Improving Throughput in Multihop Wireless Networks

Zongpeng Li and Baochun Li, *Senior Member, IEEE*

***Abstract*—One of the main characteristics of wireless *ad hoc* networks is their node-centric broadcast nature of communication, leading to interferences and spatial contention between adjacent wireless links. Due to such interferences, pessimistic concerns have been recently raised with respect to the decreasing network capacity in wireless *ad hoc* networks when the number of nodes scales to several orders of magnitude higher. Such studies assume uniformly distributed nodes in the network and randomized traffic patterns. In this paper, we argue that in all cases of end-to-end data communications—including one-to-k unicast and multicast data dissemination as well as k-to-one data aggregation—the maximum achievable end-to-end data throughput (measured on the sources) heavily depends on the strategy of arranging the topology of transmission between sources and destinations, as well as possible per-node operations such as coding. An optimal strategy achieves better end-to-end throughput than an arbitrary one. We present theoretical studies and critical insights with respect to how these strategies may be designed so that end-to-end throughput may be increased. After all, under all circumstances—in either a lightly loaded or a congested network—increasing end-to-end throughput from its baseline is always beneficial to applications using *ad hoc* networks to communicate.**

***Index Terms*—*Ad hoc* networks, communication systems, computer networks, network coding, routing.**

I. Introduction

WIRELESS *ad hoc* networks consist of untethered nodes that communicate with each other over multiple wireless hops, with participating nodes collaboratively forwarding ongoing traffic. Although both use multiple hops to relay traffic, data communication in wireless *ad hoc* networks is inherently different from wireline networks. Wireline networks are "link-centric": Each link connects two network interfaces, and there is no interference between any two independent links. In comparison, wireless *ad hoc* networks are "node-centric": Data communications are broadcast in nature. Data packets transmitted are broadcast by the source to all its neighboring nodes, such that communication links exist between any pairs of nodes that are within transmission range of each other.

With respect to contention, compared with wireline networks where flows contend only at the packet router with other simultaneous flows through the same router (contention in the "time domain"), the broadcasting characteristics of medium access control protocols in wireless networks show that flows also compete for shared channel bandwidth if they are within the transmission ranges of each other (contention in the "spatial domain"). This is further exacerbated by the use of control packets (Request To Send/Clear To Send, RTS/CTS) to solve the hidden and exposed terminal problems, leading to interference when either the source or the destination of two single-hop flows are in the same transmission range.

On the brighter side, we note that the broadcast nature of wireless *ad hoc* networks may be of assistance in the "multicast" scenario that, originating from the same source, multiple data flows to their respective destinations transmit identical data. In this case, data only need to be transmitted once by local broadcasts. This is identified as the "wireless advantage" [1] when studying efficient construction of multicast trees in *ad hoc* networks.

The interference model of wireless *ad hoc* networks has raised pessimistic concerns about the scalability of the network with respect to the "network capacity" [2]–[5]. The conclusion was that, under the assumption of idealized scheduling algorithms, uniformly distributed nodes, and randomized traffic patterns, the available network capacity does not scale well when the total number of nodes in a wireless *ad hoc* network scales to several orders of magnitude higher. In fact, for a network of n nodes, the achievable end-to-end throughput available to each node is only roughly $O(1/\sqrt{n})$ (or more precisely, for a network with uniformly random node placement and random traffic patterns, $O(1/\sqrt{n \log n})$ [2]).

In this paper, we propose to revisit the problem of end-to-end throughput and approach the issue from a different perspective. Rather than analyzing the achievable throughput in an *ad hoc* network with idealized assumptions such as random traffic patterns and uniformly distributed nodes, we show that it is more practical and important to "increase the end-to-end throughput" available to a multihop session connecting a set of sources and destinations in an application, from its baseline determined by previous analytical studies. From this point of view, previous work [3] has proposed the idea of localizing traffic, so that most of the flows use very few hops to reach the destination. Inasmuch as it is up to the applications to determine source–destination pairs, such a goal of localizing traffic is beyond the scope of network-level algorithms. In our work, we believe that the maximum achievable end-to-end throughput heavily depends on the strategy of: 1) arranging the "network topology" between the sources and destinations, including the "end-to-end paths" that traffic may follow and 2) activating per-node algorithms such as network coding [6]–[9] for assistance.

Manuscript received September 12, 2005; revised January 14, 2006. The review of this paper was coordinated by Prof. X. Shen.

Z. Li is with the Department of Computer Science, University of Calgary, Calgary, AB T2N 1N4, Canada (e-mail: zongpeng@cpsc.ucalgary.ca).

B. Li is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: bli@eecg.toronto.edu).

Digital Object Identifier 10.1109/TVT.2006.873833

0018-9545/$20.00 © 2006 IEEE

A carefully determined optimal strategy achieves better end-to-end throughput than an arbitrary strategy.

We discuss solutions to the problem of increasing end-to-end throughput in three cases that cover all scenarios of end-to-end communications. 1) We examine the case of "unicast data dissemination," where multiple data items are disseminated to their respective destinations, from a single source. We exploit multipath data dissemination exclusively toward the goal of increasing throughput, rather than fault tolerance or load balancing as in previously work. 2) We examine the case of multicast data dissemination, where a single data item is disseminated to multiple destinations, from a single source. We argue that multipath dissemination is less effective in this case, where we exploit network coding to achieve the same objective. 3) We study the case of "data aggregation," where multiple data items from their respective sources are aggregated to a single destination. In this case, we believe that the timing and topology of aggregation play critical roles when the objective is to increase throughput. Using the definitions and theorem we develop in Section II, our discussions are generic to *ad hoc* networks without limiting to any specific protocols at the network layer. The objective of this work is to provide theoretical insights toward determining optimal strategies to increase throughput in end-to-end transmissions.

The remainder of the paper is organized as follows. Preliminaries are presented in Section II. Section III discusses the case of data dissemination, including both unicast and multicast cases. Section IV presents the case of data aggregation. Sections V and VI discuss related work and conclude the paper.

II. Preliminaries

We model a wireless *ad hoc* network as a collection of homogeneous wireless nodes deployed within a two-dimensional geographical territory. Each node is equipped with an omnidirectional antenna, where both the transmission range and the interference range are R. The single-hop wireless channel capacity is C. Data packets are relayed from the source nodes to the destination nodes via intermediate nodes in a multihop fashion. Local packet delivery is achieved by broadcasting at the medium access layer (MAC) layer. Assuming each multihop data flow consists of multiple single-hop segments of flows (hereafter referred to as "subflows"), we adopt the flow contention model presented in previous work [10], [11]: Two single-hop subflows of a multihop flow interfere with each other if and only if either the source or the destination of both flows are within the single-hop transmission range. Further, we focus on multihop flows that traverse more than two hops, thus consisting of more than two subflows, because these multihop flows exhibit spatial contention even among its own subflows.

Naturally, there exists a fundamental tradeoff between maximum achievable throughput and arbitrating fairness among greedy flows. Maintaining strict fairness may sacrifice total throughput, whereas maximizing throughput may starve certain sessions. Such a tradeoff is usually resolved by scheduling algorithms in the MAC layer [10], [11]. Inasmuch as MAC layer scheduling and arbitration among different multihop flows are beyond the scope of this work, we make the following assumptions with respect to this tradeoff. 1) We assume ideal MAC layer scheduling, and proceed to analyze the theoretical bound with respect to the achievable throughput, leading to strategies that may be used to increase such achievable throughput. 2) We assume that each source–destination connection is equally important and should enjoy the same throughput.

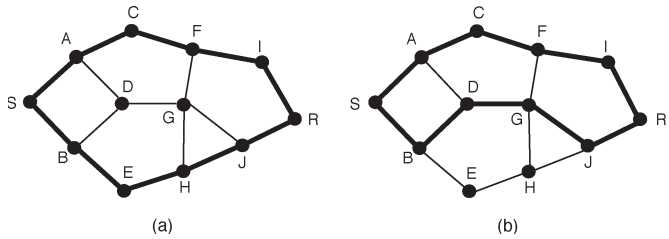


Fig. 1. Concept of one-hop away. (a) Two routes that are one-hop away. (b) Two routes that are not one-hop away.

A "flow" is the transmission of the same data along a route, which can be divided into multiple single-hop subflows. Two nodes are "one-hop away" if they are not within transmission range of each other. Two routes are one-hop away if, beside the end nodes, each node on one route is one-hop away from any node on the other route. Note that we allow the end nodes to be identical, as in the case of multipath routing. For example, in Fig. 1, route $S{-}A{-}C{-}F{-}I{-}R$ and route $S{-}B{-}E{-}H{-}J{-}R$ are one-hop away, whereas route $S{-}A{-}C{-}F{-}I{-}R$ and route $S{-}B{-}D{-}G{-}J{-}R$ are not. Although the latter pair of routes does not intersect at intermediate nodes, there are interroute links connecting some of their intermediate nodes, namely, link AD and link FG. Therefore, they are not one-hop away.

When we study the achievable throughput r, we focus on "source–end throughput," i.e., the throughput measured collectively at the sources. Our fairness rule requires that each source–destination connection has the same throughput. This implies that, during a given time period, the time that a source transfers data for each of the sessions originating from this source is identical for all sessions and at all sources. Therefore, we can analyze the achievable throughput by examining the smallest time T it takes to schedule the subflows without interference, such that each source transmits data for each of its sessions for the same amount of time t_0, and all these data are successfully transmitted to the corresponding destinations. Let S be the set of sources, and t_i be the amount of time source i is scheduled to transmit during the scheduling period T; we have $r = C \cdot \sum_{i \in S} t_i / T$.

Henceforth in this paper, we label links with numerical "weights," such that each weight is equal to the total time all the subflows at the corresponding link are scheduled to transmit during the scheduling period T. To facilitate presentation, we scale time such that $t_0 = 1$ s.

We use an example to illustrate the concepts and definitions above. Fig. 2 shows two unicast sessions, the sources are S_1 and S_2, respectively, and the destination is R for both sessions. There are two flows, transmitting on route $S_1{-}A{-}C{-}D{-}R$ and route $S_2{-}B{-}C{-}D{-}R$, respectively. Our scheduling rule requires each of them to deliver $C \cdot (1\ \text{s})$ amount of data; therefore, each subflow of them needs to be scheduled for 1 s

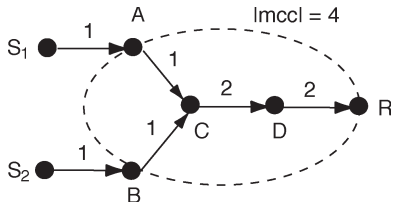


Fig. 2. Two independent unicast sessions with the same destination.

during one schedule period. There are two subflows at link CD and link DR, respectively; hence, they both have weight 2. Each of the other four links has weight 1 because they each serves one flow only.

Intuitively, the achievable throughput depends on the level of contention among subflows. The more intense the contention is, the lower throughput may be achieved. We proceed to introduce the concept of a maximum contention clique (mcc), which is used to characterize the above intuition in a following theorem. The proved theorem will be used in our analysis throughout the remainder of this paper.

A "contention clique" is a set of links such that any two links within the set interfere with each other. The size of a contention clique is the summation of all the weights on its links. The contention clique with the maximum size is called the mcc, its size denoted as |mcc|. For example, in Fig. 2, we observe that the contention clique formed by links AC, BC, CD, and DR, which has size 6, is the mcc of the transmission network.

Theorem 2.1: For a transmission network consisting of one or more sessions, the achievable throughput $r \leq C\sum_{i\in S} t_i/|\text{mcc}|$; equality holds if the transmission topology is a forest.

Proof: Inasmuch as $r = C\sum_{i\in S} t_i/T$, we need to show that $T \geq |\text{mcc}|$ always holds, and $T = |\text{mcc}|$ if the underlying topology is a forest, i.e., there are no cycles in it. The time period it takes to schedule all subflows is at least as long as it takes to schedule the mcc, if interference is to be avoided. Therefore, it is immediate that $T \geq |\text{mcc}|$. In the rest of this proof, we focus on showing that equality holds when the network topology is cycle-free.

We take a constructive approach in the proof by extending a local interference-free schedule of length |mcc| to the entire network. Consider a breadth-first traversal of subflows in the network, according to their topological distance to the mcc (ties are broken arbitrarily). Upon examining each subflow in the traversal, we try to fit it into the schedule without increasing the total schedule length, |mcc|. We claim the following property of such a traversal. ■

Claim: At each step during the traversal, the current subflow and visited subflows that interfere with it together form a contention clique.

Proof of claim: Suppose the current subflow being examined is on link uv. Consider removing uv from the topology. Inasmuch as the network topology is cycle-free and a breadth-first traversal is taken, one of the two subtrees resulted contains unexamined nodes only. Without loss of generality, suppose this is the subtree containing v. Now, instead of removing link uv, consider removing u and all its incident links. Among the smaller subtrees resulted, at most one can contain more than one examined subflows (denote this subtree as t^*. Each of the other subtrees either contains no examined subflows or contains exactly one examined subflow incident to u. There are two types of subflows that are already scheduled and interfere with the current subflow on uv: 1) in t^* or 2) not in t^*. Denote the set of such subflows not in t^* as F_1, and the set of such subflows in t_* as F_2. Subflows in F_1 interfere with each other because they share the same end node u. Subflows in F_2 interfere with each other because they also share a common end node, which is the neighbor node of u in t^*. Subflows in F_1 interfere with subflows in F_2 because: 1) uv interfere with subflows in F_2 and 2) the topological distance between a subflow in F_1 is the same as that between uv and a subflow in F_2. That concludes our proof of the claim.

The size of this contention clique containing subflow uv is upper-bounded by the size of the mcc in network, which is |mcc|. Therefore, we are able to extend the current schedule to accommodate subflow uv without introducing interference or prolonging the schedule length. When the breadth-first traversal terminates, we eventually obtain a schedule of all links without interference using time |mcc|. We conclude that $T = |\text{mcc}|$ in this case. ■

For example, the transmission network in Fig. 2 is acyclic, and its mcc has size 6. By Theorem 2.1, we have $r = (C \cdot 1 + C \cdot 1)/6 = C/3$, and the achievable throughput for each session is $r/2 = C/6$.

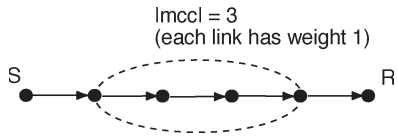


Fig. 3. Achievable throughput of a single route is $C/3$.

III. Data Dissemination

Data dissemination refers to the form of data transmission where information is being propagated from one source to one or more destinations within the network. Both unicast and multicast belong to this category. In a unicast session, data are transmitted from a single source to a single destination; in a multicast session, identical data are transmitted from one source to multiple destinations. In this section, we examine mechanisms that may be used to increase the throughput of unicast and multicast sessions, including 1) one-hop away multipath and 2) network coding.

A. Unicast

Consider a single route that serves a multihop unicast session. In wireline networks, if all links have capacity C and there is no background traffic, the throughput of the unicast session is able to achieve C as well, because all links along the route can be active concurrently. In comparison, in wireless *ad hoc* networks where all radios have capacity C, even in the absence of background traffic, the achievable session throughput r is only $C/3$, because the mcc has size 3, as shown in Fig. 3. The underlying intuition is that, due to intraroute spatial contention, only one out of every three links can be transmitting at a given

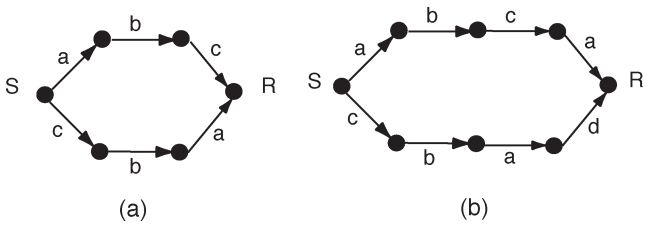


Fig. 4. Two one-hop away routes can achieve $2C/3$ or $C/2$. (a) When total number of subflows is a multiple of 3, achievable throughput is $2C/3$. (b) When total number of subflows is not a multiple of 3, achievable throughput is $C/2$.

time, and the radio at the source is sending data during one third of the time.

The above example shows that one route is not sufficient to effectively utilize the available channel capacity at the source. We argue that multipath routing can be employed to break through the $C/3$ bound by taking advantage of the wireless advantage (the broadcasting nature) at the source. As we will show, two routes may bring the throughput for a 1-to-1 unicast session up to $2C/3$, and adding more routes may achieve a throughput as high as $5C/6$. Existing research [12]–[15] on wireless multipath routing has been focusing on load balancing and fault tolerance, as it has been the case in wireline networks. To achieve these two goals, the set of routes being chosen are usually required to be disjoint, where two routes do not share a common node beside the end nodes, or partially disjoint, which is a weaker requirement that allows two routes to intersect at some intermediate nodes. However, intense contention may still exist among links from disjoint or partially disjoint routes. We argue that to reduce interroute interference and, therefore, achieve a higher session throughput, the transmission routes need to be one-hop away.

Fig. 4 shows examples where two one-hop away routes are used to transmit data between one pair of source and destination. In cases where the total number of hops on both routes is a multiple of 3, all subflows can be scheduled without interference in three equal-length phases, a, b, and c. Therefore, the achievable throughput $r = C\sum_{i\in S} t_i/T = 2C/3$. In cases where the total number of subflows is not a multiple of three, it takes four phases to schedule all of them, achieving a throughput of $C/2$. Assuming the number of hops on a route is a uniformly distributed random variable, the expected throughput is then $2/3C \cdot 1/3 + 1/2C \cdot 2/3 = 5C/9$, which is a 66.7% improvement over the achievable throughput in the single route case.

If we further increase the number of one-hop away routes, the achievable throughput can be further increased, with decreasing amounts of improvement. When three routes are used, it takes four or five phases to schedule all the subflows, and r is $3C/4$ or $3C/5$. Similarly, for four routes, r is $4C/5$ or $4C/6$. This pattern of improvement stops when the number of routes is beyond five, because a wireless node can have at most five one-hop away neighbors, as shown in Fig. 5. Therefore, the throughput of a single unicast session is bounded by $5C/6$.

In cases where a source has data to transmit to multiple unicast destinations, one-hop away multipath routing may also be applied to increase the achievable throughput. The underlying topology of one-hop away multipath from one source to multiple destinations is a tree. The mcc can always be identified around the source, and $|\text{mcc}| = k + 1$, where k is the number of routes used. Therefore, by Theorem 2.1, the achievable throughput $r = C\sum_{i\in S} t_i/|\text{mcc}| = kC/(k+1)$, for $k \leq 5$. Fig. 6 shows cases where $k = 2$ and $k = 3$. When $k > 5$, again, the achievable throughput is always $5C/6$ due to the bound on the number of one-hop away neighbors around the source.

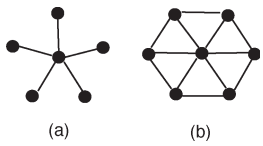


Fig. 5. Upper bound on the number of one-hop away neighbors. (a) Five one-hop away neighbors: Possible. (b) Six one-hop away neighbors: Impossible.

Essentially, the above approach "interleaves" unicast sessions that would otherwise need to be transmitted sequentially, without using the multipath strategy at the source. As we have shown, this can utilize the radio capacity at the source more efficiently and, consequently, reduces both the transmission time for all the unicast sessions as a whole and the average completion time for each single session.

Finally, we provide a practical mechanism for finding multiple one-hop-away paths from the sender S to the receiver T. We take an iterative approach, in which a new $S \rightarrow T$ path is constructed during each round. Essentially, we employ source routing to find each one-hop away path, avoiding using nodes that have appeared in or are within one-hop distance to paths previously established. Synchronization among rounds can be coordinated by the sender and the receiver: A new round i starts when S broadcasts a route request message RReq_i, and it ends when S receives a route reply message RRpl_i. We describe the algorithm in Table I.

In the table, $\text{fresh}_i(u)$ denotes whether u has seen a route request message of round i already or not, $\text{serve}_i(u)$ indicates whether u serves on the $S \rightarrow T$ path constructed during round i, and $\text{nearby}_i(u)$ indicates whether u is within one-hop range to the $S \rightarrow T$ path constructed during round i. Δ is an adjustable time window the receiver T waits in each round for route request messages to arrive, and k is the total number of one-hop-away paths to be constructed. A dynamic source routing (DSR)-like protocol is used to set up the first route. When the second route request packet is sent out by the source, any node that is already on the first route or is a neighboring node of the first route will refrain itself from further relaying the route request packet. Therefore, the route request message can arrive at the destination only via routes that are one-hop away from the first route. A second one-hop away route may be set up after the source has received the route reply message from the destination.

B. Multicast

As previously noted, we focus on source–end throughput. In a multicast session, this translates to the throughput at the single source, although each data packet being multicasted is received by multiple destinations. Similar to the case of a single unicast session, the achievable throughput of a single multicast

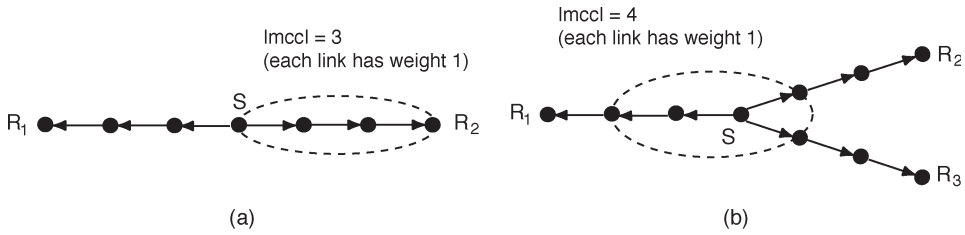


Fig. 6. 1-to-k unicast can achieve a throughput of $kC/(k+1)$ at the sender, for $k \leq 5$. (a) Two one-hop away routes achieve $2C/3$. (b) Three one-hop away routes achieve $3C/4$.

TABLE I
CONSTRUCTING ONE-HOP-AWAY PATHS

Initialization

$\forall u, \forall i$, $fresh_i(u)$ = TRUE
$serve_i(u)$= FALSE, $nearby_i(u)$= FALSE

In each round $i \in [1..k]$**, at each node** u**:**

if $u = S$, then *broadcast* $RReq_i$ containing empty path
if $u = T$, then
  count t_i to Δ
  if $fresh_i(u)$ = TRUE then terminate
  choose best route p_i
  broadcast $RRpl_i$ *containing* p_i
if $u \notin \{S, T\}$ then
  upon receiving $RReq_i$
    if $\exists 1 \leq j < i$ such that $serve_j(u)$= TRUE or $nearby_j(u)$ = TRUE then ignore message
    if $fresh_i(u)$ = FALSE then *ignore message*
    expand path in $RReq_i$ *with* u
    further broadcast $RReq_i$
    $fresh_i(u)$ = FALSE
  upon receiving $RRpl_i$
    if $u \in p_i$ and $RRpl_i$ *is from upstream neighbor in* p_i then
      $serve_i(u)$ = TRUE
      further broadcast $RRpl_i$
    if $u \notin p_i$ then
      $nearby_i(u)$ = TRUE

session is also bounded by $C/3$. The strategy of using one-hop away multipaths can be extended to include multicast sessions. In the scenario where a single source has multiple concurrent unicast and multicast data to transmit, the throughput may be increased by activating one-hop away routes to reach the respective destinations, similar to the previously discussed cases with multiple unicast destinations. Fig. 7 shows an example in which the source S activates one unicast session and two multicast sessions, where the total throughput at the source is increased to $3C/4$. Note that subflows with the same sender in a multicast tree can be scheduled concurrently because they are transmitting identical data that may be broadcast by the sender to multiple receivers. Therefore, the corresponding links are regarded as one unit in finding mcc and computing |mcc|. Such links are connected by a curve segment in the figures.

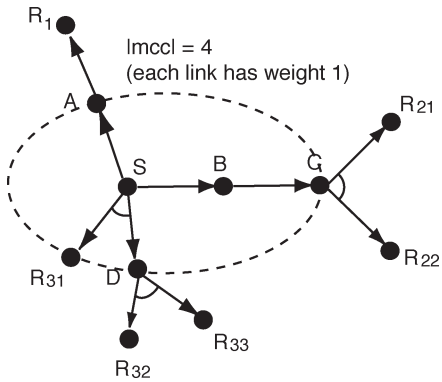


Fig. 7. One source node that is transmitting one unicast session, $S \rightarrow R_1$, and two multicast sessions, $S \rightarrow R_{2i}$ and $S \rightarrow R_{3i}$, concurrently. Achievable total throughput is $3C/4$.

Further, we discuss the effects of branching points in the multicast tree on throughput. In a multicast session, identical data are transmitted to each receiver. Incoming packets at a branching node are merely replicated into multiple copies and relayed further. Therefore, the strategy of branching early and maintaining multiple one-hop away branches will not increase the throughput of a multicast session compared with the strategy of branching late, because the multiple routes are only used to transmit redundant data in early branching. This leads to a waste of bandwidth rather than an improvement of throughput.

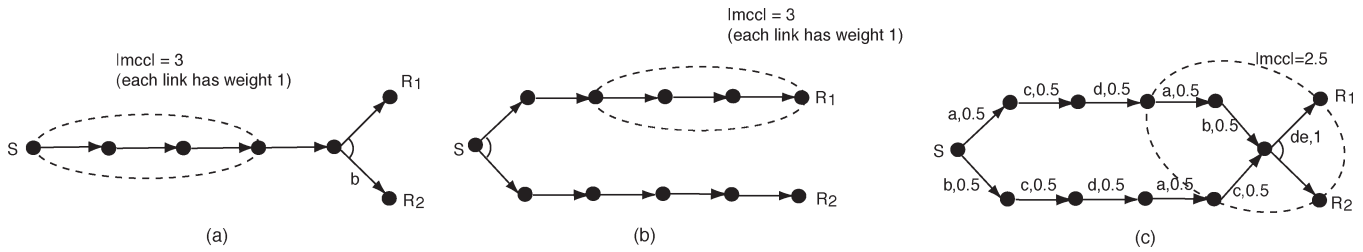


Fig. 8. (a) Late branching, (b) early branching, and (c) late branching with two one-hop away routes.

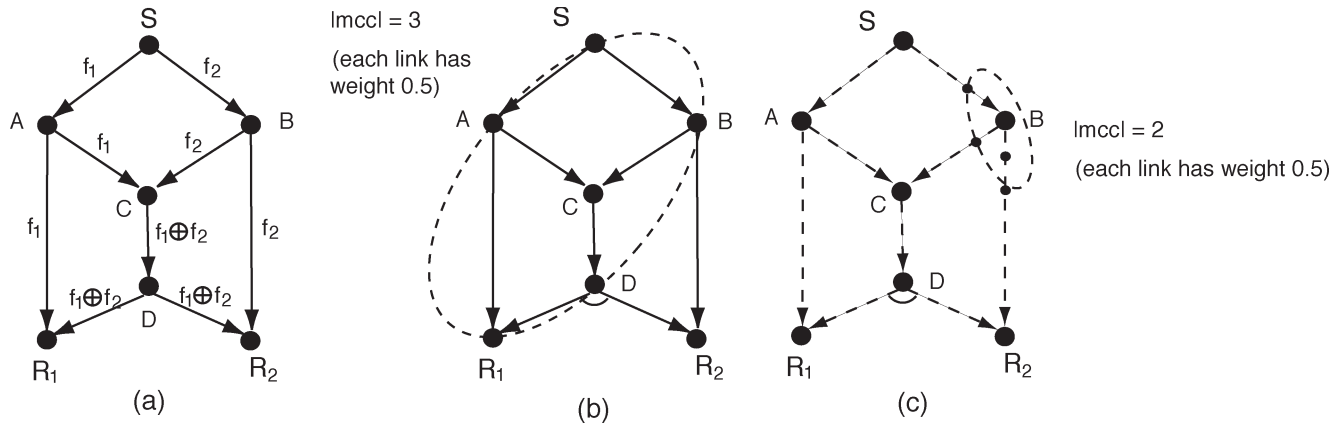


Fig. 9. Effects of coding in (a) dense and (b) sparse wireless *ad hoc* networks.

As shown in the example in Fig. 8(b), if we branch immediately at the source, and then transmitting (identical) data to the two destinations along two one-hop away routes, a throughput of $C/3$ can be achieved. In comparison, branching at the last hop [shown in Fig. 8(a)] achieves $C/3$ as well and consumes only approximately half of the bandwidth as that of early branching.

In the case of late branching, if we use multiple one-hop away routes to "strengthen" the longer routes before late branching, the session throughput may be increased. In the example of Fig. 8(c), if the sender transmits (independent) data along two one-hop away routes to the node that broadcasts it to the destinations, we may prove that r can be increased to $2C/5$: It is possible to transmit $C \cdot (1\ \text{s})$ data from the source to each destination within a scheduling period consisting of five phases, $a-e$, each of length 0.5 s. Therefore, $r \geq 2C/5$. Furthermore, the mcc has size 2.5, which implies that $r \leq 2C/5$. Therefore, $r = 2C/5$.

C. Network Coding

It is not always possible for a source to find one-hop away routes, especially for multicast, because a multicast tree usually spans a broader range around the source. Furthermore, because the secondary routes may not be as short as the primary route, one-hop away multipath routing pays a price in network bandwidth. We examine a different mechanism, "network coding," proposed in the area of information theory for multicast sessions in wireline networks [6], [7]. As opposed to multipath routing, network coding does not usually lead to a transmission network that spans a larger geographical range; also, it usually consumes less bandwidth rather than more.

Network coding (henceforth referred to as "coding") is a strategy to increase end-to-end throughput, in which bits of data are not merely treated as "atoms" that may only be replicated and forwarded in intermediate nodes; rather, data may be coded before being forwarded further. Coded data may be decoded by a downstream or destination node, based on its knowledge of the coding strategy.

Fig. 9(a), an example taken from previous work [6], shows how coding facilitates the increase of throughput in a 1-to-2 multicast session in wireline networks. In the figure, f_1 and f_2 represent two independent data flows originating from the source S, and $f_1 \oplus f_2$ represents the flow resulted from taking a bit-wise exclusive-or on f_1 and f_2. Node C transmits the coded flow $f_1 \oplus f_2$ along the "bottleneck" link CD to node D, which then forwards the coded flow to both destinations R_1 and R_2. Note that R_1 receives f_1 as is; it may decode $f_1 \oplus f_2$ by taking another bit-wise exclusive-or operation because $f_1 \oplus (f_1 \oplus f_2)$ recovers f_2. R_1 has then successfully received both f_1 and f_2. Similarly, R_2 is able to decode $f_1 \oplus f_2$ and obtain complete knowledge about both f_1 and f_2 as well. Therefore, the session achieves a throughput of $2C$ (assuming each link has capacity C), which is impossible with data forwarding and replication only.

However, it is "not" as advantageous to apply coding in *ad hoc* networks, especially for small and dense ones. This is due to the different contention model used for wireless transmissions. First, applying coding always involves a more complicated cyclic transmission topology because coding yields no improvement on trees. Second, applying coding also involves nonidentical data flows, and certain nodes must transmit different data flows along different outgoing links. Compared with the case where data are transmitted along a

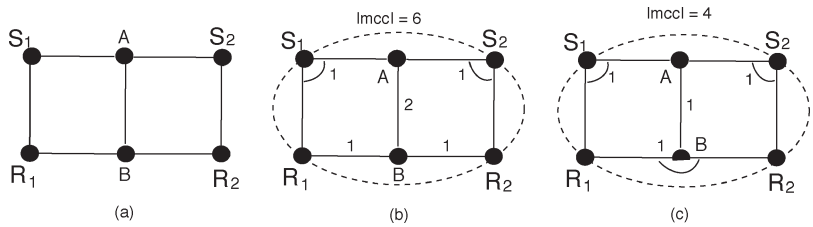


Fig. 10. Two multicast sessions and upper bounds of their total throughput with and without coding. (a) Two multicast sessions. (b) Without coding, $|\text{mcc}| = 6$, r is bounded by $2C/6 = C/3$. (c) With coding, $|\text{mcc}| = 4$, r is bounded by $2C/6 = C/2$.

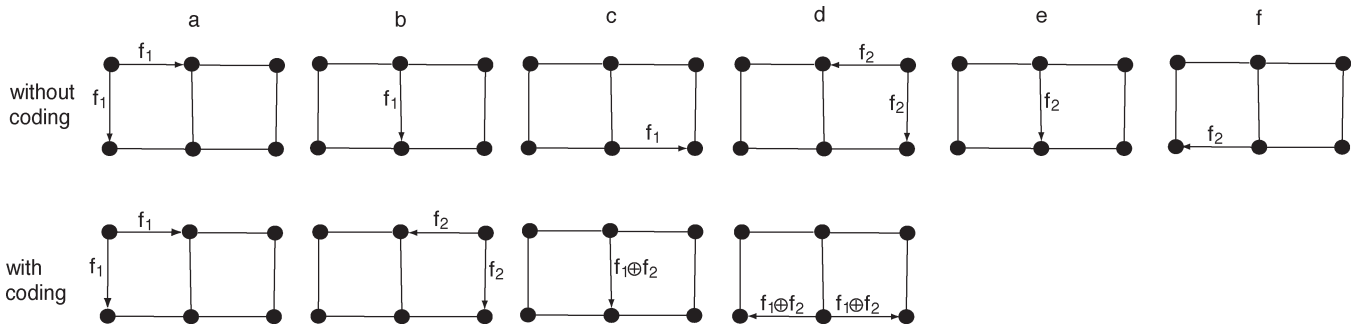


Fig. 11. Without coding, achievable total throughput is $C/3$; with coding, achievable total throughput is $C/2$.

multicast tree without coding, both facts above lead to more intense spatial contention in wireless *ad hoc* networks. We emphasize again that, according to the wireless advantage, outgoing subflows at the same node in a multicast tree do not contend with each other.

Consider the same multicast session as in the previous wireline example, but in wireless *ad hoc* networks. If the same coding strategy is used, the size of mcc is 3, as shown in Fig. 9(b). Therefore, the achievable throughput is bounded by $C/3$. In comparison, it is easy to verify that a straightforward multicast tree without coding using routes $S{-}A{-}R_1$ and $S{-}B{-}R_2$ is able to achieve a throughput of $C/2$. In this example, the disadvantage of spatial contention in wireless networks overshadows the advantage of coding.

Nevertheless, we observe that the advantage of coding to increase throughput can outweigh the disadvantage introduced by spatial contention if 1) transmission network is large and sparse; or 2) spatially nearby multicast sessions exist concurrently. If the transmission network is large and sparse, spatial contention is not intense. A sparse transmission network can have $|\text{mcc}|$ as small as 4, whereas the $|\text{mcc}|$ of a multihop multicast tree is 3. The difference is much less than the case of a small and dense transmission network. Fig. 9(c) shows such an example. The topology is similar to that in Fig. 9(b), and each link in Fig. 9(b) is replaced by a multihop route (shown as a dashed line). For this multicast session, it may be easily verified that the achievable throughput without coding is $C/3$; with coding, it can be as high as $C/2$.

In the case where multiple spatially nearby multicast sessions exist simultaneously, throughput of the straightforward multicast tree approach (without coding) drops dramatically due to intertree spatial contention. In comparison, we have nonidentical flows being transmitted on a cyclic transmission network "automatically"; coding no longer comes with a price. Therefore, it is more likely that coding may facilitate the increase of throughput.

Below, we illustrate the above observations with a concrete example. In this example, two multicast sessions are placed on a small-scale wireless topology in an interleaved way. Network coding reduces contention both by taking advantage of the broadcast nature of wireless transmission (the wireless advantage) and by reducing the amount of data transmitted at the bottleneck link. Recall that we assume the protocol interference model among wireless link flows, i.e., two link flows may transmit concurrently only if neither node in one flow is within one-hop distance from a node in the other flow.

As shown in Fig. 10(a), two 1-to-2 multicast sessions coexist in an *ad hoc* network. S_1 and S_2 are the two senders, and the destinations are R_1 and R_2 in both sessions. Fig. 11 shows two schedules for both sessions. Without coding, all subflows can be scheduled using six equal-length phases. We can verify that this schedule is optimal by noting the following fact: No two steps in the schedule can be merged into one without introducing interference according to the protocol interference model. With coding, the number of phases is reduced to four. One cause for the improvement is the activation of coding at node A, which reduces the amount of data forwarded on the bottleneck link AB by half. This also reduces the total bandwidth usage in the entire session. Another reason is the fact that, with respect to the subflows BR_1 and BR_2, in the case without coding, f_1 and f_2 have to be scheduled in different phases; however, in the case of using coding, the same data, $f_1 \oplus f_2$, may be broadcast in one phase to both destinations. From Fig. 11, we know that without coding, the achievable throughput $r \geq 2C/6 = C/3$; with coding, $r \geq 2C/4 = C/2$. Furthermore, Fig. 10(b) and (c) shows that $C/3$ and $C/2$ are actually the maximum achievable throughput for those two cases, respectively. It may then be proved that,

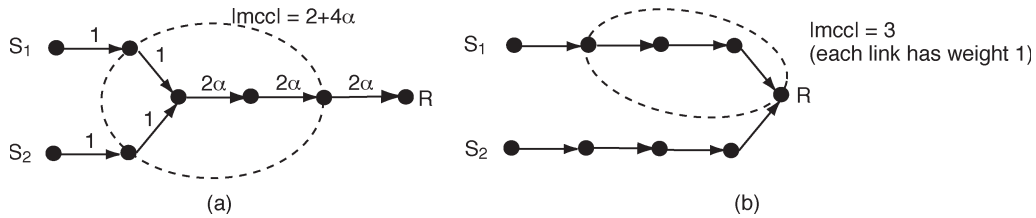


Fig. 12. Early aggregation versus late aggregation in a 2-to-1 data aggregation session. (a) Early aggregation, $r = C/(1 + 2\alpha)$. (b) Late aggregation, $r = 2C/3$.

without coding, $r = C/3$; with coding, $r = C/2$. In this case, coding increases the total throughput by 50% in wireless *ad hoc* networks.

We have shown that, although coding is not as advantageous in the cases where it works well in wireline networks, it does help to increase throughput when multiple multicast sessions are present. Inasmuch as the cases where coding may be applied involve small-scale topologies, it is easy to identify patterns showing these topologies in multicast sessions and to promptly activate coding. Coupled with the strategy of late branching and one-hop away routes before the branching point, we believe that end-to-end throughput in the case of multicast data dissemination may be increased with adequate strategies.

IV. Data Aggregation

Data aggregation refers to the form of data transmission in which data from multiple sources is transmitted toward a common destination. For example, Estrin *et al.* have studied data aggregation in wireless sensor networks [15], [16], where data corresponding to physical events observed by sensors are routed toward one common "data sink," possibly a gateway or data processing node. The data flows that are transmitted toward the sink can be independent or correlated. Consequently, when two flows merge at an intermediate node, that node may be able to combine the incoming flows and reduce the amount of data being further relayed. The ratio of the amount of combined data after aggregation over the amount of uncombined data before aggregation is referred to as the "aggregation ratio," denoted as α. For two separate units of data that come from each of the incoming flows, respectively, the amount of aggregated data at the outgoing link, 2α, usually ranges between 1 and 2, depending on the specific application and the amount of knowledge (such as application semantics) that a node has about the data flows. Correspondingly, the range of α is between 0.5 and 1. We call the $\alpha = 0.5$ case "perfect aggregation" and that of the $\alpha = 1$ case "zero aggregation." The flows that enter the sink are called "final flows." Intermediate nodes at which flows aggregate are called "aggregation nodes."

Due to the presence of data compression upon aggregation, the total amount of data that leaves the sources may not be equal to the amount of data that arrives at the sink. These two amounts are equal only in the zero aggregation case; otherwise, the source-side amount is larger than the destination-side amount. Again, we focus on the sources and consider the summation of the transmission rate at each source as the throughput of the data aggregation session.

For the same data aggregation session, the routing algorithm may decide to aggregate flows earlier near the sources or later near the sink. These are called "early aggregation" and "late aggregation," respectively. As being pointed out by Intanagonwiwat *et al.* [16], the tradeoffs between early aggregation and late aggregation include the following:

- — early aggregation may reduce the overall amount of data being transmitted and, therefore, reduce the total amount of energy consumption;
- — late aggregation is more robust because the loss of nonaggregated packets is less severe than the loss of aggregated packets;
- — early aggregation may introduce a higher latency.

We examine another dimension of the tradeoff, from the perspective of increasing throughput, and show that the value of α and the number of source flows n both play critical roles in determining which form of data aggregation can achieve a higher throughput. We first examine how the tradeoff varies as the number of flows increases. We show that from the point of view of increasing throughput, late aggregation is more suitable for very small number of sources; as the number of sources increases, early aggregation starts to outperform late aggregation over a certain range of α, and the range is getting wider and wider.

The concepts of early aggregation and late aggregation are rather vague. To make a comparison, we consider the rather extreme cases of them: For early aggregation, we consider the case where all data flows merge into one final flow before entering the sink; for late aggregation, we consider the case where all data flows are final flows and meet at the sink without previous aggregation. To analyze the maximum achievable throughput, we make the following two assumptions to reduce contention: 1) Flows aggregate along one-hop away paths; and 2) aggregation nodes are one-hop away from one another. Also, in cases where the number of sources is large, we assume aggregation is done in a balanced way, i.e., two branches in the aggregation tree contain roughly the same number of sources before they aggregate.

Fig. 12 shows an aggregation session with two sources. With early aggregation, the size of mcc is $2 + 4\alpha$, and $r = 2C/(2 + 4\alpha) = C/(1 + 2\alpha)$. Inasmuch as $\alpha \in [0.5, 1]$, $r \in [C/3, C/2]$. With late aggregation, $r = 2C/3$, similar to the 1-to-2 independent unicast case. Therefore, late aggregation can achieve a higher throughput than early aggregation in the case involving two sources, regardless of α.

However, this is not always the case. As the number of source flows increases, the achievable throughput of late aggregation soon increases to $5C/6$ where it stops, whereas the achievable throughput of early aggregation keeps increasing, and depending on α, it may soon become higher than C.

TABLE II
COMPARISON OF EARLY AGGREGATION AND LATE AGGREGATION

# of sources	r_{early}	r_{late}	range of α s.t. $r_{\text{early}} \geq r_{\text{late}}$
2	$\frac{C}{1+2\alpha} \in [\frac{C}{3}, \frac{C}{2}]$	$\frac{2C}{3}$	ϕ
3	$\frac{C}{(1+2\alpha)^2} \in [\frac{C}{3}, \frac{3C}{4}]$	$\frac{3C}{4}$	$\{0.5\}$
4	$\frac{C}{\alpha(1+2\alpha)} \in [\frac{C}{3}, C]$	$\frac{4C}{5}$	[0.5, 0.58]
5	$\frac{5C}{4\alpha^3+8\alpha^2+3\alpha} \in [\frac{C}{3}, \frac{5C}{4}]$	$\frac{5C}{6}$	[0.5, 0.63]

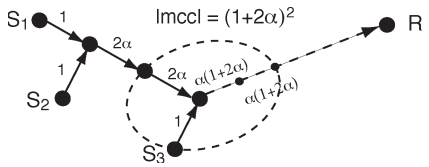


Fig. 13. Early aggregation on a 3-to-1 aggregation session: $|\text{mcc}| = (1+2\alpha)^2$, $r = 3C/(1+2\alpha)^2$.

Table II shows the corresponding values of r that we are able to derive for early aggregation and late aggregation in 2-to-1, 3-to-1, 4-to-1, and 5-to-1 aggregation sessions, respectively. Similar to the 1-to-n unicast cases, the achievable throughput for n-to-1 late aggregation is $nC/(n+1)$, for $n \leq 5$. The analysis on the achievable throughput using earlier aggregation for the $n > 2$ cases is similar to that of the $n = 2$ case. Given that aggregation nodes are one-hop away from each other and flows aggregate in a balanced way, the mcc is always identified around the aggregation node on the final flow. For example, Fig. 13 shows the $n = 3$ case.

As we can observe from the table, the value of r for early aggregation ranges from $C/3$ to $nC/4$, which correspond to zero aggregation and perfect aggregation, respectively. For zero aggregation, each unit of data leaving a source corresponds to one unit of data that needs to be transmitted along the final route. The throughput of the session is bounded by the throughput of the final route, $C/3$. For perfect aggregation, the quantity of an aggregated flow is the same as each of the flows being aggregated. Therefore, the load is equal across all the links and routes. As shown in Fig. 14, the mcc of such a transmission network has size 4, and the achievable throughput $nC/4$ can easily break through the bound of the sink's receiving capacity C. The intuition of this is that, in perfect early aggregation, one unit of data transmitted along the final route corresponds to multiple unit of data transmitted by the sources. To illustrate this, we have labeled subflows in Fig. 14 with their actual throughput and the corresponding source–end throughput.

When the number of sources grows beyond 5, the throughput that is achievable by late aggregation is always $5C/6$. Early aggregation outperforms late aggregation on an even larger range of α. However, in early aggregation, aggregating all flows onto one final flow cannot effectively utilize the radio capacity at the sink; in late aggregation, letting all flows enter the sink directly (thus become final flows) gives up the opportunity of aggregating them onto more "dense" flows that may help reduce the contention around the sink.

In what follows, we examine the impact that "the number of final flows" (denoted as k) has on the achievable session throughput in cases where $n \gg 1$ and show that, generally, neither $k = 1$ nor $k = n$ is the optimal choice.

We use r_k to denote the achievable session throughput of a data aggregation session with k final flows. Fig. 15 shows the cases for $k = 2$ and $k = 3$. Note that with balanced aggregation, each subflow of a flow aggregated from m sources has weight $(2\alpha)^{\log_2 m}$. Therefore, we can derive r_k as follows:

$$
\begin{aligned}
r_1 &= \frac{nC}{2(2\alpha)^{\log_2 n} + 2(2\alpha)^{\log_2 n - 1}} \\
&= \frac{\alpha}{1+2\alpha} n^{-\log_2 \alpha} C \in \left[\frac{C}{3}, \frac{nC}{4}\right] \\
r_2 &= \frac{nC}{2(2\alpha)^{\log_2 \frac{n}{2}} + 2(2\alpha)^{\log_2 \frac{n}{2} - 1}} \\
&= \frac{2\alpha^2}{1+2\alpha} n^{-\log_2 \alpha} C \in \left[\frac{2}{3}C, \frac{n}{4}C\right] \\
r_k(k = 3, 4, 5) &= \frac{nC}{(k+1)(2\alpha)^{\log_2 \frac{n}{k}}} \\
&= \frac{k}{k+1} \left(\frac{n}{k}\right)^{-\log_2 \alpha} C \in \left[\frac{k}{k+1}C, \frac{n}{k+1}C\right] \\
r_k(k \geq 6) &= \frac{nC}{6(2\alpha)^{\log_2 \frac{n}{k}}} \frac{5}{k} \\
&= \frac{5}{6} \left(\frac{n}{k}\right)^{-\log_2 \alpha} C \in \left[\frac{5}{6}C, \frac{5n}{6k}C\right].
\end{aligned}
$$

Fig. 16 plots the throughput computed as above when $n = 20$, against the value of α. From the above figure and formulas, we can see that in all cases, the achievable session throughput r decreases as α increases. The decreasing speed is higher for $\alpha \in [0.5, 0.75]$ than for $\alpha \in [0.75, 1]$. The highest achievable throughput for an n-to-1 data aggregation session is $nC/4$, which is realized when $\alpha = 0.5$ and $k \leq 3$. The session shown in Fig. 14 is such an example. When α is very small (near 0.5), r decreases as k increases; when α is very large (near 1), very small values of k performs worse than the larger values of k, but the difference between different choices of k is rather moderate, compared with the case where α is small. The range of r_1 agrees with what has been computed in Table II. The range of α for which early aggregation with one final route outperforms late aggregation has increased to [0.5, 0.8]. For a wide range of α, $k = 3$ performs quite well. It dominates the other choices except for very large α, in which case the difference is moderate.

To conclude, we have examined how the achievable throughput r of a data aggregation session is affected by the topology of the aggregation tree and have shown that the appropriate design of the topology depends crucially on the aggregation ratio α, as well as on the number of sources. In particular, we have shown that, for very small number of sources, using one-hop away routes to transmit data directly to the sink without aggregation is the best choice regardless of α; otherwise, to achieve a higher throughput, source flows need to aggregate to a smaller number of final flows before reaching the sink, except for cases where the aggregation ratio is very high (near 1). With respect to the number of final flows that enter the sink, "three" is usually a good choice: It utilizes the receiving capacity at the sink

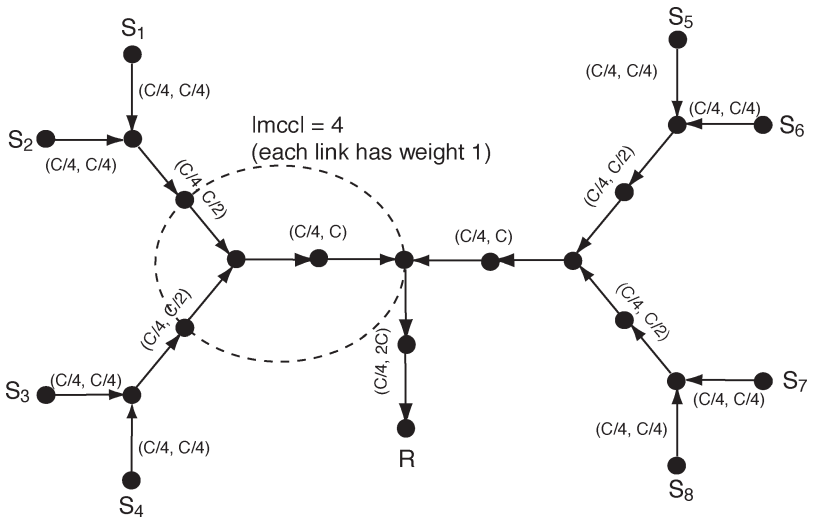


Fig. 14. Perfect 8-to-1 early aggregation session, $r = 2C$.

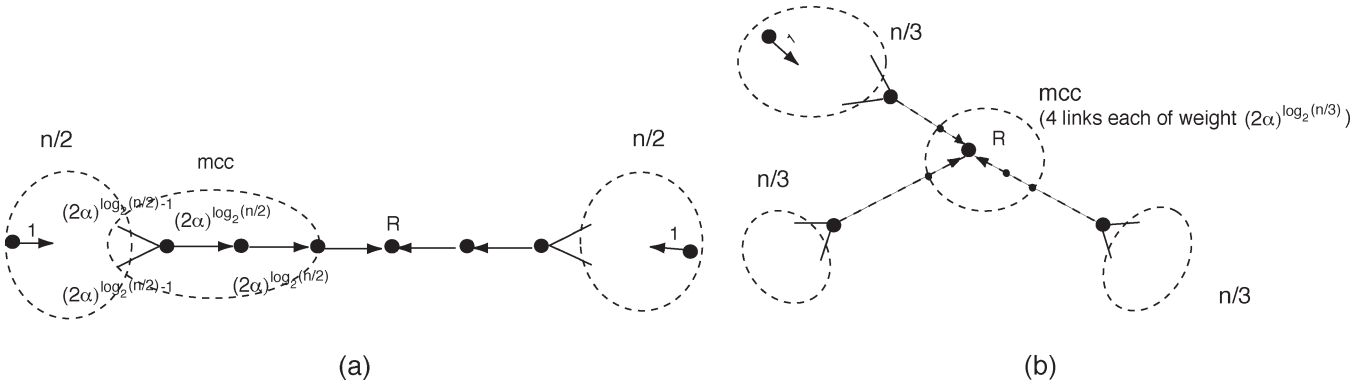


Fig. 15. Number of flows entering the data sink. (a) Two final flows. (b) Three final rows.

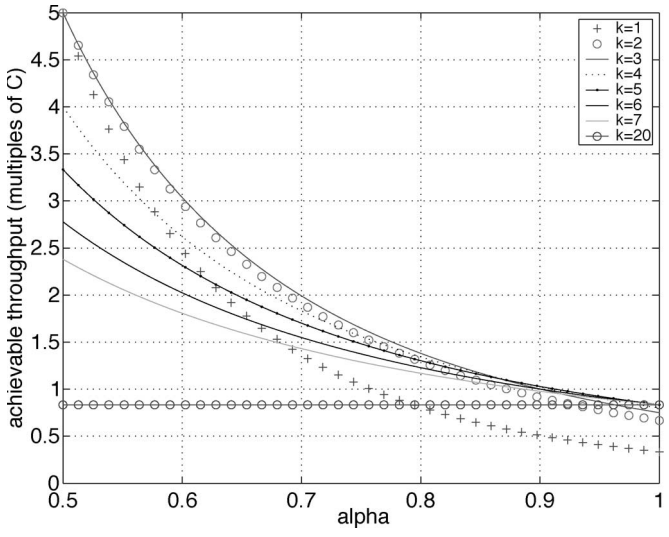


Fig. 16. Achievable throughput versus number of final flows.

quite well because it 1) allows source flows to aggregate into more "condensed flows" before being received by the sink and 2) allows the sink to receive data from different final flows in an interleaving way to reduce its idling time. If the number is too small, the single final flow forms a throughput bottleneck; if the number is too large, the intense contention around the sink forms the throughput bottleneck.

V. Related Work

In the Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) category of MAC protocols, data transmission is preceded by handshaking of control packets (RTS/CTS) [17], [18]. Nodes within the neighborhood of either the sender or the receiver of a transmitting link have to defer transmission to avoid collision. Therefore, two independent local transmissions will interfere if the sender or receiver of one transmission is within one-hop range of the sender or receiver of the other transmission [10], [11].

Classical studies of multipath routing in wireline networks has been focusing on the objectives of load balancing and fault tolerance [19]–[23]. Increasing end-to-end throughput is neither a design goal nor a major advantage of wireline multipath routing. Research in wireless multipath routing so far has been focusing on the same direction. Both the issues of load balancing [12], [13] and fault tolerance [14], [15] have been examined. We apply multipath routing explicitly toward the goal to counteract the unique intraroute interference in wireless *ad hoc* networks that leads to a reduced end-to-end throughput.

Network coding was first proposed and studied by Ahlswede *et al.* in the context of wireline networks [6]. It has been shown that applying coding over a multicast network may increase its capacity. Koetter and Médard then examined network coding from an algebraic perspective [7]. In this paper, we apply network coding to decrease medium contention in wireless networks and, therefore, to increase transmission throughput.

Estrin *et al.* has studied data aggregation in wireless sensor networks [16], [24]. The focus is to reduce energy consumption due to data transmission. It is shown that constructing the most energy-efficient aggregation tree is nondeterministic polynomial-time hard (NP-hard). Several heuristic solutions have been proposed.

The capacity of *ad hoc* networks has been studied in previous works [2], [3], [5], where the focus is the traffic-forwarding capability of the *ad hoc* network as a whole, under certain traffic patterns. We analyze and attempt to increase the capacity of a "part of the network" that is transmitting data for the session(s) of interest. The insights from our studies have a direct influence on the throughput and completion time of a session, especially when the network is lightly loaded.

In this paper, we discussed potential approaches that heuristically increase throughput. For discussions on how to approach the absolutely maximum throughput in wireless *ad hoc* networks, we refer to a cross-layer optimization framework presented in [25]. Jain *et al.* [26] and Kodialam and Nandagopal [27] also proposed approximate mechanisms to achieve maximum unicast throughput in wireless *ad hoc* network.

A parallel goal of improving throughput is reducing transmission cost. To achieve the fixed end-to-end data transmission rate, we would like to incur the minimum cost possible. In wireless *ad hoc* networks, cost typically present in the form of bandwidth consumption or radio power consumption because both the wireless medium and battery energy are scare resources. From an optimization perspective, achieving high throughput and achieving low transmission cost are primal and dual problems to each other. Lun *et al.* [28] proposed a mathematical programming framework for cost minimization in wireless networks. Their work may handle both unicast and multicast transmissions, with network coding considered in the later case.

VI. Conclusion

We illustrate in this paper that, using strategies that include 1) multiple end-to-end paths, 2) per-node algorithms such as coding, and 3) rearranging transmission network topologies, it is feasible and practical to increase data throughput in various scenarios of wireless communications. Although we concur that the overall network capacity of *ad hoc* networks is not scalable when the number of nodes increases, we believe that adopting the best possible strategy based on the insights in this paper may help to alleviate such problems. As part of our future work, we aim to design distributed algorithms to approximate the theoretical strategies in this paper in all three cases, so that at any given time, a flow may enjoy the best possible end-to-end throughput. We are also interested in studying the effects of greedy behavior in *ad hoc* networks and seek to maintain equilibriums with the presence of aggressive behavior on each of the flows.

References

[1] J. Wieselthier, G. Nguyen, and A. Ephremides, "On the construction of energy-efficient broadcast and multicast trees in wireless networks," in *Proc. IEEE INFOCOM*, 2000, pp. 585–594.

[2] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.

[3] J. Li, C. Blake, D. Couto, H. Lee, and R. Morris, "Capacity of ad hoc wireless networks," in *Proc. ACM MobiCom*, Sep. 2001, pp. 61–69.

[4] P. Gupta, R. Gray, and P. R. Kumar, "An experimental scaling law for ad hoc networks," Univ. Illinois, Urbana-Champaign, 2001. [Online]. Available: http://black.csl.uiuc.edu/~prkumar/ps_files/exp.pdf

[5] P. Gupta and P. R. Kumar, "Internets in the sky: The capacity of three dimensional wireless networks," *Commun. Inf. Syst.*, vol. 1, no. 1, pp. 33–49, Jan. 2001.

[6] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[7] R. Koetter and M. Médard, "Beyond routing: An algebraic approach to network coding," in *Proc. IEEE INFOCOM*, 2002, pp. 122–130.

[8] S. Y. R. Li and R. W. Yeung, "Network multicast flow via linear coding," in *Proc. Int. Symp. Oper. Res. Appl.*, 1998, pp. 197–211.

[9] T. Ho and R. Koetter, "A coding view of network recovery and management for single-receiver communications," in *Proc. Conf. Inf. Sci. Syst.*, 2002, pp. 590–597.

[10] H. Luo, S. Lu, and V. Bharghavan, "A new model for packet scheduling in multihop wireless networks," in *Proc. ACM MobiCom*, 2000, pp. 76–86.

[11] H. Luo, P. Medvedev, J. Cheng, and S. Lu, "A self-coordinating approach to distributed fair queueing in ad hoc wireless networks," in *Proc. IEEE INFOCOM*, 2001, pp. 1370–1379.

[12] M. R. Pearlman, Z. J. Hass, P. Sholander, and S. S. Tabrizi, "On the impact of alternate path routing for load balancing in mobile ad hoc networks," in *Proc. IEEE/ACM Mobihoc*, 2000, pp. 3–10.

[13] K. Wu and J. Harms, "On-demand multipath routing for mobile ad hoc networks," in *Proc. 3rd Eur. Pers. Mobile Commun. Conf.*, 2001, pp. 1–7.

[14] A. Nasipuri and S. R. Das, "On-demand multipath routing for mobile ad hoc networks," in *Proc. ICCCN*, 1999, pp. 64–70.

[15] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-resilient, energy-efficient multipath routing in wireless sensor networks," *Mobile Comput. Commun. Rev.*, vol. 1, no. 2, pp. 1–13, 2002.

[16] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," in *Proc. ICDCS*, 2001, pp. 457–458.

[17] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A medium access protocol for wireless LANs," in *Proc. ACM SIGCOMM*, 1994, pp. 212–225.

[18] LAN MAN Standards Committee of the IEEE Computer Society, *IEEE 802.11: Wireless LAN MAC and PHY Specifications, Chapter 11*, 1999.

[19] R. Krishnan and J. A. Silvester, "Choice of allocation granularity in multipath source routing schemes," in *Proc. IEEE INFOCOM*, 1993, pp. 322–329.

[20] D. Sidhu, R. Nair, and S. Abdallah, "Finding disjoint paths in networks," in *Proc. ACM SIGCOMM*, 1991, pp. 43–51.

[21] S. Murthy and J. J. Garcia-Lunna-Aceves, "Congestion-oriented shortest multipath routing," in *Proc. IEEE INFOCOM*, 1996, pp. 1028–1036.

[22] J. Chen and D. Subramanian, "An efficient multipath forwarding method," in *Proc. IEEE INFOCOM*, 1998, pp. 1418–1425.

[23] I. Cidon, R. Rom, and Y. Shavitt, "Analysis of multi-path routing," *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, pp. 885–896, Dec. 1999.

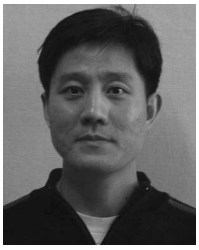
[24] B. Krishanamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *Proc. Int. Workshop Distrib. Event Based Syst.*, 2002, pp. 575–578.

[25] J. Yuan, Z. Li, W. Yu, and B. Li, "A cross-layer optimization framework for multicast in multi-hop wireless networks," in *Proc. 1st IEEE Int. Conf. Wicon*, 2005, pp. 47–54.

[26] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," in *Proc. 9th Annu. ACM Int. Conf. MobiCom*, 2003, pp. 66–80.

[27] M. Kodialam and T. Nandagopal, "Characterizing achievable rates in multi-hop wireless networks: The joint routing and scheduling problem," in *Proc. 9th Annu. ACM Int. Conf. MobiCom*, 2003, pp. 42–54.

[28] D. Lun, N. Ratnakar, R. Koetter, M. Médard, E. Ahmed, and H. Lee, "Achieving minimum-cost multicast: A decentralized approach based on network coding," in *Proc. IEEE INFOCOM*, 2005, pp. 1607–1617.



Zongpeng Li received the B.E. degree in computer science and technology from Tsinghua University, Beijing, China, in 1999 and the M.S. degree in computer science and the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2001 and 2005, respectively.

He is currently an Assistant Professor at the Department of Computer Science, University of Calgary, Calgary, AB, Canada. His research interests are in data networks and distributed algorithms.

Baochun Li (M'00–SM'05) received the B.Eng. degree in computer science and technology from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 1995 and the M.S. and Ph.D. degrees in computer science from the Department of Computer Science, University of Illinois at Urbana-Champaign, in 1997 and 2000, respectively.

Since 2000, he has been with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, where he is currently an Associate Professor. He held the Nortel Networks Junior Chair in Network Architecture and Services since October 2003 and the Bell University Laboratories Endowed Chair in computer engineering since August 2005. His research interests include application-level quality-of-service provisioning and wireless and overlay networks.

Prof. Li is a member of the Association for Computing Machinery (ACM). He was the recipient of the IEEE Communications Society Leonard G. Abraham Award in the Field of Communications Systems, in 2000.