Cross-layer Optimization Made Practical

Ajit Warrier, Long Le and Injong Rhee
Department of Computer Science,
North Carolina State University.

***Abstract*— Limited resources and time-varying nature of wireless ad hoc networks demand optimized use of resources across layers. Cross-layer optimization (CLO) for wireless networks, an approach that coordinates protocol behaviors at different layers with a goal to maximize a utility function, has received considerable attention lately. However, most existing work remains as theory and no practical CLO based on utility optimization exists today. The main difficulties in implementing theoretical CLO designs arise often from impractical assumptions about the characteristics of the wireless medium and also from computational and communication overhead of proposed solutions to achieve or approximate the optimality. In contrast, many existing practical approaches for CLO (not necessarily utility optimization) are rather ad hoc in nature and developed mostly based on intuitions. Thus, a clear gap between theory and practice in CLO exists. This paper addresses this dichotomy to close the gap by taking an optimal solution from utility-based CLO and applying practical approximation to enable a practical implementation in a wireless mesh network where nodes are statically positioned in an ad hoc fashion. We focus on the utility of maximizing throughput. We identify the impractical or computationally-intensive components of a theoretically-derived optimal throughput-maximizing solution and then propose, in most cases, practical approximation with O(1) complexity for MAC, scheduling, routing and congestion control. The result is a practical CLO solution that approximates the theoretically-derived optimal solution, but achieves much improved performance over existing practical CLO implementations.**

I. Introduction

In recent years, considerable interest has been paid to wireless ad hoc networks. The main advantage of ad hoc networks is that they can be deployed easily without an infrastructure. However, the advantages and opportunities of ad hoc wireless networks also come with significant technical challenges in the design of these networks.

Due to the characteristics and challenges of ad hoc wireless networks such as intermediate bit error rates and interferences between nodes [1], performance of network protocols such as TCP that were originally designed for wired networks degrade significantly on ad hoc wireless networks [9]. For this reason, many have argued that the characteristics and challenges of ad hoc wireless networks call for a new paradigm of protocol designs where protocols are jointly designed and optimized across different layers [3], [4], [5], [6]. Examples of CLO designs are TCP enhancements for wireless networks. A number of CLO designs allow TCP to differentiate between causes of packet losses (e.g. between congestion and link failures) and take appropriate reactions [11], [20]. Other CLO designs either modify TCP or the MAC layer to reduce packet collisions and improve TCP throughput [8], [9], [16], [19].

Existing work in CLO designs essentially falls into two categories: theoretical approaches [3], [4], [5], [14] and practical approaches [12], [16], [19]. The theoretical approaches are provably optimal, i.e., they can deliver the optimal performance and maximize a utility function (e.g. end-to-end throughput). However, since these approaches usually have certain theoretical assumptions, they are rather impractical and difficult to implement in practice. On the other hand, practical approaches are relatively easy to implement but they are rather ad hoc in nature and mostly developed based on intuitions. Since practical approaches are not derived within a theoretical framework, it is hard to understand their performance analytically and to determine their performance bounds (i.e., whether the bounds of performance for practical solutions have been reached). Further, great cautions need to be exercised in practical approaches that are designed without an analytical analysis (such as stability and convergence) to avoid undesirable protocol behaviors and unintended or unforeseen adverse interactions between protocol layers [13].

As the networking research community moves forward, we believe that there are two alternate paths. (1) We can continue to fine-tune practical approaches until a point of diminishing return is reached (since we do not know the performance bounds). (2) Theoreticians and practitioners can come together and make theoretical approaches practical by replacing the theoretical assumptions with practical approximations. We follow the second path in this work and attempt to bridge the gap between theory and practice by making a theoretical CLO design practical. This is accomplished by replacing the assumptions of a theoretical CLO with practical approximations. Since the practical approximations achieve the intended purposes of the theoretical assumptions, our practical CLO enjoys the same analytical properties of its theoretical counterpart. The result is a practical solution that approximates the theoretically-derived optimal solution but obtains much improved performance over existing practical implementations.

The contributions of our paper are: (1) We present a practical and integrated CLO design for MAC, routing, scheduling, and congestion control. Unlike previous CLO designs that focused on specific problems [2], [7], [20], our solution is an integrated CLO across the aforementioned protocol layers to improve overall system performance. Further, to the best of our knowledge, our work is the first practical CLO design that is derived from the optimization framework for CLO design. (2) We designed and implemented ZMAC+ as an alternate mechanism to IEEE 802.11e [18] for MAC prioritization. MAC prioritization is important in a CLO design because it allows congested nodes to receive prioritized access to the wireless channel and drain their packet queues.

The rest of the paper is organized as follows. Section II gives a survey of CLO designs in wireless networks. Section III reviews the theoretical CLO framework that our design is based on and points out its impractical assumptions. Section IV presents a practical CLO design that approximates a theoretical CLO by using practical approximations. Section V presents a performance evaluation of our CLO design and compares it with existing practical CLO approaches. Section VI summarizes and concludes our paper.

II. Related Work

A number of CLO designs have been proposed. However, these designs usually focused on specific problems for ad hoc wireless networks. We will review and discuss existing CLO designs mostly related to our work in this section.

De Couto et al presented the expected transmission count (ETX) as a new routing metric that predicts the number of retransmissions using per-link measurements of packet loss ratios in both directions between nodes [7]. They showed that when routing protocols incorporate the link quality by using ETX, the throughput of multi-hop routes can be improved by up to a factor of two over the traditional minimum hop-count metric. We use ETX as a routing metric for nodes to find paths with good link quality but also incorporate interactions between routing, scheduling, and congestion control to allow nodes to find routes around congested links.

Hull et al presented *Fusion*, a combination of hop-by-hop flow control, rate limiting source traffic when transit traffic exists, and a prioritized MAC to improve network efficiency [12]. Hop-by-hop flow control allows nodes to provide back pressure toward the source and prevents nodes from transmitting packets faster than the downstream nodes can forward them. Rate limiting meters traffic entering the network to prevent unfairness between transit flows and flows originating at a local node. A prioritized MAC allows congested nodes to receive prioritized access to the wireless channel and drain their packet queues. Our CLO design is similar to Fusion in that nodes perform hop-by-hop flow control and rate limiting at the sources. However, our CLO design also incorporates interactions between routing and congestion control to attempt to route packets around congested areas of a network.

Nahm et al investigated the impact of TCP on on-demand routing protocols such as DSR. They showed that the fast growth rate of TCP's congestion window can cause packet losses at the link layer due to MAC contention and the hidden terminal problem [16]. These packet losses are perceived as routing failure by routing protocols such as DSR. This routing failure causes TCP to time out. During the time-out, the routing failure is recovered. The cycle between routing failure and routing recovery continues after the time-out when TCP starts transmitting packets again. Nahm et al proposed TCP fractional window increment (TCP-FeW), a modification of TCP that increases the congestion window at a slower rate to avoid the adverse impact of TCP on routing protocols. We note that TCP-FeW only focuses on the specific problem of congestion control in wireless networks and could benefit from support of MAC and routing protocol.

Chiang proposed a distributed joint algorithm for congestion control and power control in CDMA wireless networks [5]. Chiang's algorithm used a delay-based algorithm similar to TCP Vegas to adapt the source rates. Further, nodes increase their power level proportional to their current shadow price (computed as a function of their queues) and inversely proportional to the current power level. The idea is that if a node has a large queue, it increases its transmit power to improve its signal to interference ratio and thus its transmission rate to drain its queue faster. However, the transmit power should only be increased moderately when the current power level is already high. We note that Chiang's algorithm only applies for CDMA networks and not necessarily applicable for CSMA networks such as prevalent IEEE 802.11 and sensor networks.

Chen et al proposed a joint design of congestion control, routing, and scheduling to achieve high end-to-end throughput and efficient resource utilization [3]. This design is based on the optimization framework and provides provably optimal end-to-end throughput as a utility function. Chen et al decomposed their design into distributed algorithms for congestion control and routing/scheduling that interact through congestion prices. The congestion prices are computed locally at each node based on the difference between bandwidth demand and the capacity of that node. The congestion control algorithm regulates sources' sending rates according to the congestion price. Routing and scheduling are done locally at a node based on back pressure from the differential prices of neighboring nodes as follows. Each node collects congestion prices from its neighbors, computes the differential price between itself and its neighbors for each destination, and passes this information to its neighbors. Each node then performs a distributed algorithm to choose a *matched link* to one of its neighbors that maximizes the differential prices of these two nodes (a link is logical concept and is defined as a wireless channel between two nodes). Packets are forwarded over matched links for a certain time interval. After that, nodes exchange their congestion prices and choose their matched links again.

In sum, existing work in CLO designs falls into two categories: theoretical and practical approaches. Our work attempts to bridge this gap by taking an optimal solution from the utility-based CLO design developed by Chen et al and applying approximations to make this theoretical solution practical. We will review Chen et al's CLO design in more details and point out its impractical assumptions in section III. Section IV present our CLO design with practical approximations for these impractical assumptions.

III. Theoretical Framework of Cross-Layer Optimization

Since Chen et al's CLO design is derived from the optimization framework, it can maximize a utility function (in this case the utility function is end-to-end throughput) and can provide provably optimal performance. However, the CLO design proposed by Chen et al makes a number of assumptions that are rather impractical. In this section, we first review this CLO design in details and then point out its impracticality.

A. Joint Design Algorithm

Each node i maintains per-destination queues for packets. Given a link graph L and a positive scalar step size γ_t, each node i updates its per-destination congestion price with respect to destination k as follows.

$$\begin{aligned} p_i^k(t+1) \quad \leftarrow \quad & p_i^k(t) + \gamma_t(x_i^k(p(t)) - \\ & (\sum_{j:(i,j)\in L} p(t) f_{i,j}^k + \sum_{j:(j,i)\in L} p(t) f_{j,i}^k)) \end{aligned} \quad (1)$$

Here $f_{i,j}^k$ is the rate of packets destined to k flowing from node i to node j during time slot t. The packet rate $f_{i,j}^k(t)$ is computed based on the price information at the beginning of time slot t. Further, x_i^k is the source rate of node i and is adjusted based on the local congestion price at the beginning of time slot t. Given a utility function $U(x)$ that is continuously differentiable, increasing and strictly concave, the source rate can be computed as follows.

$$x_i^k(t) \leftarrow U_s^{'-1}(p_i^k(t)) \quad (2)$$

Each node also exchanges its price information with all of its neighbors and find destination $k^*(t)$ such that

$$k^*(t) \in arg\ max(p_i^k(t) - p_j^k(t)) \quad (3)$$

The rationale here is that for packets destined to k, node i would choose j as the next hop such that $(p_i^k(t) - p_j^k(t))$ is maximized since j is the least congested node among all neighbors j of i. Further, node i first services packets destined for the destination $k^*(t)$ with the most backlog (the highest difference in price).

After node i collects congestion price from all its neighbors, it calculates the differential price and passes this information to all its neighbor.

$$w_{i,j}(t) \leftarrow p_i^{k^*(t)}(t) - p_j^{k^*(t)}(t) \quad (4)$$

To maximize the utility function (e.g., total throughput), nodes allocate a capacity $f_{i,j}(t)$ over link (i,j) such that

$$\sum_{(i,j)\in L} w_{i,j}(t) \times f_{i,j} \quad (5)$$

is maximized. This optimal solution can be arrived at via non-linear programming. Nodes schedule and route packets over each link $(i,j) \in L$ during time slot t according to the rates determined in equation (5). Nodes exchange price information and repeat the same algorithm for each time slot.

B. Impracticality

Since Chen et al's CLO design is derived from the optimization framework, it can deliver provably optimal performance and maximize a utility function such as end-to-end throughput. However, it is difficult to implement this design due to a number of its impractical assumptions.

First, the design only considers primary interference and assumes that communications between nodes can occur simultaneously as long as no two adjoining nodes send and receive at the same time. While simultaneous communications are possible with orthogonal CDMA or FDMA channels, an ad hoc channel assignment scheme is either inefficient or incurs large overhead. Further, since the design does not consider secondary interference, it is not applicable for CSMA networks such as widely deployed IEEE 802.11 and sensor networks.

Second, the design does not provide a mechanism that ensures the rate allocations determined by the optimization framework. The desired rate allocations can be approximated by having nodes match links and transmit exclusively over the matched links. However, this would incur immense communication overhead. Further, support at the MAC layer such as prioritization is necessary to achieve the rate allocations in a broadcast medium.

Third, nodes need to communicate their price and also differential price with each other. While the price information can be broadcast by nodes, this incurs a communication overhead and reduces the effective data rate. The communication overhead can be amortized by reducing the rate of message exchanges. However, this would also degrade the granularity and accuracy of the implementation.

Fourth, routing is coupled with scheduling in Chen et al's CLO design where nodes route packets to the neighbor with the lowest congestion price. However, this can cause nodes to route packets over very long paths. We believe that routing should be guided with topology information to increase resource efficiency and reduce end-to-end delay.

Fifth, lossy links are not considered in Chen et al's CLO design. However, price information without consideration of link's quality can be deceptive because a node could attempt to route and retransmit a packet over links with poor quality multiple times. When this occurs, other packets behind the pending packet are blocked even though they may be destined for a different node and could be routed over different paths.

Impractical assumptions in Chen et al's CLO design make it difficult to be implemented. We will present our approximations for these assumptions that help us obtain a practical CLO solution in section IV. We also extend our CLO design and take into consideration routing topology and link's quality to further improve the overall system performance.

IV. Practical Approximations for Cross-Layer Optimization

In our CLO design, the following components interact with each other to jointly achieve overall system performance: MAC prioritization, scheduling, hop-by-hop flow control, rate adaptation at the sources, and link-aware and congestion-aware routing. Scheduling and MAC prioritization allow nodes with large queue backlogs to take precedence over other nodes in obtaining access to the wireless channel. Hop-by-hop flow control prevents nodes from sending packets too fast and causing buffer overflows at downstream nodes. Rate adaptation at the sources is an extension of hop-by-hop control. It prevents packets from entering the network a faster rate than the network can forward these packets. Link-aware and congestion-aware routing helps nodes route packets over links with good quality and avoid congested areas. In this section, we will first present an overview of our CLO design and then follow with the details of its architectural components.

A. Design Overview

In our design, nodes use an algorithm similar to the ETX algorithm to measure their links' quality [7]. Each node periodically broadcasts an ETX message that contains an increasing sequence number and the number of ETX messages from each of its neighbors that it recently received. A node computes the forward delivery ratio, d_f, to a neighbor as the successful ratio of its ETX messages that this neighbor recently received (and reported). Further, a node computes the reverse delivery ratio, d_r, to a neighbor as the ratio of ETX messages that it recently received from this neighbor. Having d_f and d_r to each of its neighbors j, node i uses a dampening coefficient α to estimate the link's quality between i and j.

$$ETX_{ij} \leftarrow \alpha \times ETX_{ij} + (1-\alpha) \times \frac{1}{d_f \times d_r} \qquad (6)$$

The ETX metric represents the number of expected retransmission count and reflects the quality of a link between two nodes. However, information about link quality alone is not sufficient for nodes to make good routing decisions. When a node chooses the next hop for a packet, it must also have some knowledge about its neighbors' congestion levels in order to make good routing decisions.

Each node in our design maintains per-destination queues for packets that they either generate or forward for other nodes toward the destinations. When a node transmits a packet, it includes the queue length of the corresponding per-destination queue in the packet header. Taking advantage of the broadcast nature of the wireless medium, nodes passively listen to their neighbors' transmissions to collect information about their neighbors' queue lengths. By piggy-backing the queue length information, we obviate the communication overhead that was necessary in Chen et al's CLO design.

A node makes its decisions about routing and packet scheduling based on its ETX metrics to its neighbors and the queue differences between itself and its neighbors. The details of these algorithms are discussed in the subsequent sections.

B. Hop-by-hop Flow Control

Hop-by-hop flow control prevents nodes from transmitting packets too fast and causing buffer overflows at downstream nodes. Further, hop-by-hop flow control can also help reduce end-to-end latency by controlling the queuing delay and limiting the numbers of enqueued packets at each node.

Hop-by-hop flow control is performed by using back pressure between neighboring nodes. When a node detects that its queue is larger than the queue of one of its neighbors (through the mechanisms in section IV-A), it stops forwarding packets to that neighbor and explores alternate paths via other neighbors. Further, if a node receives more traffic to forward than it can pass on to its neighbors, its queue will grow and eventually causes the upstream nodes to either reduce their transmission rates or to find alternate paths to the destinations. This congestion feedback propagates back to the sources and prompt them to reduce their transmission rates.

C. Rate Adaptation

Rate adaptation allows sources to seek an equilibrium state of the network where sources do not inject packets into the network at a rate faster than the rate of packets leaving the network. Unlike TCP that performs rate adaptation on an end-to-end basis, our rate adaptation estimates the congestion level of the network based on the source's queue dynamics and adapts its transmission rate accordingly.

The source samples its own queue periodically and computes the queue difference between the current and the previous sample. If the queue difference is 0, the source probes for available bandwidth by increasing its transmission rate additively $rate = rate + \delta$. Otherwise, the source adjusts its transmission rate based on the queue difference $rate = rate + \frac{QueueDiff}{UpdateInterval}$.

Note that our algorithm for rate adaptation is only intended to replace TCP's congestion control mechanism. A separate mechanism similar to that of mTCP [21] could be used to provide reliable data transport and packet reordering.

D. Scheduling

In order to avoid head-of-the-line blocking where a packet to a certain destination is transmitted over a link with poor quality and would block other packets behind it, each node maintains per-destination queues for packets. A node decides which per-destination queue it services next and where to route a packet from this queue based on a differential congestion price that reflects the ETX metrics and on the queue difference between that node and its neighbors.

Let l_i^k and l_j^k be the per-destination queue length for destination k at nodes i and j. For each destination k, node i computes the differential congestion price C_{ij}^k between itself and each of its neighbors j as follows.

$$C_{ij}^k \leftarrow \frac{l_i^k - l_j^k}{ETX_{ij}} \qquad (7)$$

The differential congestion price C_{ij}^k indicates whether node i should route packets destined for k over node j. If ETX_{ij} is the same for all neighbors j of node i, the largest value of C_{ij}^k hints that node i should route packets destined for node k over the neighbor j with the smallest queue length l_j^k. On the other hand, if l_j^k has the same value for all neighbors j of node i, the largest value of C_{ij}^k suggests that node i should forward packets destined for k to the neighbor j with the most reliable link (the smallest ETX_{ij}).

If multiple per-destination queues at node i are backlogged, node i will first schedule a packet destined for k and forward this packet to its neighbor j so that the differential price C_{ij}^k is maximized for all destinations k and all neighbors j.

Since multiple nodes in a collision range can attempt to obtain access to the wireless channel simultaneously, nodes must use a distributed scheduling algorithm to avoid collision. Depending on its maximum differential price, a node competes for the wireless channel with a high, medium, or low priority. The rationale behind this is that we want to give prioritized access to the wireless medium to nodes that are congested so that they can drain their queues fast (a large queue implies a large differential price). The mechanism for MAC prioritization is ZMAC+ and is discussed in section IV-F. Our scheduling algorithm approximates Chen et al's CLO design in scheduling packets and choosing the next hop but it comes with the advantage that there is no communication overhead of message exchanges.

E. Link-aware and Congestion-aware Routing

The scheduling algorithm discussed in section IV-D allows a node to forward packets to the next hop that is least congested. This scheduling algorithm has to be combined with the routing algorithm to prevent nodes from routing packets over too long or unreliable paths. In our design, nodes run the DSDV routing protocol with ETX as a routing metric. For each destination k, a node restricts its choices for the next hop to K neighbors for the K shortest paths to the destination k (using ETX as the routing metric). The node then runs the scheduling algorithm discussed in section IV-D to choose the next hop for a packet from these K neighbors.

When a node starts, K is initialized to 1. A node increments K to explores new paths when the queue lengths of all current K next hops exceed a threshold $L_{highmark}$. A node decrements K to avoid routing packets over long paths when the queue lengths of all current K next hops fall below a threshold $L_{lowmark}$. In general, the value of K presents a trade-off between high throughput and low delay. The higher K is, the more paths a node can explore and the higher throughput it can achieve. However, some of the taken paths may include a long detour around congested areas and incur considerable delays. For this reason, we also restrict K to $\beta \times N$ where N is a node's number of neighbors and $0 < \beta < 1$. We found in our simulations that $\beta = 0.1$ is a good compromise that allows our design to achieve good throughput without incurring considerable delays.

F. MAC Prioritization

We design ZMAC+, a mechanism for MAC prioritization, that supports the scheduling component discussed in section IV-D to give congested nodes preferential access to the wireless channel and approximate the optimal rate allocations. ZMAC+ is based on ZMAC, a hybrid MAC which combines the strengths of CSMA and TDMA [17]. ZMAC uses CSMA as the basic mechanism for MAC but uses a TDMA slot assignment as a "hint" to enhance contention resolution. In ZMAC, nodes run a distributed scheduling algorithm for time slot assignment. A node that is assigned to a time slot is an *owner* of that slot and other nodes are *non-owners* of that slot.

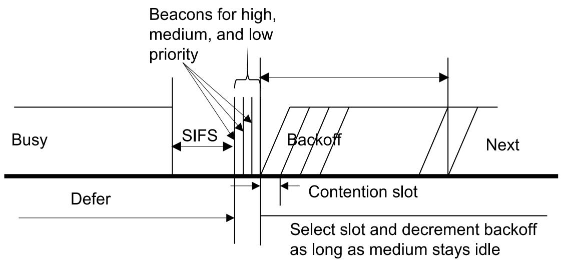

Beacons for high, medium, and low priority
Busy
SIFS
Backoff
Next
Defer
Contention slot
Select slot and decrement backoff as long as medium stays idle

Fig. 1. ZMAC+ backoff mechanism

A node performs carrier-sensing and can transmit a packet when the channel is idle. However, the owner of the current time slot has a shorter initial contention window size and thus higher priority in obtaining access to the channel than the non-owners. Thus, when owners have data to transmit in their time slots, ZMAC reduces the chance of collision. However, when owners do not use their time slots, non-owners can opportunistically transmit their data.

ZMAC+ is built on ZMAC and supports three priority levels: high, medium, and low. The number of priority levels represents a trade-off between implementation overhead and fine granularity. Since ZMAC+ is intended to be an alternate mechanism for IEEE 802.11e that supports four priority levels, we implemented four priority levels in our design (our intention is that our CLO design could run on top of IEEE 802.11e seamlessly). However, the highest priority level is reserved for routing messages.

A node chooses a priority level for a packet based on its differential congestion price. The details of ZMAC+ are depicted in Figure 1. When a node with high priority detects that the channel is idle, it waits for a short interframe space (SIFS) before it attempts to transmit a packet. If the channel is still idle after SIFS, the node transmits a beacon (a short burst) to contend for the channel. Ideally, we would like a node to immediately proceed to transmit a packet if there is no collision during its beacon transmission. However, since nodes cannot sense the channel while transmitting, they use ZMAC's mechanism to resolve possible contention after the beacon transmission.

When a node with medium priority senses that the channel is idle, it waits for (T + SIFS), where T is the transmission period of a beacon. If the channel becomes busy during that time interval, the node backs off. If the channel is still idle after that time interval, the node transmits a beacon and then executes the ZMAC algorithm to contend for the wireless channel.

A node with low priority defers to nodes with high and medium priority by waiting for (2*T + SIFS). If the channel

TABLE I
THE DEFAULT SETTINGS FOR DCLO AND DPCLO

Radio Channel Data Rate	2Mbps
Radio Transmission Range	250m
Radio Interference Range	550m
Packet Size	200 bytes
Per-Destination Queue Size	3 packets
$L_{lowmark}$	2 packets
$L_{highmark}$	2 packets
β	0.1

becomes busy during this period, it backs off. Otherwise, the node with low priority proceeds to compete for the wireless channel by transmitting a short beacon and then falling back to the mechanism of ZMAC for contention resolution.

V. Performance Evaluation

We evaluate the performance of our practical cross-layer optimization framework with existing congestion control schemes – Fusion [12], TCP, TCP-FeW [16], and our implementation of the distributed matching based algorithm proposed by Chen et al. [3]. Note that our cross-layer optimization framework is a complex system which requires interactions between transport, network and MAC layers. To accurately pinpoint the contribution of each layer to the overall performance, we first test the performance on small, simple topologies, where it is easier to do so. Then we proceed to test the performance on a large 108 node multi-hop wireless network with different flow scenarios.

A. Implementation Details

Chen's et al. CLO algorithm described in section III requires a centralized implementation to construct the network topology as a weighted graph and would cause an immense communication overhead. For this reason, Chen et al. proposed a distributed algorithm that approximates their theoretical CLO design. Due to space limitation, we omit details of Chen's distributed CLO algorithm (DCLO) and only summarize that DCLO calculates the differential prices and link weights as before. After that DCLO has each node find a *matching* neighbor from its "free" neighbors so that the link weight between these two nodes is maximized. During a matching interval, nodes forward data packets over matched links exclusively. We implemented DCLO in ns-2 and compare it with our own distributed practical CLO algorithm (DPCLO). Further, we implemented DCLO and DPCLO on top of Z-MAC, Z-MAC+, and IEEE 802.11e to explore the effects of MAC implementation on the overall performance of a CLO design.

For DPCLO, at the network layer, we modify the DSDV routing protocol in ns to broadcast ETX messages at regular periods of time for link error estimation. In addition, we modify the routing queue to implement per destination queues. At the transport layer, source nodes update their transmission rates based on the rate-control algorithm introduced in Section IV-C. Intermediate nodes schedule packets based on the price based queuing introduced in Section IV-D. The price of each packet is passed on to the prioritized MAC, which then transmits the packet with a priority decided by the price associated with the packet. The default parameters used for DCLO and DPCLO are given in Table I.

B. Comparison with Distributed Matching (DCLO)

We first compare the performance of DCLO (based on distributed matching) to DPCLO (based on Z-MAC+). Consider the topology shown in Figure 2 (A). The edges show connectivity between nodes. We create two flows A→E and F→D from T=200s to T=300s. The throughput obtained by each flow in DCLO and DPCLO is shown in Figure 2 (B) and (C) respectively. While the rates converge in both cases, DCLO achieves somewhat less throughput than DPCLO. This is because of two reasons: 1) the communication overhead involved in the creation of the matching. 2) Due to the absence of routing information in DCLO, some packets traverse several hops before eventually reaching their destination, wasting valuable bandwidth. Due to the small size of the topology, the performance degradation is not significant in this case. However, as we shall see later, this has a much more adverse effect on larger and/or denser topologies.

C. Micro-benchmarks

In this section, we will study the performance of DPCLO on small topologies to illustrate the contribution of each component of the algorithm to the overall system performance.

1) Single-Path Vs Multi-Path Routing: We first look at how the multi-path nature of DPCLO improves fairness between flows and allows flows to dynamically change their paths to react to congestion. We run an experiment with two flows, A→E and D→F from time T=200s to time T=300s.

We compare two variants of DPCLO, 1) DPCLO restricted to single path routing – each node forwards packets to the next hop based on shortest path routing, but source rate control, hop by hop flow control and prioritized MAC are left in place, and 2) DPCLO with all features enabled – each node forwards packets to neighbors based on their queue occupancies. Figure 3 shows the throughput achieved by each flow for both cases. In the case of single path routing, A forwards packets to E based on shortest path routing through the node D. Hence D has to forward packets from A as well as its own generated packets, reducing the rate of flow 2. In the case of DPCLO with multi-path routing, A utilizes paths A-B-C-E as well, reducing the load on node D, leading to better fairness between flows. In doing so, however, the aggregate throughput reduces by about 50%.

2) Link Aware Routing and Scheduling (ETX): We now consider the case of lossy wireless links. Maintaining the same topology as in Figure 3 (A), at the beginning of the simulation, each link is assigned a constant loss rate chosen randomly from the interval [0,0.6]. We maintain two flows, A→E and D→F. The average throughput of each flow over 200s of simulation time are shown in Figure 4. The increase in throughput occurs due to two main reasons: 1) The use of ETX as a routing metric allows each node to select those next hop neighbors

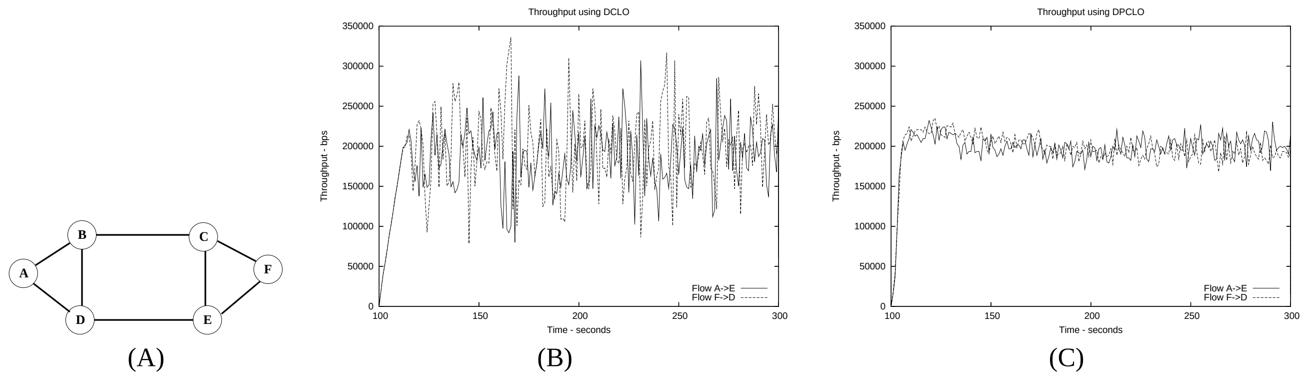


Fig. 2. Throughput performance comparison between Chen et al.'s DCLO and our DPCLO design. Two flows A→E and F→D are generated on the topology shown in (A) from time T=200s to time T=300s.

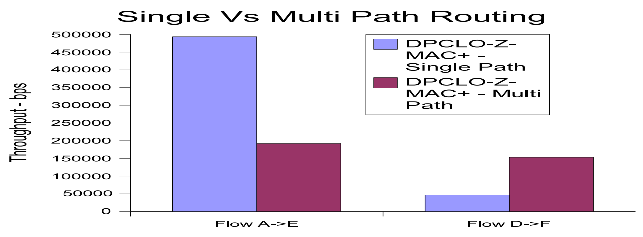


Fig. 3. Comparison of flow throughputs when DPCLO is run with and without multi path routing. We use the same topology as in Figure 2 (A). Two flows, A→E and D→F start at time T=200s and stop at time T=300s.

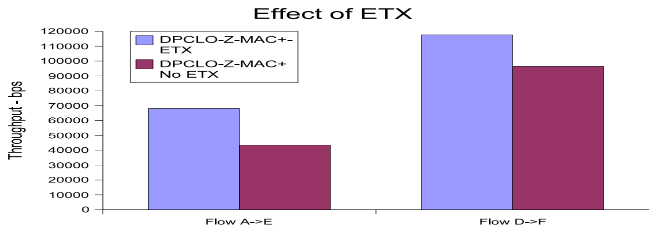


Fig. 4. The effect of ETX when DPCLO-Z-MAC+ is used on a wireless network with lossy links. We use the same topology and flow scenario as Figure 2 (A). Links are assigned loss rates chosen uniformly from the interval [0,0.6].

as forwarders to which they have high transmission success probability. 2) This selection is augmented by the use of price based queuing, which takes ETX into account, as explained in Section IV-D.

3) TDMA Vs CSMA MAC: Z-MAC+ uses a TDMA based MAC, namely Z-MAC, to resolve contention, whereas IEEE 802.11e uses a CSMA based MAC, namely 802.11, to resolve contention. The use of these two kinds of MAC for implementing the prioritized MAC in DPCLO has different kinds of ramifications on the performance. Consider the three scenarios shown in Figure 5 (A), (D) and (G). In all three cases, the edges indicate the transmission *as well as the interference range*. In Figure 5 (A), node A cannot sense the transmissions by node C. When using a CSMA based MAC, due to lack of coordination between the transmissions of A and C, A sees much higher loss rates compared to C, causing starvation for node A. A similar case holds for topologies in Figure 5 (D) and (G), however due to symmetric flow orientation, the starvation induced is short-term and eventually both flows get equal throughput.

Figure 5 (B) and (C) show the throughput of flows A→B and C→D on topology A when using DPCLO-Z-MAC+ and DPCLO-802.11e respectively. We can observe that in the case of DPCLO-802.11e, flow A→B gets much less throughput compared to the flow C→D. In the case of DPCLO-Z-MAC+, flow A→B does get reduced throughput, but the disparity is not as much as with DPCLO-802.11e. This is because DPCLO-Z-MAC+ is based on TDMA, hence A and C both begin transmissions at the beginning of the TDMA slot, reducing the packet loss rate for A considerably, compared to the case where both A and C transmit in an uncoordinated manner as in DPCLO-802.11e.

Figure 5 (E) and (F) show the throughput obtained by flows A→B and D→C on topology D when using DPCLO-Z-MAC+ and DPCLO-802.11e respectively. In the case of DPCLO-802.11e, although each flow captures the channel for significant periods of time, eventually however they relegate access to the other flow, resulting in long term fairness. This is because of the exponential backoff present in 802.11e. A node which has just finished packet transmission starts off with a small backoff window, but nodes which faced collisions exponentially increase their backoff windows, leading to capture. Again, being based on TDMA, DPCLO-Z-MAC+ does not face this problem, providing both flows with same throughput. A similar case can be observed for the topology C, where the corresponding flow throughputs are shown in Figure 5 (H) and (I). However, interestingly 802.11e shows better aggregate throughput compared to Z-MAC+. We attribute this to the use of RTS/CTS in 802.11e which effectively synchronizes nodes A and C and prevents capture. Note that RTS/CTS is ineffective in the case of topology D since sources A and D are more than two hops away from each other.

DPCLO-Z-MAC+ is highly effective in dealing with starvation, given tight time synchronization. Without time synchronization, DPCLO-802.11e represents a reasonable compromise between performance and implementation feasibility.

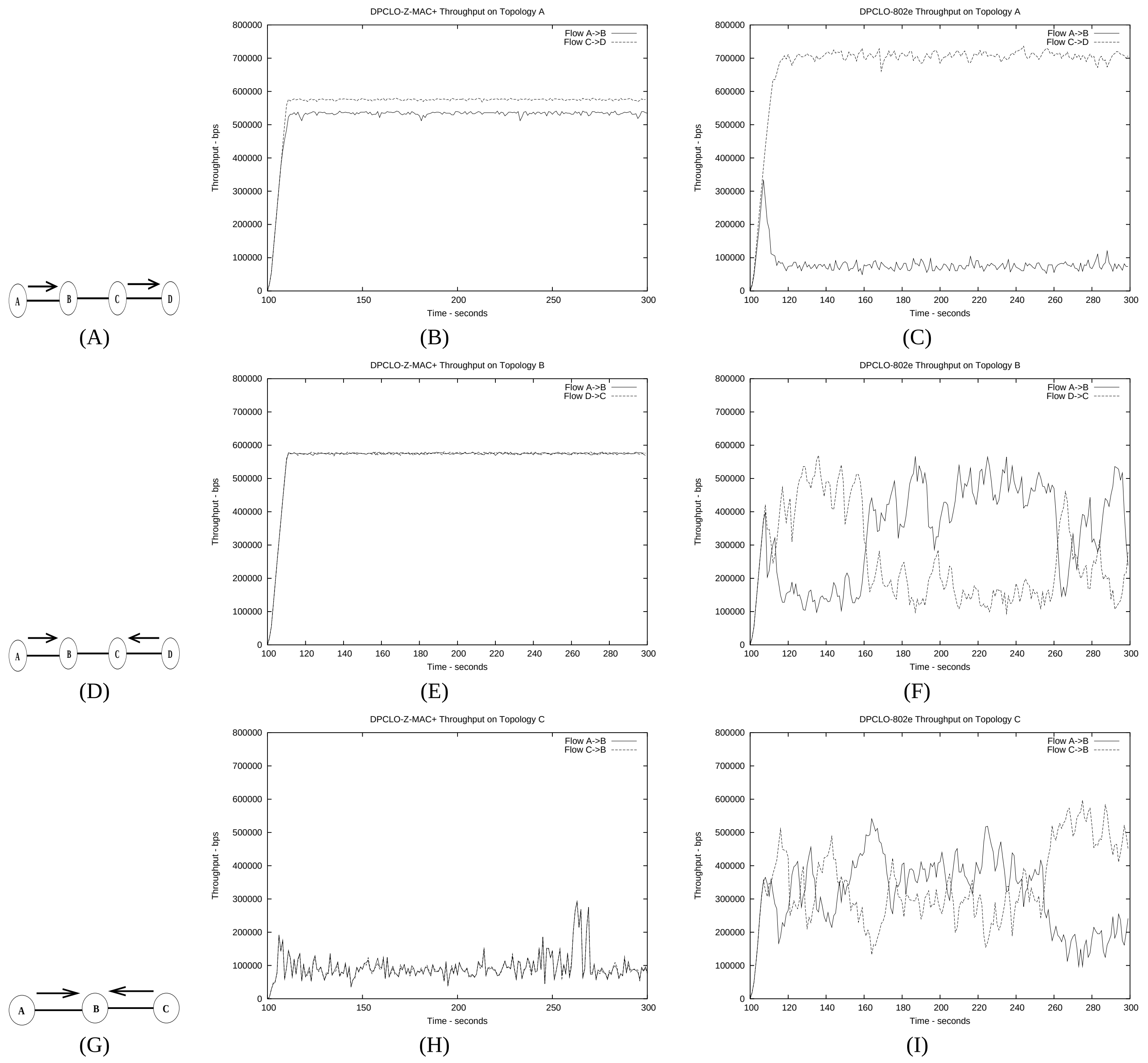
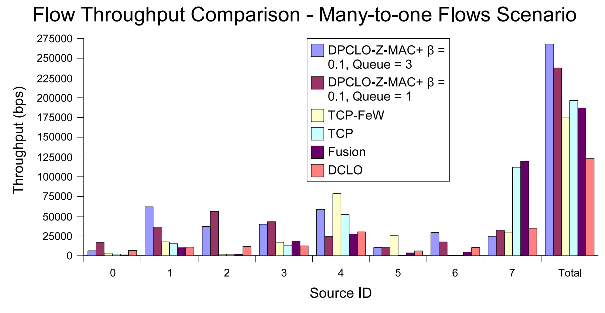


Fig. 5. Throughput of flows in three possible starvation scenarios when using DPCLO with TDMA-based Z-MAC+ and CSMA-based 802.11e. Figure (A) is one instance of the well known Information Asymmetry problem [10] which results in long-term starvation when used with randomized uncoordinated CSMA MAC protocols. Figures (D) and (G) represent symmetric flow scenarios where starvation is short-term.

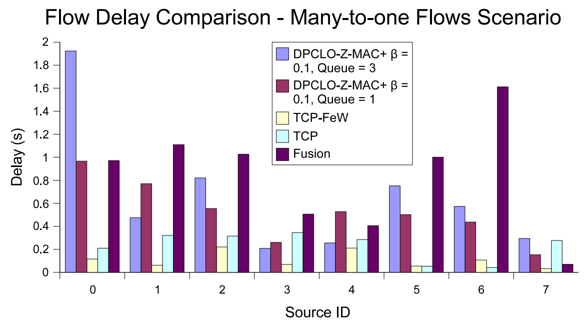
D. *Macro-benchmarks*

We now test DPCLO in large multi-hop networks. We first create a topology consisting of 108 nodes randomly placed in a 1250m x 1250m area. Since nodes have a radio transmission range of 250m, this results in a network with a diameter of 8 hops and an average one-hop neighborhood size of 13 nodes. We then create two flow scenarios: a) a many-to-one flow scenario scenario shown in Figure 6, where 7 selected nodes transmit packets to the designated sink node. This kind of scenario is most often found in sensor networks. b) a many-to-many flows scenario where flows are generated between randomly selected nodes. We will later describe in detail the methodology for generating such flows. This kind of scenario is most often found in mesh networks.

1) Many-to-one flow scenario: In this experiment, all sources begin transmission at time T=200s and finish transmission at T=300s. We report the average throughput and delay obtained at the sink node under DPCLO-Z-MAC+, TCP, TCP-FeW (with $\alpha = 0.01$), Fusion and DCLO. Figure 7 (A) shows the per flow throughput as well as the aggregate throughput in this experiment. The sources are ordered with respect to their distance from the sink node. We observe that TCP heavily

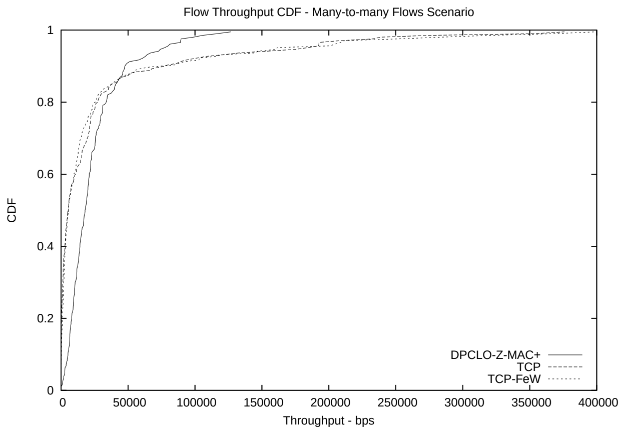


(A)

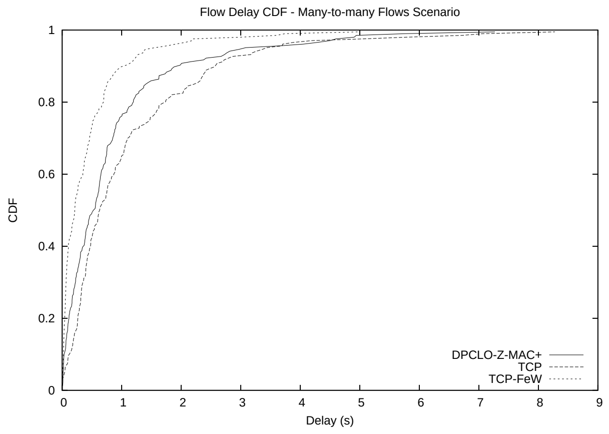


(B)

Fig. 7. (A) and (B) show the average flow throughput and end-to-end delay obtained. Note that on the x-axis, the sources are ordered in terms of their distance to the sink node.



(A)



(B)

Fig. 8. (A) and (B) show the CDF of the average flow throughput and end-to-end delay for the many-to-many flow scenario. 206 flows are generated during a simulation time of 800s, with source and destination nodes chosen randomly on the topology shown in Figure 7 (A). Flow inter-arrival times and flow durations are picked from a Weibull and Log-normal distribution respectively.

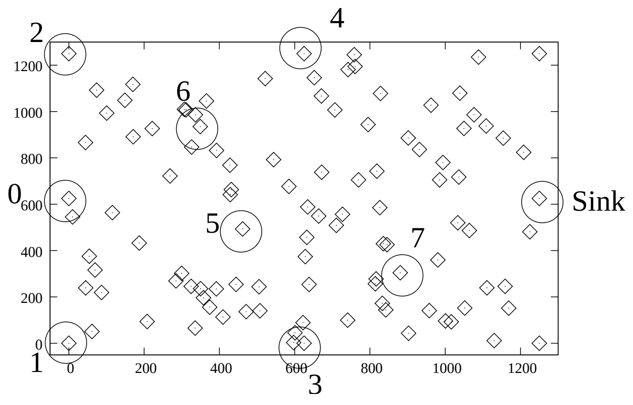


Fig. 6. Topology and position of sources for the many-to-one flows scenario. All 8 sources transmit as fast as possible towards the sink node located on the right edge of the topology.

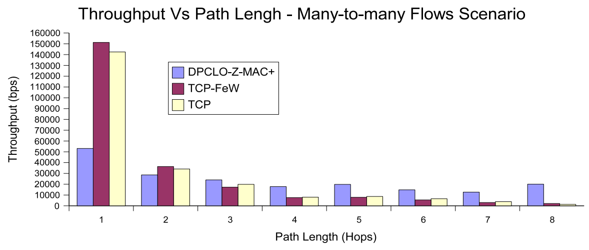


Fig. 9. Distribution of the throughput among flows based on their path lengths in terms of hops.

favors short flows. Hence, source node 7 which is closest to the sink node attains the highest throughput, but in the process it drastically reduces the throughput of other flows. TCP-FeW reduces this bias and allows longer flows to obtain some throughput but the bias is still apparent. DPCLO-Z-MAC+, on the other hand does not starve any flows, and allows long flows to attain significant throughput. Interestingly, source node 7 shows less throughput than other longer flows. This is because the source node 7 is near the sink and hence the sources farther away from the sink may forward their packets through this source, reducing its own rate (remember that DPCLO-Z-MAC+ allocates per destination queues). DPCLO-Z-MAC+ also attains the highest aggregate throughput on this scenario. However, we found that the DPCLO-Z-MAC+ maximizes throughput at the expense of end-to-end delay. Figure 7 (B) shows the average end-to-end delay experienced by each flow in this scenario. While delay when using DPCLO-Z-MAC+ is comparable to TCP and TCP-FeW for most flows, in the worst case (for source 0), we see an increase of 10x in the average delay. The cause of the delay was found to be excessive queuing at intermediate nodes. By using a queue size of just one packet per destination, it was possible to reduce this delay significantly, but at the expense of aggregate throughput. Fusion performs very similar to TCP. Lastly, we note that DCLO performs significantly worse than DPCLO on this topology – DCLO experienced delays of the order of few tens of seconds, and so we do not report it in Figure 7 (B). This is because of the high density of this network, due to which the overhead for creating the matching increases, and packets traverse several hops before reaching their destinations.

2) Many-to-many flows scenario: For this experiment, we generate 206 flows on the topology shown in Figure 6 starting at time T=200s and ending at time T=1000s. Flow inter-arrivals times are modeled on the Weibull distribution, and flow durations are based on the Log-normal distribution with parameters based on a study of flows in a large wireless network [15]. On average, a flow lasts for 30s, but due to the heavy-tailed nature of the Log-normal distribution, a small fraction of the flows last for periods as long as 100s. Source and destination nodes for each flow are chosen randomly from the 108 nodes in the topology.

Because of the large number of flows we present the CDF of the average delay and throughput per flow in Figure 8 (A) and (B) respectively. From Figure 8 (A), we make the following observations. DPCLO-Z-MAC+ tends to distribute flow throughputs equally among all the flows, resulting in a very steep slope in the CDF. About 85% of the flows obtain a higher throughput when using DPCLO-C-MAC+ than TCP. Also, in the case of TCP, a very small fraction (about 10%) of the flows achieves very high throughput, while about 20% of the flows are starved. Interestingly, while TCP-FeW does not seem to improve the throughput much, it reduces the delay considerably, as can be seen from Figure 8 (B).

To investigate the wide disparity in throughput, we plot the throughput per flow with respect to the path length in Figure 9. From this figure, we can see that the set of flows with high throughput are the 1-hop TCP flows, while the set of flows which get starved are the 8-hop TCP flows. This clearly shows the TCP bias toward flows with long path lengths. The throughput achieved by flows when using DPCLO-Z-MAC+ is roughly equal regardless of their path length.

VI. Summary and Conclusions

Cross-layer optimization (CLO) is an approach that coordinates protocol behaviors at different layers to improve overall system performance. This approach has recently received considerable attention due to the challenging nature of wireless networks. So far existing work in CLO design essentially falls into two categories: theoretical CLO designs that are provably optimal but rather impractical, and practical CLO designs whose analytical behaviors are difficult to understand.

In this work, we attempted to bridge the gap between theory and practice by taking a theoretical CLO design, pointing out its impracticality, and applying practical approximations to make it practical. We identified the important components of a CLO design for wireless networks: source rate control, hop-by-hop flow control, MAC scheduling and prioritization, link-aware and congestion-aware routing, and discussed how they can be integrated. Our result is a practical CLO solution that approximates theoretically-derived optimal solution. We performed extensive simulations to investigate the effects of each of our CLO components. Further, our simulations also demonstrated that our CLO design achieves significant performance improvement over existing practical solutions.

References

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, “Link-level measurements from an 802.11b mesh network,” in *ACM SIGCOMM’04*.

[2] S. Biswas and R. Morris, “ExOR: Opportunistic multi-hop routing for wireless networks,” in *ACM SIGCOMM’05*.

[3] L. Chen, S. Low, M. Chiang, and J. Doyle, “Optimal cross-layer congestion control, routing and scheduling design in ad hoc wireless networks,” in *IEEE Infocom’06*.

[4] L. Chen, S. Low, and J. Doyle, “Joint congestion control and media access control design for wireless ad hoc networks,” in *IEEE Infocom’05*.

[5] M. Chiang, “To layer or not to layer: balancing transport and physical layers in wireless multihop networks,” in *IEEE Infocom’04*.

[6] M. Conti, G. Maselli, G. Turi, and S. Giordano, “Cross-layering in mobile ad hoc network design,” *IEEE Computer, special issue on Ad Hoc Networks*, Feb. 2004.

[7] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris, “A high-throughput path metric for multi-hop wireless routing,” in *ACM MobiCom’03*.

[8] S. ElRakabawy, A. Klemm, and C. Lindemann, “TCP with adaptive pacing for multihop wireless networks,” in *ACM MobiHoc’05*.

[9] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, “The impact of multihop wireless channel on TCP throughput and loss,” in *IEEE Infocom’03*.

[10] M. Garetto, J. Shi, and E. Knightly, “Modeling media access in embedded two-flow topologies of multi-hop wireless networks,” in *ACM MobiCom’04*.

[11] G. Holland and N. Vaidya, “Analysis of TCP performance over mobile ad hoc networks,” in *ACM MobiCom’99*.

[12] B. Hull, K. Jamieson, and H. Balakrishnan, “Mitigating congestion in wireless sensor networks,” in *ACM SenSys’04*.

[13] V. Kawadia and P. Kumar, “A cautionary perspective on cross-layer design,” *IEEE Wireless Communications Magazine*, Feb. 2005.

[14] X. Lin and N. Shroff, “The impact of imperfect scheduling on cross-layer rate control in multihop wireless networks,” in *IEEE Infocom’05*.

[15] X. Meng, S. Wong, Y. Yuan, and S. Lu, “Characterizing flows in large wireless data networks,” in *ACM MobiCom’04*.

[16] K. Nahm, A. Helmy, and J. Kuo, “TCP over multihop 802.11 networks: Issues and performance enhancement,” in *ACM MobiHoc’05*.

[17] I. Rhee, A. Warrier, M. Aia, and J. Min, “Z-MAC: a hybrid MAC for wireless sensor networks,” in *ACM SenSys’05*.

[18] I. . WG, “Draft supplement to IEEE standard 802.11-1999: Medium access control (MAC) enhancements for quality of service (QoS),” IEEE 802.11e D6.0, Nov. 2003.

[19] K. Xu, M. Gerla, L. Qi, and Y. Shu, “Enhancing TCP fairness in ad hoc wireless networks using neighborhood red,” in *ACM MobiCom’03*.

[20] X. Yu, “Improving TCP performance over mobile ad hoc networks by exploiting cross-layer information awareness,” in *ACM MobiCom’04*.

[21] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang, “A transport layer approach for improving end-to-end performance and robustness using redundant paths,” in *USENIX’04*.