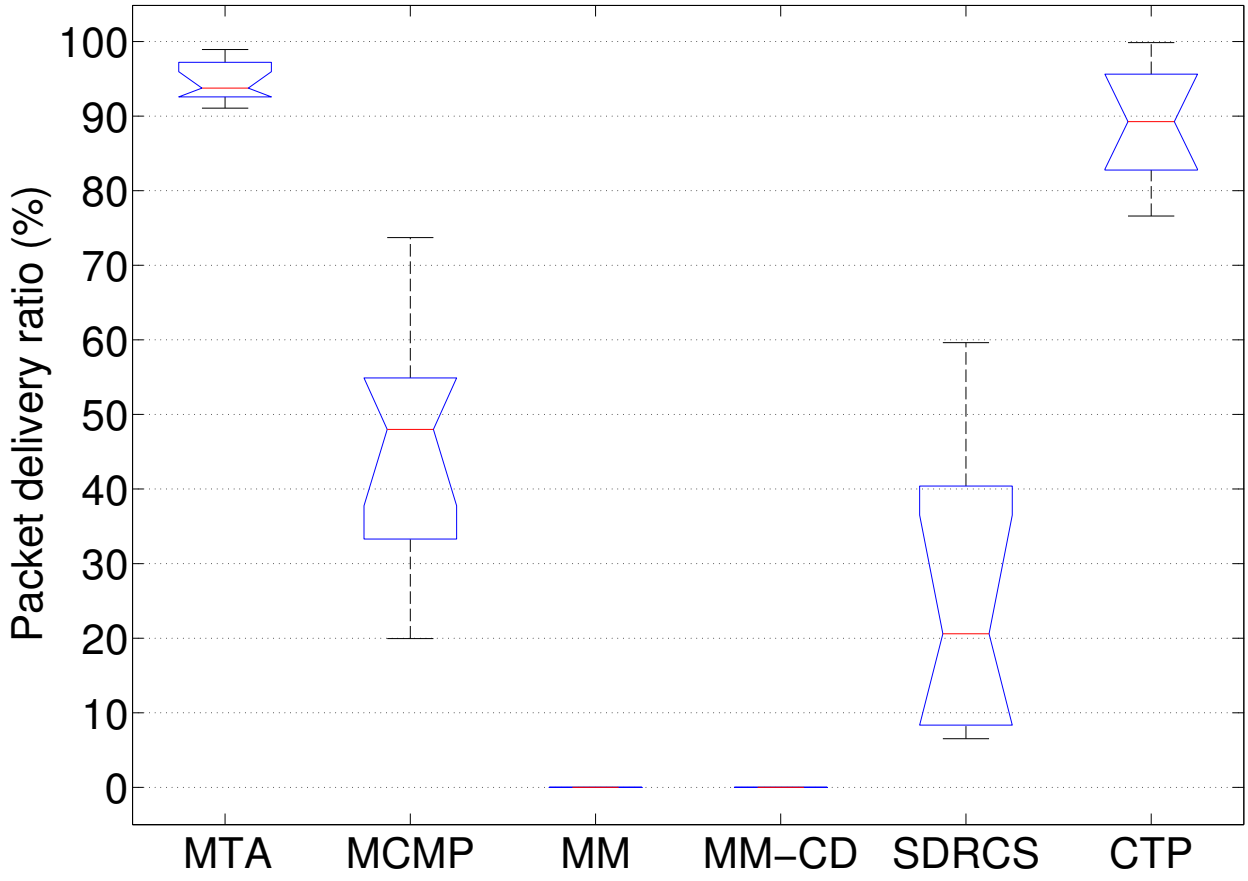

Figure 73: Packet delivery ratio of different protocols in sparse Indriya

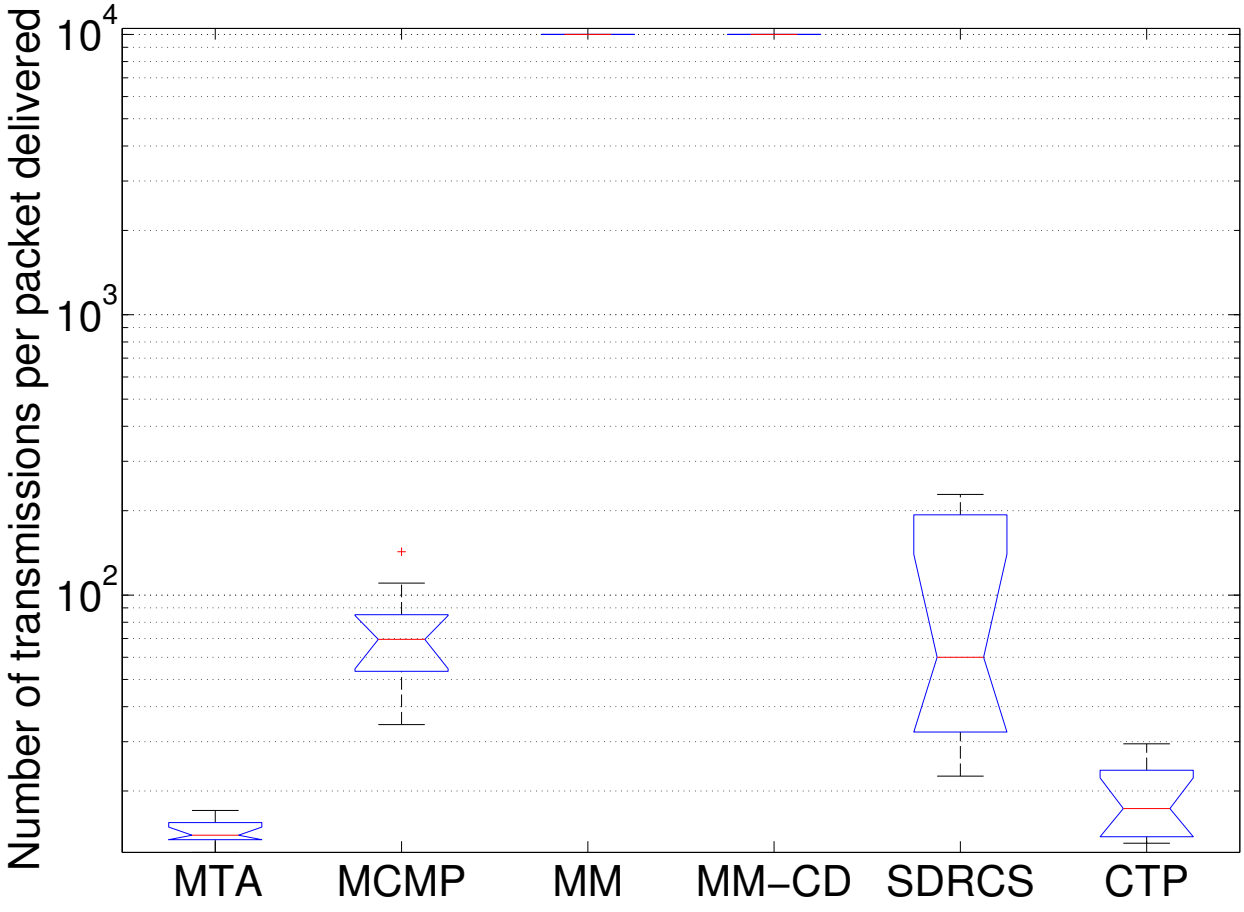

Figure 74: NTX of different protocols under periodic traffic in sparse Indriya

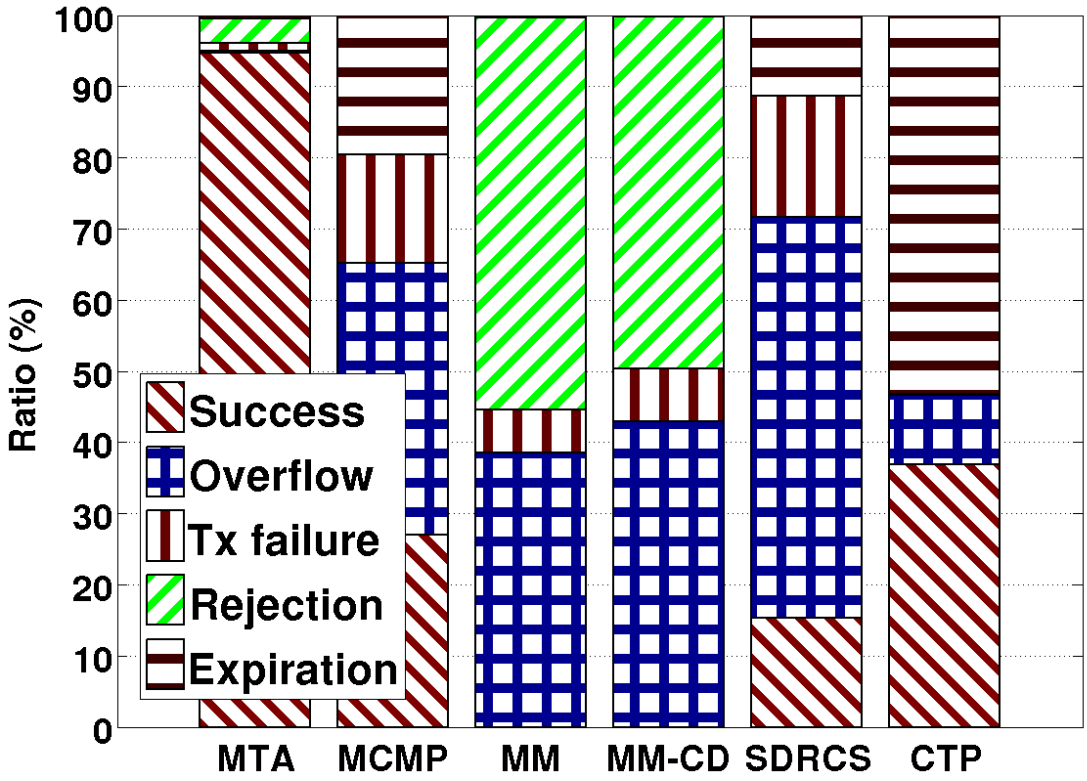

Figure 75: Packet outcomes of different protocols in sparse Indriya

7 Related work

QoS routing has been well studied for the Internet [9, 13] and wireless networks [21, 10, 12]. But most did not consider uncertainties in link/path properties (e.g., delay). Link property uncertainties were considered in [25] and [34], and it was shown that the problem of checking probabilistically guaranteed path delays is NP-hard [25]. Focusing on Internet QoS routing, these work assumed link-state routing, and their solutions were not amenable to light-weight, distance-vector-type implementation. Since link-state routing is usually not suitable for dynamic, resource constrained WSC networks where reliable network-wide link-state update itself is a challenging issue and where nodes may only have very limited memory space (e.g., up to 4KB of RAM), the approaches of [25] and [34] are not applicable to WSC networks.

Data delivery delay was also considered in wireless and sensor network routing [22, 26, 29, 36, 52]; but they only tried to minimize average path delay without ensuring probabilistic delay bounds [51, 22, 26, 36, 52], they did not consider the probabilistic nature of link/path delays [43], they were based on geographic forwarding without addressing network non-uniformity and wireless communication irregularity [26, 22], or they uniformly partitioned multi-hop QoS requirements (e.g., reliability and timeliness) along the links of a path without considering network non-uniformity [51, 29]. Huang et al. [29] used Chebyshev Inequality in single-hop real-time satisfiability testing, but they did not address the challenges of accurate, agile estimation of multi-hop probabilistic delay bounds, and they did not comparatively study Chebyshev Inequality with other well-known probability inequalities. Liu et al. [38] proposed the pseudo-polynomial time algorithm optimal-partition-minimum-delay (OPMD) for upper-bounding probabilistic path delays, but, as we have shown in Chapter 3, the bound of the OPMD algorithm is quite loose for multi-hop paths.

Multi-timescale adaptation has been considered in Internet traffic engineering [8, 33]. Focusing on load balancing, these work did not consider QoS assurance. Liu et al. [38] also studied multi-timescale adaptation in routing, but they used the OPMD method to estimate probabilistic delay bounds which are significantly looser than the bound identified through our MTE method, thus leading to real-time capacity loss. The IETF ROLL working group [30] considered building

routing trees based on directed-acyclic-graphs (DAG) for low-power wireless networks. Serving as a general reference framework, the ROLL routing proposal did not consider specific optimization methods (e.g., for real-time guarantees). For stable data delivery reliability, Lin et al. [37] proposed to route data based on long-term link properties and to address transient perturbations using power and retransmission control; they focused on data delivery reliability instead of real-time, thus they did not consider the challenges of dynamic, uncertain link/path delays in real-time routing.

The WirelessHART [11] and the ISA SP100.11a [45] standards have been recently proposed for wireless networking in industrial process measurement and control. They mostly focus on high-level system frameworks instead of specific algorithms in real-time routing. In the literature of real-time wireless networking, techniques such as power control [14] as well as joint routing and scheduling [43] have been studied; energy-efficiency [18] has also been considered too. Orthogonal to these studies, our study here has focused on addressing the challenges that dynamic, uncertain link/path delays pose to two basic elements of real-time routing, i.e., determining probabilistic path delays and addressing instability in delay-adaptive routing. Integrating our results with those work will be an interesting research avenue to pursue, but detailed study of it is beyond the scope of this paper.

8 Concluding remarks

For addressing the challenges of highly-varying path delays to distributed estimation of path delay quantiles, we have proposed the MTE method that leverages the stability of packet-time distribution and the quick diffusion of path delay statistics (i.e., mean and variance) to accurately estimate probabilistic path delay bounds in an agile manner. Based on accurate, agile characterization of path delays using MTE, our MTA routing framework enables the stability and optimality of data forwarding while adapting to fast-changing network queueing and delay. Through extensive measurement study in both the NetEye and the Indriya wireless sensor network testbeds, we have shown that MTE/MTA-based routing ensures efficient, real-time data delivery, and it significantly outperforms existing real-time routing protocols. We have mainly focused on real-time spatial flow control in this study, even though we have experimentally analyzed the benefits of using EDF instead of FCFS in intra-node scheduling; how to control temporal packet flow between neighbors and across the network and how to jointly optimize the spatial and temporal packet flow will be an important area to explore, where the MTE method and the MTA framework are expected to serve as basic systems building-blocks. The technique of leveraging different timescales of dynamics in protocol design may well be of generic interest to wireless networking in dynamic, uncertain environments too.

REFERENCES

[1] An event traffic trace for sensor networks. `http://www.cs.wayne.edu/~hzhang/group/publications/Lites-trace.txt`.

[2] Indriya testbed. `http://indriya.comp.nus.edu.sg/`.

[3] IEEE 802.15 smart utility networks task group 4g. `http://www.ieee802.org/15/pub/TG4g.html`.

[4] IEEE 802.15.4e Working Group. `http://www.ieee802.org/15/pub/TG4e.html`.

[5] TinyOS TEP 133: Packet-level time synchronization. http://www.tinyos.net/tinyos-2.x/doc/html/tep133.html.

[6] S. Bapat, V. Kulathumani, and A. Arora. Analyzing the yield of exscal, a large-scale wireless sensor network experiment. In *IEEE ICNP*, pages 53–62, 2005.

[7] S. Biswas and R. Morris. ExOR: Opportunistic multi-hop routing for wireless networks. In *ACM SIGCOMM*, 2005.

[8] M. Caesar, M. Casado, T. Koponen, J. Rexford, and S. Shenker. Dynamic route computation considered harmful. *ACM SIGCOMM Computer Communication Review*, 40(2):66–71, 2010.

[9] A. Chakrabarti and G. Manimaran. Reliability constrained routing in QoS networks. *IEEE/ACM Transactions on Networking*, 13(3):662–675, 2005.

[10] S. Chakrabarti and A. Mishra. QoS issues in ad hoc wireless networks. *IEEE Communications Magazine*, 39(2):142–148, February 2001.

[11] D. Chen, M. Nixon, and A. Mok. *Wireless*HART*: Real-Time Mesh Network for Industrial Automation*. Springer, 2010.

[12] L. Chen and W. B. Heinzelman. QoS-aware routing based on bandwidth estimation for mobile ad hoc networks. *IEEE JSAC*, 23(3):561–571, 2005.

[13] S. Chen, M. Song, and S. Sahni. Two techniques for fast computation of constrained shortest paths. *IEEE/ACM Transactions on Networking*, 16(1):105–115, February 2008.

[14] O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J. Stankovic, and T. Abdelzaher. Real-time power-aware routing in sensor networks. In *IWQoS*, 2006.

[15] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *ACM MobiCom*, 2003.

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

[17] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *ACM MobiCom*, 2004.

[18] S. C. Ergen and P. Varaiya. Energy efficient routing with delay guarantee for sensor networks. *Wireless Networks*, 13(5):679–690, 2007.

[19] M. Erol-Kantarci and H. T. Mouftah. Wireless sensor networks for cost-efficient residential energy management in the smart grid. *IEEE Transactions on Smart Grid*, 2(2):314–325, 2011.

[20] Y. P. Fallah, C. Huang, R. Sengupta, and H. Krishnan. Design of cooperative vehicle safety systems based on tight coupling of communication, computing and physical vehicle dynamics. In *ACM/IEEE ICCPS*, 2010.

[21] X. Fang, D. Yang, P. Gundecha, and G. Xue. Multi-constrained anypath routing in wireless mesh networks. In *IEEE SECON*, 2010.

[22] E. Felemban, C.-G. Lee, E. Ekici, R. Boder, and S. Vural. Probabilistic QoS guarantee in reliability and timeliness domains in wireless sensor networks. In *IEEE INFOCOM*, 2005.

[23] H. Gharavi and B. Hu. Multigate communication network for smart grid. *Proceedings of the IEEE*, 99(6):1028–1045, 2011.

[24] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *ACM SenSys*, 2009.

[25] R. A. Guerin and A. Orda. QoS routing in networks with inaccurate information: Theory and algorithms. *IEEE/ACM Transactions on Networking*, 7(3):350–364, 1999.

[26] T. He, J. Stankovic, C. Lu, and T. Abdelzaher. SPEED: A stateless protocol for real-time communication in sensor networks. In *IEEE ICDCS*, 2003.

[27] J. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury. *Feedback Control of Computing Systems*. Wiley-IEEE Press, 2004.

[28] B. Hobijn, P. H. Franses, and M. Ooms. Generalizations of the KPSS-test for stationarity. *Statistica Neerlandica*, 58(4):483–502, 2004.

[29] X. Huang and Y. Fang. Multiconstrained QoS multipath routing in wireless sensor networks. *Wireless Networks*, 14:465–478, 2008.

[30] IETF. Routing over low power and lossy networks (ROLL) working group. `http://www.ietf.org/html.charters/roll-charter.html`.

[31] R. Jain and I. Chlamtac. The P^2 algorithm for dynamic calculation of quantiles and histograms without storing observations. *Communications of the ACM*, 28(10):1076 – 1085, 1985.

[32] X. Ju, H. Zhang, and D. Sakamuri. NetEye: A user-centered wireless sensor network testbed for high-fidelity, robust experimentation. *International Journal of Communication Systems (Wiley)*, 25(9):1213–1229, 2012.

[33] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *ACM SIGCOMM*, 2005.

[34] T. Korkmaz and M. Krunz. Bandwidth-delay constrained path selection under inaccurate state information. *IEEE/ACM Transactions on Networking*, 11(3):384 – 398, 2003.

[35] A. LaJoie. Wireless sensors enjoy timely role in factory automation. *Industrial Ethernet Book*, April 2010.

[36] H. Li, Y. Cheng, and C. Zhou. Minimizing end-to-end delay: A novel routing metric for multi-radio wireless mesh networks. In *IEEE INFOCOM*, 2009.

[37] S. Lin, G. Zhou, K. Whitehouse, Y. Wu, J. A. Stankovic, and T. He. Towards stable network performance in wireless sensor networks. In *IEEE RTSS*, 2009.

[38] X. Liu, H. Zhang, and Q. Xiang. Towards predictable real-time routing for wireless networked sensing and control. In *CPS Week RealWin workshop*, 2011.

[39] X. Liu, H. Zhang, Q. Xiang, X. Che, and X. Ju. Taming uncertainties in real-time routing for wireless networked sensing and control. Technical Report DNC-TR-11-04 (`https://sites.google.com/site/dnctrs/DNC-TR-11-04.pdf`), Wayne State University, 2011.

[40] M. Loeve. *Probability Theory*. Springer-Verlag, 1977.

[41] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *ACM SenSys*, 2004.

[42] J. R. Moyne and D. M. Tilbury. Control and communication challenges in networked real-time systems. *Proceedings of the IEEE*, 95(1):9–28, 2007.

[43] S. Munir, S. Lin, E. Hoque, S. M. S. Nirjon, J. A. Stankovic, and K. Whitehouse. Addressing burstiness for reliable communication and latency bound generation in wireless sensor networks. In *IEEE/ACM IPSN*, 2010.

[44] S. M. Ross. *Introduction to Probability Models*. Academic Press, 2006.

[45] ISA SP100.11a. http://www.isa.org//MSTemplate.cfm?MicrositeID=1134&CommitteeID=6891.

[46] H.-M. Tsai, C. Saraydar, T. Talty, M. Ames, A. Macdonald, and O. K. Tonguz. Zigbee-based intra-car wireless sensor network. In *IEEE ICC*, 2007.

[47] J. V. Uspensky. *Introduction to Mathematical Probability*. McGraw-Hill Book Company Inc., 1937.

[48] L. Wang and G. Yin. Quantized identification under dependent noise and fisher information ratio for communication channels. *IEEE Transactions on Automatic Control*, 55:674–690, 2010.

[49] W. Wang, Y. Xu, and M. Khanna. A survey on the communication architectures in smart grid. *Computer Networks*, 55:3604–3629, 2011.

[50] A. Willig. Recent and emerging topics in wireless industrial communications: A selection. *IEEE Transactions on Industrial Informatics*, 4(2):102–124, May 2008.

[51] Y. Xue, B. Ramamurthy, and M. C. Vuran. SDRCS: A service-differentiated real-time communication scheme for event sensing in wireless sensor networks. *Computer Networks*, 55:3287–3302, 2011.

[52] S. Yin, Y. Xiong, Q. Zhang, and X. Lin. Traffic-aware routing for real-time communications in wireless multi-hop networks. *Wireless Communications and Mobile Computing*, 6:825–843, 2006.

[53] F. R. Yu, P. Zhang, W. Xiao, and P. Choudhury. Communication systems for grid integration of renewable energy resources. *IEEE Network*, 25(5):22–29, September/October 2011.

[54] H. Zhang, A. Arora, Y. ri Choi, and M. Gouda. Reliable bursty convergecast in wireless sensor networks. *Computer Communications (Elsevier)*, 30(13), 2007.

[55] H. Zhang, A. Arora, and P. Sinha. Link estimation and routing in sensor network backbones: Beacon-based or data-driven? *IEEE Transactions on Mobile Computing*, 8(5):653 – 667, May 2009.

[56] H. Zhang, L. Sang, and A. Arora. On the convergence and stability of data-driven link estimation and routing in sensor networks. *ACM Transactions on Autonomous and Adaptive Systems*, 4(3):18:1–29, July 2009.

[57] H. Zhang, L. Sang, and A. Arora. Comparison of data-drive link estimation methods in low-power wireless networks. *IEEE Transactions on Mobile Computing*, 9(11):1634–1648, November 2010.

[58] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *ACM SenSys*, 2003.

ABSTRACT

TAMING UNCERTAINTIES IN REAL-TIME ROUTING FOR WIRELESS NETWORKED SENSING AND CONTROL

by

Xiaohui Liu

August 2014

Advisor: Dr. Hongwei Zhang

Major: Computer Science

Degree: Master of Science

Real-time routing is a basic element of closed-loop, real-time sensing and control, but it is challenging due to dynamic, uncertain link/path delays. The probabilistic nature of link/path delays makes the basic problem of computing the probabilistic distribution of path delays NP-hard, yet quantifying probabilistic path delays is a basic element of real-time routing and may well have to be executed by resource-constrained devices in a distributed manner; the highly-varying nature of link/path delays makes it necessary to adapt to in-situ delay conditions in real-time routing, but it has been observed that delay-based routing can lead to instability, estimation error, and low data delivery performance in general. To address these challenges, we propose the *Multi-Timescale Estimation (MTE)* method; by accurately estimating the mean and variance of per-packet transmission time and by adapting to fast-varying queueing in an accurate, agile manner, MTE enables accurate, agile, and efficient estimation of probabilistic path delay bounds in a distributed manner. Based on MTE, we propose the *Multi-Timescale Adaptation (MTA)* routing protocol; MTA integrates the stability of an ETX-based directed-acyclic-graph (DAG) with the agility of spatiotemporal data flow control within the DAG to ensure real-time data delivery in the presence of dynamics and uncertainties. We also address the challenges of implementing MTE and MTA in resource-constrained devices such as TelosB motes. We evaluate the performance of MTA using the NetEye and Indriya sensor network testbeds. We find that MTA significantly outperforms

existing protocols, e.g., improving deadline success ratio by 89% and reducing transmission cost by a factor of 9.7.

AUTOBIGRAPHICAL STATEMENT

Xiaohui Liu is a Ph. D. student in the Computer Science Department of Wayne State University. He is a member of the Dependable Networking and Computing Group, led by Dr. Hongwei Zhang. His primary interest lies in predictable and reliable communication for wireless and sensor networks, especially with real-time networking for sensing and control. He works at the confluence of theory and systems, designing MAC, routing, and cross-layer protocols with theoretical foundation as well as empirical considerations. Prior to joining Wayne State University, he received his bachelor degree in Computer Science from Wuhan University, China. Xiaohui has published in premier journals and conferences such as IEEE Transactions on Smart Grid, the ACM Transactions on Sensor Networks, IEEE Transactions on Mobile Computing, IEEE Real-Time Systems Symposium, The ACM International Symposium on Mobile Ad Hoc Networking and Computing, and IEEE International Conference on Sensing, Communication, and Networking. In addition, he has led a student team to become one of the 15 US finalists (out of several thousand teams nationwide) in the 2009 Microsoft Imagine Cup software design competition. He is a recipient of the Outstanding Graduate Research Assistant Award in Computer Science Department at Wayne State University. He is a student member of ACM. More detailed information about Xiaohui can be found from his website at http://www.cs.wayne.edu/xliu.