On Increasing End-to-End Throughput in Wireless Ad Hoc Networks

Zongpeng Li, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto
$\{$*arcane, bli*$\}$*@eecg.toronto.edu*

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. 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 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.

1 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. Though 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 (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 needs 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, 3, 4, 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 multi-hop 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. Since 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, 7, 8, 9] for assistance. A carefully determined optimal strategy achieves better end-to-end throughput than an arbitrary strategy.

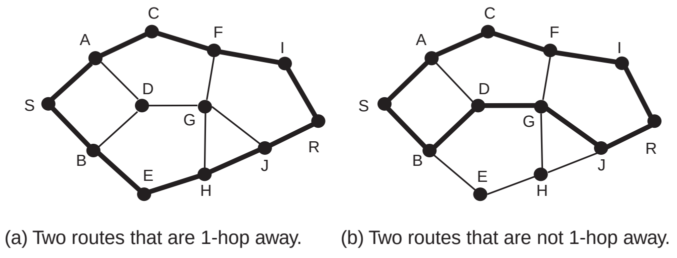
The remainder of the paper is organized as follows. Preliminaries are presented in Sec. 2. Sec. 3 discusses the case of data dissemination, including both unicast and multicast cases. Sec. 4 presents the case of data aggregation. Sec. 5 and 6 discuss related work and conclude the paper.

2 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 omni-directional antenna, where both the transmission range and the interference range is 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 multi-hop fashion. Local packet delivery is achieved by broadcasting at the MAC layer. Assuming each multi-hop 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 multi-hop 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 multi-hop flows that traverse more than two hops, thus consisting of more than two subflows, since these multi-hop flows exhibit spatial contention even among its own subflows.

We assume ideal MAC layer scheduling during the analysis of achievable throughput. We also assume that each source-destination connection is equally important, and should enjoy the same throughput.

A *flow* is the transmission of the same data along a route, which can be divided into multiple single-hop subflows. Two nodes are *1-hop away* if they are not within transmission range of each other. Two routes are *1-hop away* if beside the end nodes, each node on one route is 1-hop away from any node on the other route. We allow the end nodes to be identical. The concept is illustrated with an example in Fig. 1.



A
C
F
I
S
D
G
R
B
E
H
J
(a) Two routes that are 1-hop away.
A
C
F
I
S
D
G
R
B
E
H
J
(b) Two routes that are not 1-hop away.

Figure 1. The concept of "1-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 $t_0 = 1$ second.

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*, 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 *maximum contention clique (mcc)*, its size denoted as $|mcc|$.

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: We only give a sketched proof here due to space limitations. Since $r = C \sum_{i \in S} t_i / T$, we need to show that $T \geq |mcc|$ always holds, and $T = |mcc|$ if the underlying topology is a forest, *i.e.*, there are no cycles in it. It is immediate that $T \geq |mcc|$. Furthermore, when the underlying topology is a forest, we can extend a schedule of time $|mcc|$ on links in the *mcc* to all links without using a longer scheduling period. In fact, extending the schedule to all links on the same tree as the *mcc* is sufficient. This can be achieved by scheduling links not in the *mcc* one at a time, in a breadth-first order, *i.e.*, first consider links that are neighbors of the *mcc*, then links that are at 1-hop distance from the *mcc*, etc. At each step, upon scheduling link i, links that are already scheduled and interfere with i form a contention clique together with i — the correctness of this claim crucially depends on the property that the underlying network has no cycles and that links are considered in breadth-first order. Since the size of this contention clique is no larger than $|mcc|$, we will be able to fit i into the schedule, which is of length $|mcc|$, without introducing contention with links already scheduled. Therefore, the extension can proceed smoothly, and eventually we obtain a schedule of all links without interference using time $|mcc|$. We conclude that $T = |mcc|$ in this case. □

3 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 is transmitted from a single source to a single destination; in a multicast session, identical data is 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) 1-hop away multi-path; and (2) network coding.

3.1 Unicast

Consider a single route that serves a multi-hop 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, since 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 $C/3$, since the *mcc* has size 3, as shown in Fig. 2. The underlying intuition is that, due to intra-route spatial contention, only one out of every three links can be transmitting at a given 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 multi-path routing can be employed to break through the $C/3$ bound, by taking advantage of the *wireless advantage* (the broadcasting nature) at the source.

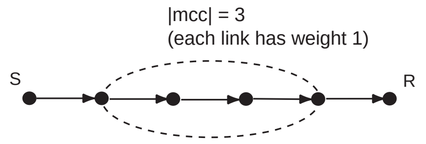


Figure 2. Achievable throughput of a single route is $C/3$.

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, 13, 14] on wireless multi-path 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 in order to reduce inter-route interference, and therefore achieve a higher session throughput, the transmission routes need to be 1-hop away.

Fig. 3 shows examples where two 1-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 3 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 4 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 $\frac{2}{3}C \cdot \frac{1}{3} + \frac{1}{2}C \cdot \frac{2}{3} = 5C/9$, which is a 66.7% improvement over the achievable throughput in the single route case.

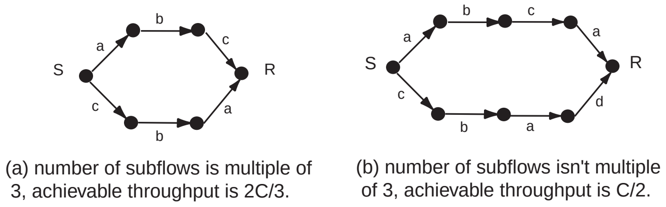

S
a
b
c
R
c
b
a
(a) number of subflows is multiple of 3, achievable throughput is 2C/3.
S
a
b
c
a
R
c
b
a
d
(b) number of subflows isn't multiple of 3, achievable throughput is C/2.

Figure 3. Two 1-hop away routes can achieve $2C/3$ or $C/2$.

If we further increase the number of 1-hop away routes, the achievable throughput can be further increased, with decreasing amounts of improvement. When three routes are used, it takes 4 or 5 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 1-hop away neighbors, as shown in Fig. 4. Therefore the throughput of a single unicast session is bounded by $5C/6$.

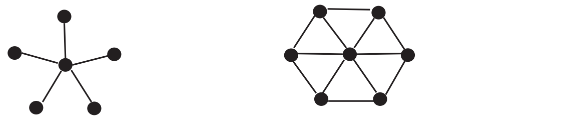

(a) Five 1-hop away neighbors.
(b) Six 1-hop away neighbors: impossible.

Figure 4. Upper bound on the number of 1-hop away neighbors.

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

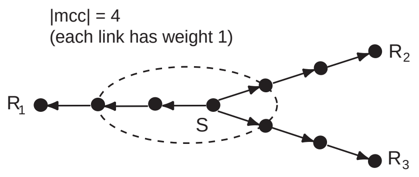


Figure 5. 1-to-k unicast can achieve a throughput of $kC/(k+1)$ at the sender, for $k \leq 5$.

Essentially, the above approach *interleaves* unicast sessions that would otherwise need to be transmitted sequentially, without using the multi-path 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.

3.2 Multicast

As previously noted, we focus on source-end throughput. In a multicast session, this translates to the throughput at the single source, though 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 session is also bounded by $C/3$. The strategy of using 1-hop away multi-paths 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 1-hop away routes to reach the respective destinations, similar to the previously discussed cases with multiple unicast destinations.

Further, we discuss the effects of branching points in the multicast tree on throughput. In a multicast session, identical data is 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 1-hop away branches will not increase the throughput of a multicast session compared to the strategy of branching late, since 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. As shown in the example in Fig. 6(b), if we branch immediately at the source, and then transmitting (identical) data to the two destinations along two 1-hop away routes, a throughput of $C/3$ can be achieved. In comparison, branching at the last hop (shown in Fig. 6(a)) achieves $C/3$ as well, and consumes only approximately half of the bandwidth as that of early branching.

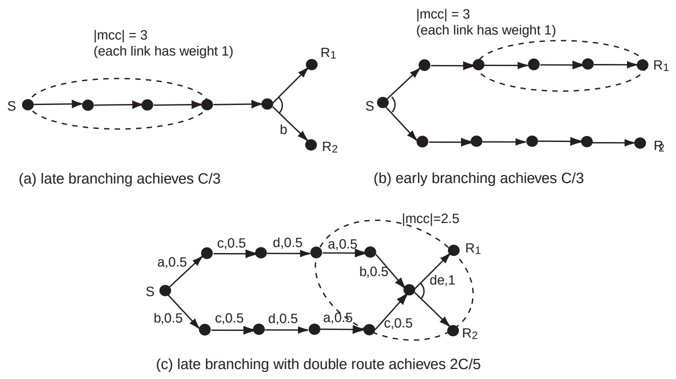


Figure 6. (a) Late branching; (b) early branching; and (c) late branching with two 1-hop away routes.

In the case of late branching, if we use multiple 1-hop away routes to "strengthen" the longer routes before late branching, the session throughput may be increased. In the example of Fig. 6(c), if the sender transmits (independent) data along two 1-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 second) data from the source to each destination within a scheduling period consisting of five phases, $a-e$, each of length 0.5 second. Therefore, $r \geq 2C/5$. Furthermore, the *mcc* has size 2.5, which implies that $r \leq 2C/5$. Therefore $r = 2C/5$.

Network coding

It is not always possible for a source to find 1-hop away routes, especially for multicast, since a multicast tree usually spans a broader range around the source. Furthermore, since the secondary routes may not be as short as the primary route, 1-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 multi-path 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 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. 7(a), an example taken from pervious work ([6]), shows how coding facilitates the increase of throughput in a 1-to-2 multicast session in wireline networks. 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, since coding yields no improvement on trees. Second, applying coding also involves non-identical data flows, and certain nodes must transmit different data flows along different outgoing links. Compared with the case where data is transmitted along a 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. 7(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.

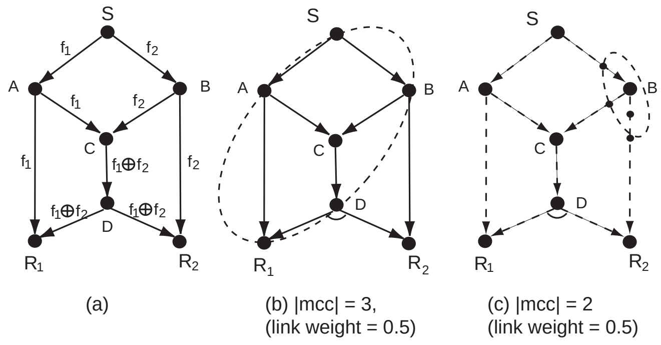


Figure 7. The effects of coding in (a) dense and (b) sparse wireless ad hoc networks.

Nevertheless, we observe that, the advantage of coding to increase throughput can outweigh the disadvantage introduced by spatial contention if (1) the 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 $|mcc|$ as small as 4, while the $|mcc|$ of a multi-hop multicast tree is 3. The difference is much less than the case of a small and dense transmission network. Fig. 7(c) shows such an example. The topology is similar to that in (b), and each link in (b) is replaced by a multi-hop 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 inter-tree spatial contention. In comparison, we have non-identical 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.

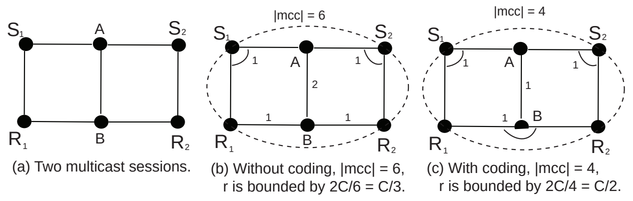

S1
A
S2
R1
B
R2
(a) Two multicast sessions.
|mcc| = 6
S1
S2
A
1
1
2
1
1
R1
B
R2
(b) Without coding, |mcc| = 6,
r is bounded by 2C/6 = C/3.
|mcc| = 4
S1
S2
A
1
1
1
1
B
R1
R2
(c) With coding, |mcc| = 4,
r is bounded by 2C/4 = C/2.

Figure 8. Two multicast sessions and upper bounds of their total throughput with and without coding.

Below we illustrate the above observations with a concrete example, as shown in Fig. 8 and Fig. 9. 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.

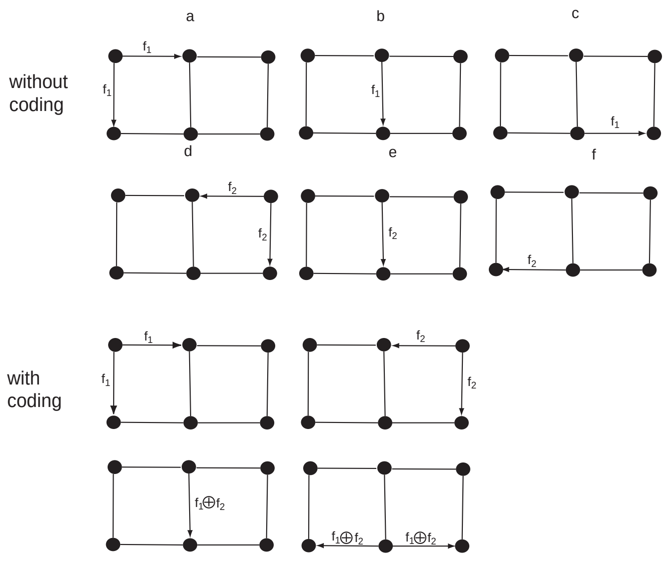


Figure 9. Without coding, achievable total throughput is $C/3$; with coding, achievable total throughput is $C/2$.

We have shown that, though 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. Since 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 1-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.

4 Data aggregation

Data aggregation refers to the form of data transmission in which data from multiple sources is transmitted towards 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 towards one common "data sink," possibly a gateway or data processing node. The data flows that are transmitted towards 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 call 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*.

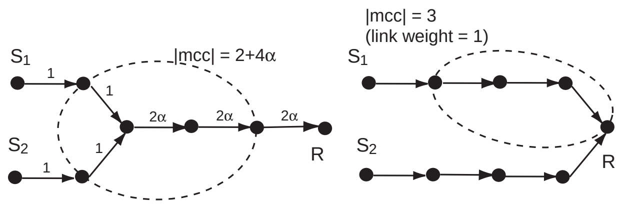
Due to the presence of data compression upon aggregation, the total amount of data that leaves the sources may not be equal to the amount of data that arrives at the sink. These two amounts are equal only in the zero aggregation case, otherwise the source-side amount is larger than the destination-side 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 Estrin *et al.* ([16]), the trade-offs between early aggregation and late aggregation include:

- 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, since the loss of non-aggregated packets is less severe than the loss of aggregated packets;
- early aggregation may introduce a higher latency.

We examine another dimension of the trade-off, 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 trade-off 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. In order 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 1-hop away paths, and (2) aggregation nodes are 1-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.



(a) Early aggregation, r = C/(1+2α). (b) Late aggregation, r = 2C/3.

Figure 10. Early aggregation versus late aggregation in a 2-to-1 data aggregation session.

Fig. 10 shows an aggregation session with two sources. With early aggregation, the size of *mcc* is $2 + 4\alpha$, and r = $2C/(2 + 4\alpha) = C/(1 + 2\alpha)$. Since $\alpha \in [0.5, 1]$, $r \in [C/3, C/2]$. 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 increase, the achievable throughput of late aggregation soon increases to $5C/6$ where it stops, while the achievable throughput of early aggregation keeps increasing, and depending on α, it may soon become higher than C.

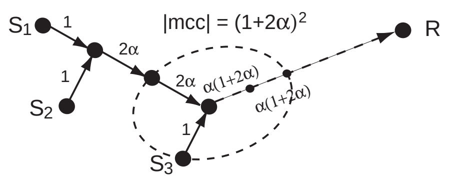


Figure 11. Early aggregation on a 3-to-1 aggregation session: $|mcc|$ **=** $(1 + 2\alpha)^2$**,** $r = 3C/(1 + 2\alpha)^2$**.**

Table 1 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 1-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. 11 shows the $n = 3$ case.

Table 1. Early *v.s.* 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]

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. 12, 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.

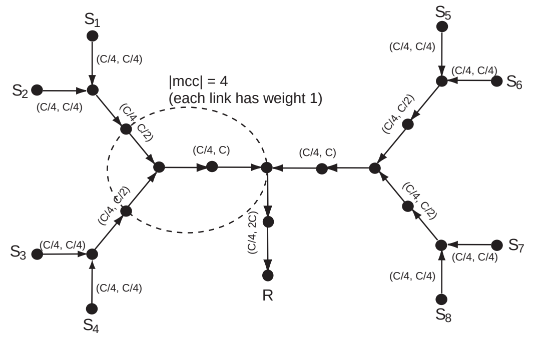


Figure 12. A perfect 8-to-1 early aggregation session, $r = 2C$**.**

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 $k = 1$ or $k = n$ is the optimal choice.

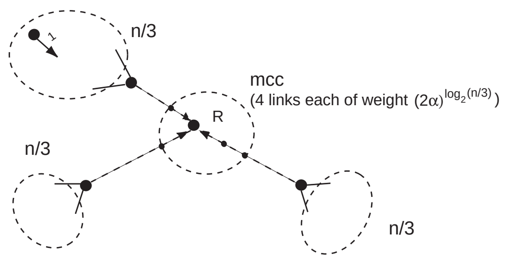


Figure 13. The case of three final entering flows.

We use r_k to denote the achievable session throughput of a data aggregation session with k final flows. Fig. 13 shows the cases where $k = 3$. 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:

$$r_1 = \frac{nC}{2(2\alpha)^{\log_2 n} + 2(2\alpha)^{\log_2 n-1}} \in [\frac{C}{3}, \frac{nC}{4}]$$

$$r_2 = \frac{nC}{2(2\alpha)^{\log_2 \frac{n}{2}} + 2(2\alpha)^{\log_2 \frac{n}{2}-1}} \in [\frac{2}{3}C, \frac{n}{4}C]$$

$$r_k\,(k = 3, 4, 5) = \frac{nC}{(k+1)(2\alpha)^{\log_2 \frac{n}{k}}} \in [\frac{k}{k+1}C, \frac{n}{k+1}C]$$

$$r_k\,(k \geq 6) = \frac{nC}{6(2\alpha)^{\log_2 \frac{n}{k}}}\frac{5}{k} \in [\frac{5}{6}C, \frac{5n}{6k}C]$$

Fig. 14 plots the throughput computed as above when $n = 20$, against the value of α. 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.

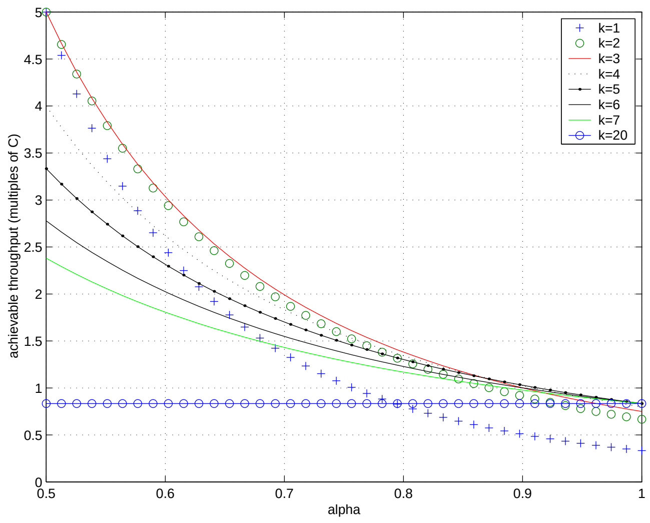


Figure 14. Achievable throughput vs. number of final flows.

5 Related work

In the CSMA/CA category of MAC protocols, data transmission is preceded by handshaking of control packets (RTS/CTS) [17]. 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 1-hop range of the sender or receiver of the other transmission [11, 10].

Classical studies of multi-path routing in wireline networks has been focusing on the objectives of load balancing and fault tolerance [18, 19, 20]. Increasing end-to-end throughput is neither a design goal nor a major advantage of wireline multi-path routing. Research in wireless multi-path 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 multi-path routing explicitly towards the goal to counteract the unique intra-route 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, 21]. The focus is to reduce energy consumption due to data transmission. It is shown that constructing the most energy-efficient aggregation tree is NP-hard. Several heuristic solutions has been proposed.

The capacity of ad hoc networks has been studied in previous work [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 [22].

6 Conclusions

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. Though 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. of IEEE INFOCOM*, 2000.

[2] P. Gupta and P. R. Kumar, "The Capacity of Wireless Networks," *IEEE Transactions on Information Theory*, vol. IT-46, no. 2, pp. 388–404, March 2000.

[3] J. Li, C. Blake, D. Couto, H. Lee, and R. Morris, "Capacity of Ad Hoc wireless networks," in *Proc. of ACM Mobicom 2001*, September 2001, pp. 61–69.

[4] P. Gupta, R. Gray, and P. R. Kumar, "An Experimental Scaling Law for Ad Hoc Networks," in *Technical Report*, 2001.

[5] P. Gupta and P. R. Kumar, "Internets in the Sky: The Capacity of Three Dimensional Wireless Networks," *Communications in Information and Systems*, vol. 1, no. 1, pp. 33–49, January 2001.

[6] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[7] R. Koetter and M. Medard, "Beyond Routing: An Algebraic Approach to Network Coding," in *Proc. of IEEE INFOCOM*, 2002.

[8] S. Y. R. Li and R. W. Yeung, "Network Multicast Flow Via Linear Coding," in *Proceedings of International Sym. Oper. Res. and its Appl.*, 1998.

[9] T. Ho and R. Koetter, "A Coding View of Network Recovery and Management for Single-receiver Communications," in *Conference on Information Sciences and Systems*, 2002.

[10] H. Luo, S. Lu, and V. Bharghavan, "A New Model for Packet Scheduling in Multihop Wireless Networks," in *Proc. of ACM MobiCom*, 2000, pp. 76–86.

[11] H. Luo, P. Medvedev, J. Cheng, and S. Lu, "A Self-Coordingating Approach to Distributed Fair Queueing in Ad Hoc Wireless Networks," in *Proc. of IEEE INFOCOM*, 2001.

[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 *IEEE/ACM Mobihoc*, 2000.

[13] K. Wu and J. Harms, "On-Demand Multipath Routing for Mobile Ad Hoc Networks," in *Proc. of the 3rd European Personal Mobile Communications Conference*, 2001.

[14] A. Nasipuri and S. R. Das, "On-Demand Multipath Routing for Mobile Ad Hoc Networks," in *Proc. of International Conference on Computer Communications and Networks (ICCCN)*, 1999.

[15] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks," *Mobile Computing and Communications Review*, 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. of International Conference on Distributed Computing Systems (ICDCS 2001)*, 2001.

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

[18] D. Sidhu, R. Nair, and S. Abdallah, "Finding Disjoint Paths in Networks," in *ACM SIGCOMM 91*, 1991.

[19] J. Chen and D. Subramanian, "An Efficient Multipath Forwarding Method," in *IEEE INFOCOM 98*, 1998.

[20] I. Cidon, R. Rom, and Y. Shavitt, "Analysis of Multi-path Routing," *ACM/IEEE Transactions on Networking*, vol. 7, no. 6, pp. 885–896, December 1999.

[21] B. Krishanamachari, D. Estrin, and S. Wicker, "The Impact of Data Aggregation in Wireless Sensor Networks," in *International Workshop of Distributed Event Based Systems*, 2002.

[22] J. Yuan, Z. Li, W. Yu, and B. Li, "A Cross-Layer Optimization Approach for Multicast in Multi-hop Wireless Networks," in *The First IEEE International Conference on Wireless Internet (Wicon 2005)*, 2005.