Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 08-033

Design of forwarder list selection scheme in opportunistic routing protocol

Yanhua Li, Wei Chen, and Zhi-li Zhang

October 22, 2008

Design of forwarder list selection scheme in opportunistic routing protocol

Yanhua Li, Wei Chen, Zhi-Li Zhang

***Abstract*—Unlike traditional wireless routing protocols which use a single fixed path, *opportunistic routing* explicitly takes advantage of the broadcast nature of wireless communications by using a set of forwarders to *opportunistically* perform packet forwarding. A key issue in the design of opportunistic routing protocols is the *forwarder list selection* problem. In this paper we establish a general theory for analyzing the forwarder list selection problem, and develop an optimal solution, the *minimum transmission selection (MTS)* algorithm, which minimizes the expected number of transmissions. Through extensive simulations using the MIT Roofnet dataset, we demonstrate that in more than 90% cases the MTS algorithm outperforms the forwarder selection scheme used in ExOR, the best known opportunistic routing protocol in the literature.**

I. Introduction

The *opportunistic routing* paradigm [2], [3] has opened a new avenue for designing routing protocols in multi-hop wireless networks. Unlike traditional routing wireless protocols such as DSR, AODV [6], [7] which rely on a (pre-selected) single *fixed* path for delivering packets from a source to a destination, opportunistic routing explicitly takes advantage of the broadcast nature of wireless communications to allow multiple (pre-selected) forwarders to *opportunistically* deliver packets from a source to a destination. The path a packet takes depends on which forwarders happen to receive it, and is thus non-deterministic. As a result, opportunistic routing can better cope with the *lossy, unreliable* and *varying* link qualities that are typical of wireless networks.

While the basic idea of opportunistic routing is fairly straightforward, there are several key issues that must be addressed in designing an opportunistic routing protocol. For example, given a source and a destination in a multi-hop wireless network, which nodes should be selected as (intermediate) forwarders for opportunistic packet forwarding? When a packet is transmitted (or rather *broadcasted*), multiple forwarders may receive it. Which one of them should forward it so as to avoid unnecessary *duplicate* transmissions? These questions are part of the *forwarder list selection problem* in opportunistic routing. In a nutshell, in order to fully realize the benefits of opportunistic routing, a list of forwarders must be judiciously selected, and these forwarders must *co-ordinate* their packet forwarding–either explicitly or implicitly–in order to successfully and efficiently deliver packets from the source to the destination.

Yanhua Li is with Beijing U. of Posts & Telecommunications, Beijing, China, Wei Chen is with U. of Electronic Sci. & Tech. of China, Chengdu, China, and Zhi-li Zhang is with Dep. of CS, U. of Minnesota, Minneapolis, USA. (Email: yanhua2008@gmail.com, chenwei@uestc.edu.cn, and zhzhang@cs.umn.edu.)

Take ExOR [2], the seminal and primary opportunistic routing protocol in the literature, as an example. Based on a link metric, *ETX*[1], ExOR ranks the potential forwarders using their *minimum path ETX*, namely, the sum of the constituent link ETX's along the "best path" (in terms of path ETX) from a node to the destination. The nodes that have smaller minimum path ETX's than the source are selected as forwarders, and they are ordered into a *global prioritized forwarder list*. Using such a forwarder list, ExOR employs carefully designed and elaborate packet forwarding mechanisms to realize the benefits of opportunistic routing. We see that while ExOR is an opportunistic routing protocol, it in fact utilizes a metric defined on a *single fixed path* for forwarder list selection and ordering, thereby ignoring the very effect of opportunistic routing. As will be shown later in this paper, such a forwarder list selection method is in general *sub-optimal*!

This paper is devoted to addressing the fundamental problem of *forwarder list selection* in opportunistic routing. We establish a general event-based analysis methodology and theory for studying the forwarder list selection problem, and develop an *optimal* solution which minimizes the expected number of transmissions under an ideal setting– the perfect ACK assumption (see Section II). To illustrate our analysis methodology, we first consider two simpler scenarios, 3-node and 4-node topologies in section III. In section IV we then develop the general theory and present an optimal solution, referred to as the *minimum transmission selection (MTS)* algorithm, for solving the forwarder list selection problem. Given a general topology, the MTS algorithm selects and computes the *optimal forwarder list* for *any* node to a given destination that minimizes the total expected number of transmissions, using a dynamic programming formulation (an iterative procedure) that is analogous to the one used in the Dijkstra's shortest path algorithm. In section V we discuss how to relax the perfect ACK assumption by proposing two heuristics to address the unreliable, asymmetric link qualities, thereby dealing with the effect of imperfect ACKs. In section VI we evaluate and compare the performances of the MTS algorithm and the forwarder list selection scheme of ExOR using extensive simulations. The paper is concluded in section VII.

II. Background and Problem Setting

As in ExOR [2], we assume that a *global prioritized forwarder list* is used for opportunistic packet forwarding, and the

[1]The ETX of a link (say, from node i to node j) is the inverse of the packet delivery probability p_{ij}, from node i to node j (measured and estimated based on some appropriate time interval).

same (batch mode-based, round-by-round, prioritized) *packet forwarding mechanisms* are employed, which is described below. Let $\{s, u_m, u_{m-1}, \ldots, u_1, d\}$ be an (ordered) forwarder list, where s is the source, d is the destination, and u_1,...,u_m are a set of m (intermediate) forwarders. The nodes are ordered based on their increasing priorities from left to right: namely, the destination node d has the highest priority, s the lowest priority, and for $1 \leq i < j \leq m$, u_i has higher priority over u_j. (If $m = 0$, no intermediate forwarder is used.) The priority of the node in the forwarder list is important in coordinating the packet forwarding in the (prioritized) packet forwarding mechanisms used in ExOR. In [2], these mechanisms are described using the *batch mode*, where a batch of packets, say, 100 packets, are transmitted in a round-by-round fashion using tightly controlled packet scheduling. For clarity, we will illustrate the ExOR packet forwarding mechanisms using a single packet (i.e., a batch of one packet) below.

Suppose the source s has a single packet to deliver to the destination d, using the forwarder list $\{s, u_m, u_{m-1}, \ldots, u_1, d\}$. The source s first transmits (or rather, broadcasts) the packet. One or multiple nodes in the forwarder list may receive the packet. If the destination d (the highest priority node) receives, it immediately broadcasts back[2] an acknowledgement (ACK). All lower priority nodes "overhearing" the ACK would therefore cancel their transmission and drop the packet, thereby completing the packet delivery from the source to the destination. Now consider the scenario where d does not receive the packet, and suppose u_i ($1 \leq i \leq m$) is the highest priority node that receives it. The node u_i would first defer its transmission by waiting for i time slots (each time slot equals to the transmission time of a single packet), while listening to any transmission by higher priority nodes during these time slots. (Note that such deferment thereby allows for the higher priority nodes before u_i to have a better chance to transmit first, just in case one of them has received it also.) Since no transmission is heard during the deferment, node u_i would transmit the packet immediately afterwards. Any lower priority node that "overhears" this transmission from u_i would then cancel its transmission and drop the packet. Whereas, if some higher priority nodes (but not the destination) receive the transmission of u_i, say, $u_j, j < i$ is the highest priority node that receives the packet, then the packet has made *forward progress* from a lower priority node u_i to a higher priority node u_j towards the destination d. The process (a new "round"[3] of transmission) would start anew at u_j. If none of the higher priority nodes "hear" its transmission from u_i, i.e., receiving the packet, u_i would start *re-transmitting* the packet again after waiting for an appropriate amount of time. Whenever the destination d receives the packet, it will broadcast an ACK, thus completing the packet transmission process.

From the description above, we see that all of the nodes in the forwarder lists must cooperate through appropriate transmission deferment and constantly listening to (or "overhearing") each other's transmissions during such deferment, and based on these transmissions (or lack thereof), coordinate (start or abort) their own packet transmissions. Due to unreliable wireless communications, in general multiple rounds of transmissions (when making forward progress from a lower priority node to a higher priority node) and re-transmissions (when no forward progress is made) are needed in order to opportunistically forward a packet from the source to the destination. Hence *the (expected) number of transmissions (including retransmissions) required when using a given forwarder list for opportunistic packet forwarding is an important metric to measure the "goodness" of a forwarding list.* We note also that unreliable wireless communications have another effect on opportunistic packet forwarding: when a lower-priority forwarder *does not overhear* the transmission of a higher-priority node (a form of *implicit* ACK to the lower-priority forwarder to abort its transmission of the same packet), it may lead to unnecessary *duplicate* transmissions, thus affecting the total number of packet transmissions. However, we argue that this *imperfect ACK* problem is not crucial to, and in a sense, orthogonal to, the forwarder list selection problem, as appropriate mechanisms can be designed to alleviate this problem which can be applied equally to any forwarder list selection scheme. For instance, the batch mode used in ExOR (and its enhancement we introduce in section V) is designed specifically for eliminating or reducing unnecessary duplicate transmissions due to this imperfect ACK problem.

In analyzing the forwarder list selection problem in this paper, we therefore employ the *expected number of transmissions* (excluding duplicate transmissions due to unreliable overhearing) as the key metric to compare forwarder lists, and address the two basic questions: *i) which forwarders to select*, and *ii) how to order the forwarders selected*? To simplify our analysis and separate the effect of duplicate transmissions due to unreliable overhearing, we introduce and consider an *ideal setting* with the *perfect ACK* assumption, namely, we assume that lower priority nodes can always overhear the transmissions of higher priority nodes, thus there will be no unnecessary duplicate transmissions. Under this assumption, in this paper we develop a general methodology and theory for analyzing the forwarder list selection problem, and develop a dynamic programming algorithm for finding the *optimal* forwarder list which minimizes the expected number of transmissions for a packet to traverse opportunistically from a source to a destination. We will relax this perfect ACK assumption in section V by introducing two mechanisms to handle unreliable and asymmetric packet delivery probabilities.

We conclude this section by showing that the forwarder list selection scheme used in ExOR is *sub-optimal* through

[2]For simplicity, here we assume that all nodes have a nonzero probability of "hearing" or "over-hearing" each other's transmissions. In fact, the design of ExOR implicitly posits a dense wireless mesh network where all nodes in the network are more or less connected, albeit with unreliable links, sometimes with extremely low packet delivery probability.

[3]Here we use the term "round" to indicate forward progress made by the packet. With $m+2$ nodes in the forwarder list, a worst-case of $m+1$ rounds may be needed. Multiple (re-)transmissions may be needed in each round to make forward progress.

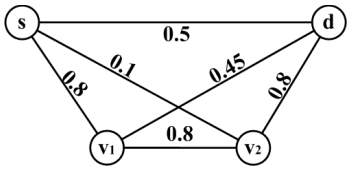


Fig. 1. An example showing that ExOR's forwarder list selection scheme is sub-optimal.

a simple example. Recall that ExOR uses the following two rules to select and order forwarders: *a) candidate set selection rule*: node with a smaller path ETX value (to the destination) than the source will be chosen in the forwarder list; and *b) ordering rule*: forwarders are ordered by their path ETX values to the destination, where a forwarder with smaller path ETX value has higher priority. To show that the forwarder list selected based on these two rules are *sub-optimal*, we consider a simple example shown in Fig.1, where node s is the source, d is the destination, and the number on each link indicates its (symmetric) packet delivery probability. Computing the path ETX using the best path, we see that the path ETX for node v_2 (to d) is $1/0.8 = 1.25$, for node v_1 it is $1/0.45 = 2.22$, and for the source node s it is $1/0.5 = 2$. Hence the forwarder list selected by ExOR is $\{s, v_2, d\}$. However, it is not hard to argue that the forwarder list $\{s, v_1, v_2, d\}$ is in fact a better choice, as it is likely to reduce the total number of transmissions to deliver packets from s to d. This is because that v_1 has two opportunistic paths, $v_1 \rightarrow d$ and $v_1 \rightarrow v_2 \rightarrow d$, which together yield a higher packet probability than using either path alone. The expected number of transmissions from v_1 to d using both paths is roughly 1.742, which is far smaller than the path ETX of v_1 (2.22) and is also smaller than the path ETX of s (2). The *sub-optimality* of the ETX forwarder list selection scheme lies in that it uses the *single path* ETX metric for selecting forwarders, and thus fails to account for the *opportunistic* nature of packet forwarding.

III. Basic Theory: Three- and Four-Node Cases

In this section we consider the problem of forwarder list selection under two simplest scenarios, namely, networks with 3 and 4 nodes respectively. Under the perfect ACK assumption, we establish several theorems to address the two fundamental questions regarding the forwarder list selection: i) when should an (intermediate) node be selected in a forwarder list, and ii) if there are multiple forwarders, how should they be ordered? As will be shown in the next section, these theorems provide the key insights into, and lay the foundation for establishing a general theory for solving the more general forwarder list selection problem.

A. Three-Node Case

We first consider the 3-node case, the simplest possible scenario, as shown in Fig. 2(a), where s and d are the source and destination nodes, and v is the intermediate node. The link qualities are labeled in the figure. We assume all probabilities are non-zero (otherwise the question become trivial), and we are interested in the following question: suppose s has one packet to transmit to d, when shall we include v as a forwarder so as to *minimize the total number of expected transmission* (under the perfect ACK assumption)?

To answer the above question, we need to compare the expected number of transmissions under two possible options: a) using the forwarder list $\{s, d\}$ (i.e., do not include v), and b) using the forwarder list $\{s, v, d\}$ (i.e., include v). (Note that the rightmost node has the highest priority.) Let $N_{s,d}$ denote the expected total number of transmissions under option a). Clearly, $N_{s,d} = 1/z$. To derive the expected total number of transmissions under option b), $N_{s,v,d}$, we proceed as follows. Let $\mathcal{E} := v \vee d$ denote the event that when s transmits the packet, either d or v receives it. The probability of this event is $Pr(\mathcal{E}) = 1-(1-x)(1-z)$. Under the forwarding mechanism described in Section II, the expected number of transmissions s must perform until the event $\mathcal{E}$ holds, denoted by $N_{s\rightarrow\mathcal{E}}$, is

$$N_{s\rightarrow\mathcal{E}} = \frac{1}{P_r(\mathcal{E})} = \frac{1}{1-(1-x)(1-z)}.$$

Given that the event $\mathcal{E}$ holds, we have two possibilities: i) d receives the packet, in this case no additional transmissions are needed; or ii) d does not receive the packet but v does (denote this event by $v \wedge \bar{d}$). In the latter case, v must repeatedly transmit the packet until d receives it. The number of expected transmissions v must perform is $N_{v\rightarrow d} = 1/y$. Hence the expected number of transmissions using the forwarder list $\{s, v, d\}$ is

$$\begin{aligned} N_{s,v,d} &= N_{s\rightarrow\mathcal{E}} + N_{v\rightarrow d} \cdot P_r(v \wedge \bar{d}|\mathcal{E}) \\ &= \frac{1 + x(1-z)\frac{1}{y}}{1-(1-x)(1-z)}. \end{aligned} \quad (1)$$

With some simple algebraic manipulations, we can show that $N_{s,v,d} < N_{s,d}$ *if and only if* $y > z$. This yields the following theorem.

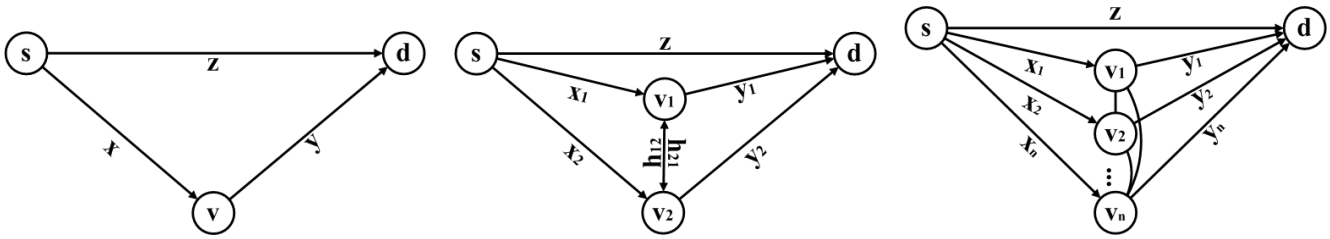
THEOREM 1 (Three Nodes). *In a three-node network as shown in Fig. 2(a), the node v should be included in the forwarder list,* if and only if $y > z$ *so as to reduce the expected number of transmissions. In addition, if $y > z$ holds, the expected number of transmissions using the forwarder list $\{s, v, d\}$ is*

$$N_{s,v,d} = \frac{1 + x(1-z)\frac{1}{y}}{1-(1-x)(1-z)} \quad (2)$$

where we have $\frac{1}{y} < N_{s,v,d} < \frac{1}{z}$.

TABLE I
Forwarder lists for topology 2(b)

Options	Forwarder lists	Condition to be best (where $y_1 \geq y_2$)
$List1$	$\{s, d\}$	$z > y_1 \geq y_2$
$List2$	$\{s, v_1, d\}$	$y_1 > z, N_{v_2,v_1,d} > N_{s,v_1,d}$
$List3$	$\{s, v_2, d\}$	N/A
$List4$	$\{s, v_2, v_1, d\}$	$y_1 > z, N_{v_2,v_1,d} < N_{s,v_1,d}$
$List5$	$\{s, v_1, v_2, d\}$	N/A



(a) Three Nodes Topology.x, y, and z are link qualities, which have value between 0 and 1.

(b) Four Nodes Topology.x_1, y_1,x_2, y_2, and z are link qualities, which have value between 0 and 1.

(c) N intermediate nodes topology.

Fig. 2. Example topologies to show the forwarder list selection theory.

B. Four-Node Case

We now consider a slightly more complex scenario with four nodes, as shown in Fig.2(b), where we have two intermediate nodes v_1 and v_2 between the source s and the destination d. The packet delivery probabilities are as labeled in the figure, where the delivery probability from v_1 to v_2 is h_{12}, and that from v_2 to v_1 is h_{21}. Given this topology, we have a total of 5 possible forwarder lists as listed in Table I. In the following, we will analyze which of these forwarders lists is the best and under what condition *under the assumption of* perfect ACK. Without loss of generality, we assume $y_1 \geq y_2$. In this case from Theorem 1, the forwarder $\{s, v_1, d\}$ is always better than $\{s, v_2, d\}$, and from our analysis below, it will be clear that $\{s, v_1, v_2, d\}$ is always better than $\{s, v_2, v_1, d\}$. The condition under which each forwarder list is the best is listed under the last column in Table I, and is summarized in the following theorem.

THEOREM 2 (Four Nodes). *Given the four-node topology with two intermediate nodes v_1 and v_2 as shown in Fig.2(b), and assuming that $y_1 \geq y_2$, we have the following forwarder list selection criteria (under the perfect ACK assumption):*

- *If $z \geq y_1 (\geq y_2)$, use the forwarder list $\{s, d\}$ only. In other words, using either v_1 or v_2 or both will not help reduce the expected number of transmissions.*
- *If $y_1 > z$, then always add v_1 in the forwarder list, and i) use the forwarder list $\{s, v_1, d\}$ if $N_{s,v_1,d} \leq N_{v_2,v_1,d}$; otherwise, ii) use the forwarder list $\{s, v_2, v_1, d\}$. In other words, v_2 should be added in the forwarder list (always after v_1), if and only if $N_{v_2,v_1,d} < N_{s,v_1,d}$.*

This theorem can be proved through a systematic event-based analysis below.

Given the four-node topology in Fig. 2(b), when s transmits a packet to d, we have one of the following events:

1) None of the nodes v_1, v_2 or d receives it: this event is denoted as $\bar{\mathcal{E}} := (\bar{v_1} \wedge \bar{v_2} \wedge \bar{d})$, and the probability of this event occurring is $Pr(\bar{\mathcal{E}}) = (1-x_1)(1-x_2)(1-z)$. Given this event, *no matter what forwarder list is used*, s needs to retransmit the packet until at least one of them receives it. The number of expected transmissions that s must perform until at least either v_1, v_2 or d receives (i.e., until the event $\mathcal{E} := v_1 \vee v_2 \vee d$ occurs) is

$$N_{s \to \mathcal{E}} = \frac{1}{P_r(\mathcal{E})} = \frac{1}{1-(1-x_1)(1-x_2)(1-z)}.$$

2) At least one of the nodes v_1, v_2 or d receives the packet: this event is denoted by $\mathcal{E} := (v_1 \vee v_2 \vee d)$, and its probability is $1-(1-x_1)(1-x_2)(1-z)$. Given this event, we have the following four possibilities:

a) if d receives the packet (and no matter whether v_1 or v_2 receives it or not), this (conditional) event is denoted as $(d|\mathcal{E})$. Given this event, no matter what forwarder list is used, no additional transmission is needed.

b) If d does not receive it but both v_1 and v_2 receive, this (conditional) event is denoted as $(v_1 \wedge v_2 \wedge \bar{d}|\mathcal{E})$.

c) If d and v_2 do not receive it but v_1 receives it, this (conditional) event is $(v_1 \wedge \bar{v_2} \wedge \bar{d}|\mathcal{E})$.

d) If d and v_1 does not receive it but v_2 does, the (conditional) event is $(\bar{v_1} \wedge v_2 \wedge \bar{d}|\mathcal{E})$.

Using the above event break-down, we first show that if $z > y_1 (\geq y_2)$, then the best forwarder list is $\{s, d\}$. From Theorem 1, it is clear that $\{s, d\}$ is better than $\{s, v_1, d\}$ and $\{s, v_2, d\}$. The question left is then: under this condition, would it be helpful to use both intermediate nodes v_1 and v_2 as forwarders? The difference between using the forwarder list $\{s, d\}$ and $\{s, v_2, v_1, d\}$ (or $\{s, v_1, v_2, d\}$ lies in the last three events, where the latter will use either v_1 or v_2 to perform transmissions instead of s. Assuming we are using the forwarder list $\{s, v_2, v_1, d\}$, let us see what the expected transmissions will be required by using either v_1 or v_2. Note that since we give v_1 higher priority than v_2, in the event of 2b) we will use v_1 to transmit instead of v_2 (this is the only difference between using the forwarder list $\{s, v_2, v_1, d\}$ as opposed to $\{s, v_1, v_2, d\}$). The conditional probability that either 2b) or2c) occurring given $\mathcal{E}$ is $Pr(v_1 \wedge \bar{d}|\mathcal{E}) = x_1(1-z)/[1-(1-x_1)(1-x_2)(1-z)]$. Under this event, the expected number of transmissions by v_1 until d receives is $1/y_1$, which is less than or equal to $1/z$, as $z \geq y_1$. This is straightforward as the probability of d eventually receiving the packet if

v_1 keeps (re-)transmitting is less than the probability of d eventually receiving the packet if s keeps (re-)transmitting.

Similarly, in the event of 2d), we will use v_2 to transmit the packet (since v_1 does not receive the packet). However, once v_2 transmits the packet, if d does not receive it but v_1 receives it, then v_1 will resume the transmission of the packet instead of v_2. In this case, the expected number of total transmissions by either v_2 or v_1 is given by the three-node scenario we studied earlier, with v_2 as the source, v_1 the intermediate node, and d the destination. Hence by Theorem 1, the expected number of total transmissions by v_1 or v_2 is

$$N_{v_2,v_1,d} = \frac{1 + h_{21}(1-y_2)1/y_1}{1-(1-h_{21})(1-y_2)} \geq \frac{1}{y_1} > \frac{1}{z}$$

where the last two inequalities hold because $z \geq y_1 \geq y_2$. This inequality implies that the probability of d eventually receiving the packet if we have v_1 and v_2 to perform (re-)transmissions is less than the probability of d eventually receiving the packet if s keeps (re-)transmitting the packet.

Using a similar argument, we can show that the forwarder list $\{s, v_1, v_2, d\}$ would in fact result in more transmissions than $\{s, d\}$. Combining all the arguments above, we therefore establish that if $z > y_1 (\geq y_2)$, the forwarder list $\{s, d\}$ is the best among all options.

Now we consider the scenario where $y_1 > z$ (and the relationship between z and y_2 is arbitrary). Again from Theorem 1, we know that the forwarder list $\{s, v_1, d\}$ is better than $\{s, d\}$. The question is now whether and when using v_2 will help reduce the number of transmissions or not. In other words, under what condition, is the forwarder list $\{s, v_2, v_1, d\}$ better than $\{s, v_1, d\}$? We claim that $\{s, v_2, v_1, d\}$ is better than $\{s, v_1, d\}$ *if and only if* $N_{v_2,v_1,d} < N_{s,v_1,d}$. Here $N_{v_2,v_1,d}$ is the expected number of transmissions in a 3-node (sub-)topology (of the four-topology in Fig. 2(b)) where v_2 is the source, d is the destination, v_1 is the intermediate node, and the packet delivery probabilities between the nodes are labeled as in Fig. 2(b). $N_{s,v_1,d}$ is similarly defined. As $y_1 \geq y_2$ and $y_1 > z$, from Theorem 1 we have $1/y_1 \leq N_{v_2,v_1,d} \leq 1/y_2$, an $1/y_1 < N_{s,v_1,d} \leq 1/z$.

To see why if $N_{v_2,v_1,d} < N_{s,v_1,d}$, then $\{s, v_2, v_1, d\}$ is better than $\{s, v_1, d\}$, consider the three events 2b), 2c) and 2d) defined above. Clearly in the event of either 2b) or 2c), both forwarder lists would use v_1 as the forwarder to keep transmitting the packet until d receives it. The only difference is in the case of event 2d). Using the forwarder list $\{s, v_2, v_1, d\}$, v_2 will transmit the packet, and it will keep transmitting until either d receives it or v_1 receives it (in the latter v_1 will take over the task of transmission to d). Hence given the event 2d), the expected number of total transmissions by v_1 and v_2 is $N_{v_1,v_2,d}$. Whereas using the forwarder list $\{s, v_1, d\}$, the source s needs to re-transmit the packet. If $N_{v_1,v_2,d} < N_{s,v_1,d}$, we see that under the event 2d), using the forwarder list $\{s, v_2, v_1, d\}$ would result in a smaller expected number of total transmissions. Given that under all other events, the two forwarder lists produce the same number of total transmissions, we hence conclude that the expected number of total transmissions using the forwarder list $\{s, v_2, v_1, d\}$ is less than that using the forwarder list $\{s, v_1, d\}$, namely, $N_{s,v_2,v_1,d} < N_{s,v_1,d}$, *if and only if* $N_{v_1,v_2,d} < N_{s,v_1,d}$.

Using the same event-based analysis, we can in fact show that given $y_1 > y_2$, the forwarder list $\{s, v_2, v_1, d\}$ is always better than $\{s, v_1, v_2, d\}$. In the event of 2b), using v_1 instead of v_2 requires only $1/y_1 < 1/y_2$ expected number of transmissions. In the event of 2c), using v_1 to directly transmit to d requires $1/y_1$ expected number of transmissions, which is fewer than that if we use v_2 as an intermediate forwarder. This is because from Theorem 1 we have $N_{v_1,v_2,d} > 1/y_1$ when $y_1 > y_2$. On the other hand, in the event of 2d), using v_1 as an intermediate forwarder instead of using v_2 to directly transmit to d would reduce the expected number of transmissions, as $N_{v_2,v_1,d} < 1/y_2$ when $y_1 < y_2$. Hence in all these three events, $\{s, v_1, v_2, d\}$ would produce fewer expected number of transmissions. In fact, given $y_1 < y_2$, we know that $\{s, v_1, d\}$ is always better than $\{s, v_1, v_2, d\}$. This is because given $y_1 < y_2$, from Theorem 1 v_2 does not help v_1 in reaching d. In contrast, given $y_1 > z$ but $y_1 > y_2$, whether y_2 is larger or smaller than z, v_2 may in fact help the source s to reach the set $\{v_1, d\}$, thus reducing the expected number of transmissions. This is the case if and only if $N_{v_2,v_1,d} < N_{s,v_1,d}$, and in this case the forwarder list $\{s, v_2, v_1, d\}$ would be better than $\{s, v_1, d\}$. Clearly, if $z = y_1$, $\{s, d\}$ and $\{s, v_1, d\}$ are equally good, and if $z = y_1 = y_2$, then $\{s, d\}$, $\{s, v_1, d\}$ and $\{s, v_2, v_1, d\}$ (and $\{s, v_1, v_2, d\}$) are equally good. Now, the theorem.2 is proved completely.

IV. Forwarder List Selection: General Theory

Based on the insights and results we obtained for the 3- and 4-node cases, we now extend them to a general topology with n intermediate forwarders, and present the MTS algorithm for finding the *optimal* forwarder list which provably minimizes the expected number of transmissions needed to transmit a packet from the source to the destination (under the perfect ACK assumption). We start by establishing a recurrent formula for computing the expected number of transmissions for a given forwarder list.

A. Expected Number of Transmissions for a Given Forwarder List

Consider a general topology as shown in Fig.2(c), where we have a source s and a destination d, and n intermediate nodes that are arbitrarily connected among themselves (and with s and d). As noted in Fig. 2(c), we use z to denote the packet probability from the source node s directly to the destination node d, x_i the packet probability from node s to the (intermediate) node v_i, y_i the packet probability from node v_i to node d, and h_{ij} the packet probability delivery probability from node v_i to node v_j, where $0 \leq z \leq 1$, $0 \leq x_i \leq 1$, $0 \leq y_i \leq 1$ and $0 \leq h_{ij} \leq 1$ (note that some of these packet delivery probabilities can be 0). Given a total of n intermediate nodes, we have a total of $\sum_{i=0}^{n} \frac{n!}{(n-i)!}$ possible forwarder lists (including $\{s, d\}$–the the default forwarder list *without*

using any intermediate node). Of these $\sum_{i=0}^{n} \frac{n!}{(n-i)!}$ forwarder lists, which one is the best and under what condition? Instead of directly answering this question, we start by establishing a *recurrent* formula for computing the expected number of transmissions for a *given* forwarder list.

For $m = 1, \ldots, n$, let $u_m, \ldots, u_1$ denote an arbitrary (ordered) list of m intermediates nodes taken from the set of n intermediate nodes $\{v_1, v_2, \ldots, v_n\}$. Then $\{s, u_m, \ldots, u_1, d\}$ is a possible forwarder list. We would like to derive a formula to compute the expected number of transmissions to reach destination d using this particular forwarder list. We denote this number by $N_{s,u_m,\ldots,u_1,d}$. More generally, for $k = 1, \ldots, m$, we use $N_{u_k,\ldots,u_1,d}$ to denote the expected number of transmissions to reach destination d *when the packet has reached node* u_k*, but not* u_j*,* $j = 1, \ldots, k-1$. In other words, $N_{u_k,\ldots,u_1,d}$ is the expected number of transmissions to reach destination d *if* u_k is the source node and $\{u_k, \ldots, u_1, d\}$ is the forwarder list. Given these notations, we have the following recurrent formula for computing $N_{s,u_m,\ldots,u_1,d}$.

THEOREM 3 (Recurrent Formula for Expected Number of Transmissions). *Consider a general topology with n intermediate nodes as depicted in Fig.2(c). For $1 \leq m \leq n$, let $\{s, u_m, \ldots, u_1, d\}$ be an arbitrary forwarder list with m (ordered) intermediate forwarders, $u_i \in \{v_1, \ldots, v_n\}$. Then using this forwarder list, the expected number of transmissions for transmitting a packet currently at the source node s to reach the destination node d is given by the following recurrent formula (under the perfect ACK assumption):*

$$N_{s,u_m,\ldots,u_1,d} = \frac{1 + \sum_{k=1}^{m} x_k \prod_{j=1}^{k-1}(1-x_j)(1-z) N_{u_k,\ldots,u_1,d}}{1 - \prod_{k=1}^{m}(1-x_k)(1-z)} \tag{3}$$

Proof: The theorem follows easily by using the following event-based analysis. We use $\mathcal{E}^m := u_m \vee \cdots \vee u_1 \vee d$ to denote the event that when s transmits the packet, either the destination node d or (at least) one of the m intermediate forwarders receives it, whereas we use $\bar{\mathcal{E}}^m (:= \bar{u}_m \wedge \cdots \wedge \bar{u}_1 \wedge \bar{d})$ to denote the complement of this event, namely, when s transmits the packet, neither the destination node d nor any of the m intermediate forwarders receives it, For $1 \leq k \leq m$, let $\mathcal{E}^{k,\not<k}$ denote the event that when s transmits the packet, u_k receives it but none of the higher priority intermediate forwarder, u_j, $1 \leq j < k$, nor d, receives it, and $(\mathcal{E}^{k,\not<k}|\mathcal{E}^m)$ is this event conditioned on $\mathcal{E}^m$ occurring. Note that $(\mathcal{E}^{k,\not<k}|\mathcal{E}^m)$, $k = 1, \ldots, m$, plus the event $(d|\mathcal{E}^m)$ (i.e., the conditional event that the destination node d receives the packet), are mutually exclusive events, and provide a partition of the event $\mathcal{E}^m$. In other words, $\sum_{k=1}^{m} Pr(\mathcal{E}^{k,\not<k}|\mathcal{E}^m) + Pr(d|\mathcal{E}^m) = Pr(\mathcal{E}^m)$. Let $N_{s\to\mathcal{E}^m}$ denote the expected number of transmissions that the source node s has to perform until $\mathcal{E}^m$ occurs, i.e., at least one of the nodes in $\{u_m, \ldots, u_1, d\}$ receives the packet. Given these notations and recall the definition of $N_{u_k,\ldots,u_1,d}$, we have the following recurrent formula for $N_{s,u_m,\ldots,u_1,d}$:

$$N_{s,u_m,\ldots,u_1,d} = N_{s\to\mathcal{E}^m} + \sum_{k=1}^{m} Pr(\mathcal{E}^{k,\not<k}|\mathcal{E}^m) N_{u_k,\ldots,u_1,d}. \tag{4}$$

Note that if $(d|\mathcal{E}^m)$ occurs, then no additional transmissions are needed. The above recurrent formula states that the expected number of transmissions using the forwarder list $\{s, u_m, \ldots, u_1, d\}$ is the sum of the expected number of transmissions by the source node s until $\mathcal{E}^m$ occurs, and given the event, for $1 \leq k \leq m$, if u_k receives it but none of the higher priority forwarders *and nor* does the destination node d receive it, the expected number of transmissions, $N_{u_k,\ldots,u_1,d}$ from the intermediate forwarder u_k to reach d using the forwarder list $\{u_k, \ldots, u_1, d\}$. Clearly, $N_{s\to\mathcal{E}^m} = 1/Pr(\mathcal{E}^m)$ and $Pr(\mathcal{E}^m) = 1 - \prod_{k=1}^{m}(1-x_k)(1-z)$. For $k = 1, \ldots, m$, $Pr(\mathcal{E}^{k,\not<k}|\mathcal{E}^m) = x_k \prod_{j=1}^{k-1}(1-x_j)(1-z)/Pr(\mathcal{E}^m)$. Plugging these formulas into eq.(4) yields eq.(3). ■

B. Minimum Transmissions Selection Scheme

Firstly, let us introduce a theorem below, which is useful to prove the optimality of our selection scheme later.

THEOREM 4 (When to Extend a Forwarder List). *Given an existing forwarder list $\{s, u_{m-1}, \ldots, u_1, d\}$, adding a neighbor, u_m, of the source node s (i.e., $x_m > 0$) as an additional forwarder to the end of this forwarder list can reduce the expected total number of transmissions* if and only if $N_{u_m,u_{m-1},\ldots,u_1,d} < N_{s,u_{m-1},\ldots,u_1,d}$.

Proof: Using Theorem 3, we can compare two forwarder lists, say, $\{s, u_{m-1}, \ldots, u_1, d\}$ and $\{s, u_m, \ldots, u_1, d\}$, and thus decide whether add an additional forwarder u_m to (the end of) an existing forwarder list $\{s, u_{m-1}, \ldots, u_1, d\}$ would help or not. Using the notations defined in the proof of Theorem 3, we note that the difference between the two forwarder lists $\{s, u_{m-1}, \ldots, u_1, d\}$ and $\{s, u_m, \ldots, u_1, d\}$ lies in the fact that in the event of $(\mathcal{E}^{m,\not<m}|\mathcal{E})$, when using $\{s, u_m, \ldots, u_1, d\}$, u_m would take over the packet transmissions from the source node s, while using $\{s, u_{m-1}, \ldots, u_1, d\}$, s would continue to re-transmit the packet until either d receives it, or one of the nodes $\{u_1, \ldots, u_{m-1}\}$ receives it and takes over the packet transmissions. Using some simple algebraic manipulations, we can show the following holds:

$$\begin{aligned} N_{s,u_{m-1},\ldots,u_1,d} = N_{s\to\mathcal{E}} &+ Pr(\mathcal{E}^{m,\not<m}|\mathcal{E}) N_{s,u_{m-1},\ldots,u_1,d} \\ &+ \sum_{k=1}^{m-1} Pr(\mathcal{E}^{k,\not<k}|\mathcal{E}) N_{u_k,\ldots,u_1,d}. \end{aligned} \tag{5}$$

Comparing eq.(5) with eq.(4), it is easy to see that $N_{s,u_m,\ldots,u_1,d} < N_{s,u_{m-1},\ldots,u_1,d}$ *if and only if* $N_{u_m,u_{m-1},\ldots,u_1,d} < N_{s,u_{m-1},\ldots,u_1,d}$ and $x_m > 0$, i.e., U_m is a neighbor of the source node s (note that the latter is necessary otherwise $Pr(\mathcal{E}^{m,\not<m}|\mathcal{E}) = 0$). ■

We are now in a position to present the *Minimum Transmissions Selection(MTS)* algorithm, which computes the optimal

forwarder list that minimizes the expected number of transmissions. In fact, given a general topology and the destination d, the MTS algorithm computes the optimal forwarder lists from *all sources* to d, using a *dynamic programming* formulation somewhat analogous to the Dijkstra's algorithm (which computes the shortest paths *from a given source to all destinations*).

Algorithm 1 The MTS algorithm for computing the optimal forwarder lists of all sources v's to the destination d.

1: **//Initialization:**
2: $\mathcal{S} :=$ the set of all nodes except d
3: **for** each vertex v in $\mathcal{S}$ **do**
4: $FL^d[v] := \{v, d\}$
5: **if** $P_r(v \rightarrow d) > 0$ **then**
6: $N^d[v] := 1/P_r(v \rightarrow d)$
7: **else**
8: $N^d[v] := \infty$
9: **end if**
10: **end for**
11: **//Iterations:**
12: **while** $\mathcal{S}$ is not empty **do**
13: $u :=$ node in $\mathcal{S}$ with smallest $N^d[\cdot]$ (i.e., $u := argmin_{w \in \mathcal{S}} N^d[w]$)
14: remove u from $\mathcal{S}$
15: **for** each neighbor v of u and v is in $\mathcal{S}$ **do**
16: $FL^d[v] :=$ **merge**$(FL^d[u], FL^d[v])$ // see the text for definition of **merge**
17: $N^d[v] := N^d_{FL^d[v]}$ (where $N^d_{FL^d[v]}$ is computed using eq.(3))
18: **end for**
19: **end while**
20: **RETURN** $FL^d[v]$ for all nodes v

Given a general topology (e.g., Fig.2(c)), initially let $\mathcal{S}$ be the set of all nodes except for a given destination node d. The MTS algorithm for computing the optimal forwarder list from any source node $v \in \mathcal{S}$ to d is described in pseudo-code in Alg. 1. At each iteration of the algorithm, for any node v in $\mathcal{S}$, $FL^d[v]$ records the (best) forwarder list from v to d discovered so far, and $N^d[v]$ denotes the expected number of transmissions using the (currently best) forwarder list $FL^d[v]$. During the initialization stage (steps 1-9), for each $v \in \mathcal{S}$, clearly $FL^d[v] := \{v, d\}$ is the currently best forwarder list for v, and $N^d[v] = 1/Pr(v \rightarrow d)$ if the packet delivery probability $Pr(v \rightarrow d)$ is non-zero.

At each subsequent iteration while $\mathcal{S}$ is not empty (steps 12-19), we pick the node $u \in \mathcal{S}$ such that $u := argmin_{v \in \mathcal{S}} N^d[v]$ (step 13), i.e., u is the node in $\mathcal{S}$ with the smallest expected number of transmissions $N^d[u]$, and remove it from $\mathcal{S}$ (step 14). It can be argued (see the next paragraph) that $FL^d[u]$ contains the optimal forwarder list for u with the minimum $N^d[u]$ (among all possible forwarder lists for u to d), and it is therefore removed from $\mathcal{S}$ for further consideration in the future iterations. Given this, we now consider any neighbor node v of u that is still in $\mathcal{S}$ (step 15). If v is a neighbor of u, we *merge* the current best forwarder list $FL^d[v]$ with that of u, $FL^d[u]$, to obtain a new forwarder list for v (step 16). The **merge** operation combines and orders the nodes in $FL^d[v]$ and $FL^d[u]$ (except for v and d) based on the order at which these nodes are removed from the set $\mathcal{S}$: the earlier a node w in $FL^d[u]$ or $FL^d[v]$ is removed from $\mathcal{S}$, the higher the priority of w will be (clearly d has the highest priority, and v the lowest), and the new *merged* forwarder list $FL^d[v]$ is thus of the form $\{v, u, \ldots, d\}$[4]. We then update $N^d[v]$ with the expected number of transmissions using this new merged forwarder list $FL^d[v]$, computed via eq.(3) (step 17). This procedure continues until the optimal forwarder list is computed for all nodes (i.e., until $\mathcal{S}$ is empty).

The optimality of the MTS algorithm can be established by induction and proof-by-contradiction. At the kth iteration, let u_k denote the node u selected in step 13 in Alg. 1, and $FL^d[u_k]$ be the corresponding optimal forwarder list for u_k.

(1) For $k = 1$, it is clear that $u_1 := argmax_v Pr(v \rightarrow d)$ and $FL^d[u_1] = \{u_1, d\}$. We claim $FL^d[u_1] = \{u_1, d\}$ is the optimal forwarder list for u_1. Suppose it is otherwise. Then the optimal forwarder list must contain at least another node (besides u_1 and d). Suppose the optimal forwarder list is $\{u_1, v, d\}$ for some d, from theorem 1 we know that this forwarder list is not optimal, as $N_{u_1,v,d} \geq N_{u_1,d}$. Using theorem 3 or the general formula eq.(3), it can be similarly argued that any other forwarder list with more than 3 nodes cannot be optimal.

(2) For more general $k = 2, \ldots, m$, suppose that all of the forwarder lists $FL^d[u_2], \ldots, FL^d[u_m]$ are all optimal ones. Then, for $k = m + 1$, there is $u_{m+1} := argmin_v N^d[v]$ and $FL^d[u_{m+1}] = \{u_{m+1}, v_i, \ldots, v_1, d\}$ is the optimal forwarder list for u_{m+1}. Suppose it is otherwise. We know that it won't be better than $FL^d[u_{m+1}]$ by removing some forwarders or re-ordering the current forwarders, otherwise the forwarder lists of $k = 1, \ldots, m$ won't be optimal. Then the optimal forwarder list of u_{m+1} must contain at least another node (besides $u_{m+1}, v_i, \ldots, v_1, d$) as the lowest forwarder. Suppose the optimal forwarder list is $\{u_{m+1}, v_{i+1}, \ldots, v_1, d\}$, we need to prove $N^d_{u_{m+1},v_{i+1},\ldots,v_1,d} > N^d_{u_{m+1},v_i,\ldots,v_1,d}$.

We have $u_{m+1} := argmin_v N^d[v]$, so for any other node $v_{i+1} \in \mathcal{S}$, there is $N^d_{v_{i+1},v_i,\ldots,v_1,d} > N^d_{u_{m+1},v_i,\ldots,v_1,d}$. Hence, based on theorem 4, we have $FL^d[u_{m+1}] = \{u_{m+1}, v_i, \ldots, v_1, d\}$ is optimal forwarder list for u_{m+1}, since adding any additional forwarder wouldn't help reducing the expected number of transmissions further.

Based on (1)and (2), we conclude that MTS algorithm can select optimal forwarder list for any source-destination pair under the perfect ACK assumption.

C. Illustration

We now illustrate how the MTS algorithm works using some examples and compare the resulting optimal forwarder lists with those obtained using the ExOR algorithm.

[4]In fact, unless the forwarding lists $FL^d[v]$ and $\{v, FL^d[u]\}$ contain *disjoint path segments* from v to d (in other words, there exist nodes in $FL^d[v]$ but not in $\{v, FL^d[u]\}$ or vice versa), the merge operation would produce a straightforward new (merged) forwarder list, namely, **merge**$(FL^d[v], FL^d[u]) = \{v, FL^d[u]\}$. For example, this would be the case when any node (except for v and d) in $FL^d[v]$ is a neighbor of some node in $FL^d[u]$—which is likely to be true in a reasonably dense network. In any case, it can be shown that by including u and the intermediate forwarders in its forwarder list $FL^d[u]$ into v's current best forwarder list $FL^d[v]$, the expected number of transmissions using the new merged forwarder list $N^d_{FL^d[v]}$ will always be smaller than $N^d[v]$ (computed using the best forward list $FL^d[v]$ before the merge operation).

TABLE IV
FORWARDER LIST SELECTION FOR TOPOLOGY 3(B)

Iteration	source s	source v_1	source v_2	source v_3
1	$\{s,d\}, 2$	$\{v_1,d\}, 1.3333$	$\{v_2,d\}, 1.4268$	$\{v_3,d\}, 1.0526$
2	$\{s,v_3,d\}, 1.9139$	-	$\{v_2,v_3,d\}, 1.3198$	-
3	$\{s,v_2,v_3,d\}, 1.8240$	$\{v_1,v_2,v_3,d\}, 1.3314$	-	-
4	$\{s,v_1,v_2,v_3,d\}, 1.8083$	-	-	-

TABLE V
FORWARDER LIST SELECTION FOR TOPOLOGY 4(A)

Iteration	source s	source v_1	source v_2	source v_3
1	$\{s,d\}, 5$	$\{v_1,d\}, 3.33$	$\{v_2,d\}, 1.11$	$\{v_3,d\}, 1.177$
2	-	$\{v_1,v_2,d\}, 1.8280$	-	$\{v_3,v_2,d\}, 1.1742$
3	$\{s,v_3,v_2,d\}, 3.907$	$\{v_1,v_3,v_2,d\}, 1.8183$	-	-
4	$\{s,v_1,v_3,v_2,d\}, 2.5015$	-	-	-

TABLE VI
FORWARDER LIST SELECTION FOR TOPOLOGY 4(B)

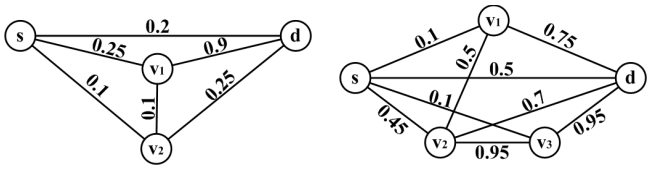
Iteration	source s	source v_1	source v_2	source v_3
1	$\{s,d\}, 5$	$\{v_1,d\}, 3.33$	$\{v_2,d\}, 1.11$	$\{v_3,d\}, 1.177$
2	-	$\{v_1,v_2,d\}, 1.8280$	-	-
3	$\{s,v_3,d\}, 3.908$	-	-	-
4	$\{s,v_1,v_3,v_2,d\}, 2.5083$	-	-	-

TABLE II
FORWARDER LIST SELECTION FOR TOPOLOGY 1

Iteration	source s	source v_1	source v_2
1	$\{s,d\}$ 2	$\{v_1,d\}$ 2.222	$\{v_2,d\}$ 1.25
2	$\{s,v_2,d\}$ 1.9318	$\{v_1,v_2,d\}$ 1.7416	-
3	$\{s,v_1,v_2,d\}$ 1.8566	-	-

TABLE III
FORWARDER LIST SELECTION FOR THE TOPOLOGY 3(A)

Iteration	source s	source v_1	source v_2
1	$\{s,d\}$ 5	$\{v_1,d\}$ 1.1	$\{v_2,d\}$ 4.0
2	$\{s,v_1,d\}$ 3.0556	-	$\{v_2,v_1,d\}$ 3.333
3	-	-	$\{v_2,s,v_1,d\}$ 3.2856



(a) An example topology to show when our theory can select less forwarders than ExOR.

(b) An example topology to show when our theory can choose forwarder list with same size but better orders over ExOR.

Fig. 3. Example topologies

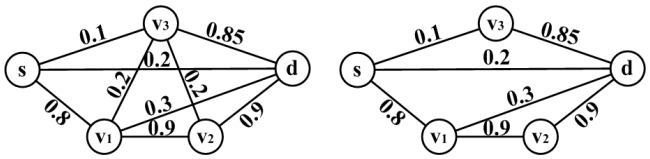


Fig. 4. Example topologies to show the merge operation.

First consider the simple 4-node topology in Fig.1 with d as the destination. The result of each iteration of the MTS algorithm is shown in Table II, where the first item in each cell is the current best forwarder list for the corresponding node, and the second item is the current smallest expected transmission number using the said list. In the first iteration, the optimal forwarder list $\{v_2,d\}$ (and $N_{v_2,d} = 1.25$) is computed for node v_2; in the second iteration, the optimal forwarder list $\{v_1,v_2,d\}$ (and $N_{v_1,v_2,d} = 1.7416$) is computed for node v_1; and in the third iteration, the optimal forwarder list $\{s,v_1,v_2,d\}$ (and $N_{s,v_1,v_2,d} = 1.8566$) is computed for node s. In contrast, using ExOR's forwarder list selection scheme, the resulting forwarder lists for v_2 and v_1 are $\{v_2,d\}$, $\{v_1,v_2,d\}$, the same as produced by the MTS algorithm. But for source s, ExOR selects the forwarder list $\{s,v_2,d\}$, a shorter list than the optimal one ($\{s,v_1,v_2,d\}$) produced by MTS, with a slightly larger expected number of transmissions $N_{s,v_2,d} = 1.9318$ (computed using Theorem3).

To further compare the MTS and ExOR algorithms, we consider two more example topologies shown in Fig.3(a) and Fig.3(b). For the topology in Fig.3(a), the iterative optimal forwarder list selection computations using the MTS algorithm are shown in table III and table IV. For the source s in Fig.3(a),

the optimal forwarder list computed by the MTS algorithm is $\{s, v_1, d\}$. In contrast, ExOR selects the forwarder list $\{s, v_2, v_1, d\}$, a *longer* forwarder list than the optimal one by unnecessarily adding v_2 into the forwarder list. For the source s in Fig.3(b), the optimal forwarder list computed by the MTS algorithm is $\{s, v_1, v_2, v_3, d\}$, with $N_{s,v_1,v_2,v_3,d} = 1.8083$. In contrast, ExOR selects the forwarder list $\{s, v_2, v_1, v_3, d\}$, of the same length as the optimal one, but in different node order. We see that because ExOR simply considers the "best path" from each forwarder to the destination, it ranks v_1 higher than v_2, ignoring the fact that node v_2 has two opportunistic routes $v_2 \rightarrow v_3 \rightarrow d$ and $v_2 \rightarrow d$, which together yield a lower number of transmissions than using v_1's single path to d alone.

Now, we will use two topologies in fig 4 to show how to do the merge operation during each iteration of MTS scheme. The iterative optimal forwarder list selection computations using the MTS algorithm are shown in table V and table VI, respectively. In the topology in fig 4(a), at any time we do the merge operation for $FL^d[v] = merge(FL^d[v], FL^d[u])$, we could simply merge them as $FL^d[v] = \{v, FL^d[u]\}$. For example, when we update node s's forwarder list in iteration 4, we have $FL^d[s] = merge(FL^d[s], FL^d[v_1]) = \{s, FL^d[v_1]\}$. That's because for this merge operation, every forwarder in $FL^d[s]$ is also in the $FL^d[v_1]$. However, if we remove two links ($v_1 \rightarrow v_3$ and $v_3 \rightarrow v_2$), the topology transfers to that in fig 4(b). In this case, there are two disjoint paths from node s to d, when we do the merge operation at s in iteration 4, this merge formula $FL^d[s] = \{s, FL^d[v_1]\}$ will not work any more. That's because there is some node(v_3) in $FL^d[s]$ is not in $FL^d[v_1]$. In this case, we need to use more general merge operation to process.The exact merge operation should be

1) Select all the forwarders of two lists in the new forwarder list.
2) Order these forwarders by their expected number of transmissions to the destination.

In this way, node v_1,v_2,v_3 are all selected into s's forwarder list in iteration 4, and based on their expected number of transmissions,we successfully get $FL^d[s] = \{s, v_1, v_3, v_2, d\}$ as node s's forwarder list.

V. Handling Imperfect ACKs

So far we have developed an optimal forwarder selection algorithm that minimizes the expected total number of transmissions under the perfect ACK assumption. In practice, due to unreliable transmissions (e.g., due to *asymmetric link qualities*), there will be likely *imperfect* ACKs. Such imperfect ACKs would lead to later forwarders no hearing the transmissions of the previous forwarders (or source), resulting in (un-necessary) *duplicate transmissions*.To minimize imperfect ACKs due to unreliable transmissions and asymmetric link qualities, we adopt two heuristic mechanisms, *batch mode* and *two-way link quality formula*, briefly described below. We provide simulation results to show that these two mechanisms together can indeed approximate the perfect ACK assumption fairly well in most scenarios.

Batch Mode. We adopt the batch mode data forwarding mechanism and its associated *(cumulative) bitmap ACK* scheme. Using the batch mode, packets are grouped in a batch of certain size (e.g., $b = 100$ packets), and each packet carries a batch id and packet id (its relative position in the batch) and a *batch map*, where the ith entry of the map contains the id of highest priority node (the destination or a forwarder, based on their relative order in the forwarder list) that has received packet i in the batch; originally it contains the id of the source. When a forwarder broadcasts the packets of a batch it has received, each of them carries the same batch map. Hence upon receiving only one of these packets, a later forwarder (in the forwarder list) will know the status of the packets that have been received by previous forwarders (including the destination) and update its batch map accordingly (please refer to [2] for details). As a result, the batch mode enhances the probability of overhearing among the forwarders and eliminates unnecessary duplicate transmissions.

To show the efficacy of batch mode, especially, the effect of batch size on reducing unnecessary transmissions, Fig. 5 compares the expected total number of transmissions under *perfect ACK* (i.e., assuming all nodes can hear each other's transmissions perfectly) (shown as the *ideal no. of Tx* in the figure) with the (average) number of transmissions actually required with unreliable and asymmetric links using batch mode of varying size. Here the simulations are conducted using the MIT Roofnet dataset (see Section VI for more details regarding the simulation settings). For each source-destination pair, the (optimal) forwarder list is computed using the MTS algorithm, and the *average* number of transmissions is computed over 100 simulation runs with different random seeds. The x-axis is the source-destination pair id ordered based on its *ideal no. of Tx.*, and the y-axis is the average or expected of transmissions. From the figure we see that as we increase the batch size, the average number of actual transmissions (which include duplicate transmissions) decreases, and approaches to the ideal number of transmissions with *perfect ACK*. Similar improvements can be observed also when we choose forwarder lists using ExOR.

Two-way Link Quality Formula. To deal specifically with asymmetrical link qualities and discourage the use of links with drastic different packet delivery probabilities along its two directions, we introduce the following two-way link quality formula to re-define the packet delivery probability from one node to another and use it in the computation of expected number of transmissions (e.g., as in eq.(3)):

$$p'_{ij} = p_{ij}[1 - (1 - p_{ji})^{S_{ij}^{ACK}}] \tag{6}$$

where S_{ij}^{ACK} is a parameter that depends on the batch size. In this paper, for batch size b=100, we set $S_{ij}^{ACK} := S^{ACK} =$

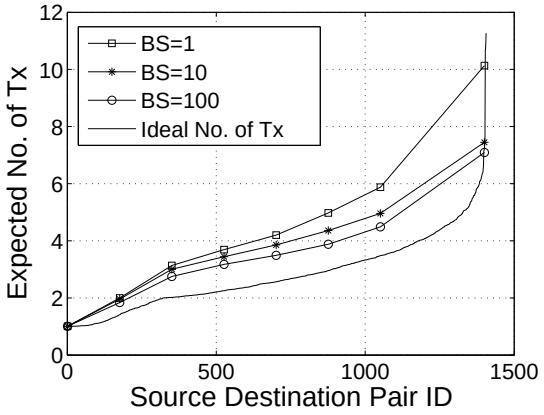


Fig. 5. Batch mode reduces the effect of imperfect ACKs.

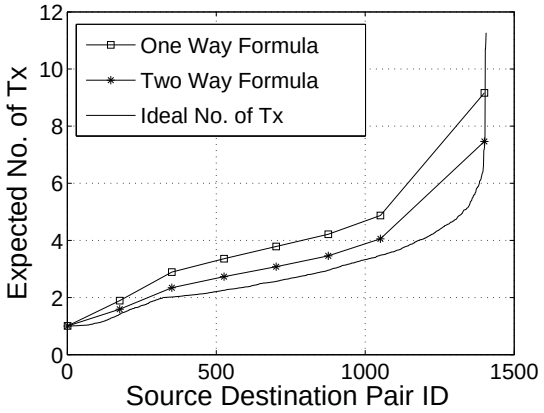


Fig. 6. Effect of two-way link formula 6 (batch size $b = 100$ and $S_{ACK} = 10$).

$0.1b = 10$ for all links[5]. This formula takes into account the link qualities of both directions as well as the effect of batch-mode based cumulative ACKs. Comparing two links, one link with the forward link quality p_{12} is 90% and backward link quality p_{21} is 5%, and other link with $p_{13} = p_{31} = 50\%$. Using the two-way link formula, we have $p'_{12} = 36.11\% < p'_{13} = 49.95\%$ (using $S^{ACK} = 10$). In other words, in order to reduce unnecessary duplicate transmissions due to ACK losses, it would be better for node 1 to use node 3 as a higher priority forwarder than node 2. Fig.6 shows the effect of using the two-way link quality formula which discourages the use of bad asymmetric links in our MTS algorithm, and as a result, further the number of unnecessary retransmissions. (The same observation also holds when the ExOR forwarder list selection algorithm is used.) Hence with a relative large batch size (e.g., 100 packets) and the two-way link quality formula, we can reasonably approximate the perfect ACK assumption and perform forwarder list selection accordingly.

[5]Note that the packets within a batch received by a node j (as an intermediate forwarder) will differ, depending on the senders and rounds of transmissions. Hence in general it will be difficult to precisely set S_{ij}^{ACK}. To be fairly conservative, we choose $S_{ij}^{ACK} = 0.1b$ in our study. Alternatively for a fixed source-destination pair and batch size, we could conduct experiments and use estimated average ACK size based on measurements to set S_{ij}^{ACK} for each link. Our simulation study in fact shows that with relatively large batch size, say, b=100, varying S_{ij}^{ACK} does not significantly affect the ordering and selection of forwarder lists in most scenarios.

VI. Performance Evaluation and Comparison

We conduct extensive simulations in $ns2$ [5] to evaluate the performance of our MTS algorithm and compare it with the performance of ExOR. The simulation results reported here are based on the MIT Roofnet topology and dataset [1]. There are 38 nodes in the Roofnet topology, and the link quality (or packet delivery probability) between any two nodes is derived from the Roofnet data. The simulation parameters are listed in TableVII. There are a total of 1406 source-destination pairs in the Roofnet topology. For each source-destination (in shorthand, src-dst) pair, we run 20 simulations with different random seeds, and the averages of these 20 simulation runs are reported in the discussion below.

TABLE VII
Simulation Parameters

Parameters	Values
Batch Size & S^{ACK}	100 & 10
Bandwidth	$1Mbps$
Forwarder list selection schemes	MTS, ExOR
Topology	Roofnet
Transmission protocol	UDP
Period of simulation	150s for each simulation
Dataflow time	120s for each simulation

We first provide some statistics regarding the forwarder lists selected by the MTS algorithm and the ExOR scheme. Out of the 1406 src-dst pairs, MTS and ExOR select exactly the same forwarder lists for only 225 pairs, about 16% of the total. Among the 84% pairs that are different, MTS scheme selects a smaller forwarder list than ExOR for 792 pairs (57%), and a longer forwarder list for 274 pairs (19%). For the remaining 115 pairs (8%), the forwarder lists selected by MTS and ExOR are of the same size, but of different nodes or order. Fig.7 shows and compares the forwarder list sizes generated by MTS and ExOR for all src-dst pairs in a scatter plot, where each point (x, y) represents the sizes of forwarder list generated by ExOR (x) and MTS (y) for a given src-dst pair. We now compare the performances of the forwarder lists selected by MTS vs. ExOR for each src-dst pair, in terms of both the (average) *actual* number of transmissions (including duplicates), as shown in Fig.8, as well as the throughput (or goodput, i.e., the number of bytes transmitted from the source to the destination per unit time, measured in KB/sec), as shown in Fig.9. In the figures, each dot corresponds to one src-dst pair, where its coordinate (x, y) represent the results under ExOR and MTS, respectively. We see that overall the forwarder lists selected by MTS outperforms that by ExOR: 92.05% src-dst pairs have fewer number of transmissions under MTS than under ETX scheme, and 90.89% src-dst pairs have larger throughputs under MTS than under ETX. There are a small number of pairs for which MTS produces poorer performances than ExOR.

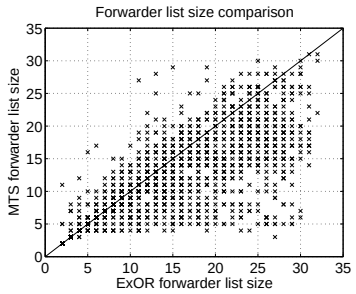


Fig. 7. Forwarder list size comparison.

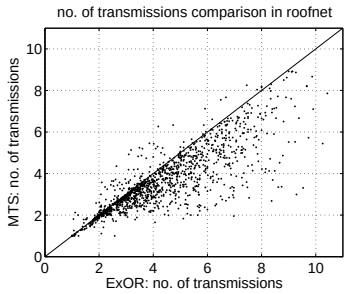


Fig. 8. No. of transmissions.

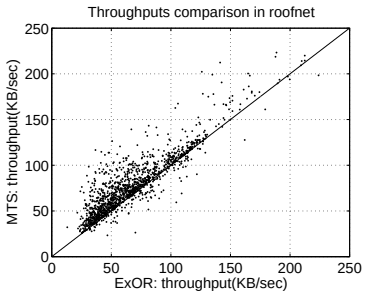


Fig. 9. Throughput.

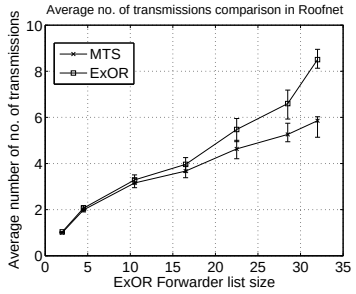


Fig. 10. No. of transmissions comparison.

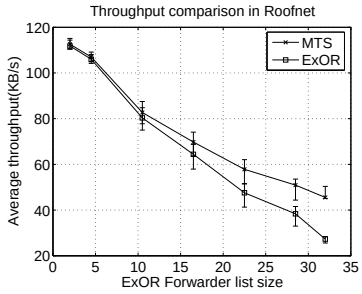


Fig. 11. Throughputs comparison.

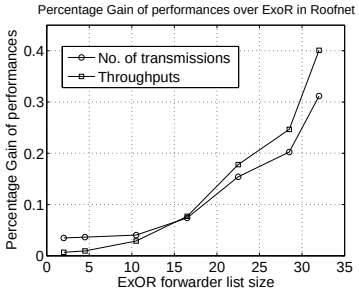


Fig. 12. Percentage gain of performances.

Detailed analysis shows that this is mostly due to the effect of imperfect ACKs, causing unnecessary duplicate transmissions. To further analyze and compare the performances of MTS and ExOR, we break down the results based on the size of forwarder lists, as shown in Figs.10,Figs.11 and Figs.12, where in the Figs.12 the performance gains are computed using the following formulas: $G_{Tx} = (Tx_{ETX} - Tx_{MTS})/Tx_{ETX}$ and $G_{Thput} = (Thput_{MTS} - Thput_{ETX})/Thput_{ETX}$. From these figures, we see that MTS produces the largest performance gains, in terms of both reduced (average) number of transmissions and increased throughput, when forwarder lists (generated by ExOR) are fairly long. For these long forwarder lists generated by ExOR, MTS can reduce the number of transmissions by up to 32.2%, and increase the throughput by up to 40.98%.

Related Work. Before we conclude this paper, we briefly discuss some related work. Zhong *et al.* [9] propose a new routing metric EAX (expected anycast transmissions) to capture the expected number of transmissions needed to opportunistically deliver a packet between two nodes, and resort to heuristic algorithms for computing a set of candidate forwarders. Dubois-Ferriere *et al.* [4] introduces a specific cost function defined with respect to a set of candidate forwarders, and propose the LCOR (least-cost opportunistic routing) algorithm to identify the best candidate set that minimizes the said cost function. Due to its potentially exponential time complexity, heuristic policies have to be incorporated in LCOR. [8] studies the end-to-end throughput, or capacity, of opportunistic routing in *multi-rate* wireless networks using a linear programming framework. There are also a number of other studies and protocols with various opportunistic routing flavors. Due to space limitation, we will not discuss them here.

VII. Conclusion

In this paper we established a general theory for analyzing the forwarder list selection problem, and developed an optimal forwarder selection algorithm–the *minimum transmission selection (MTS)* algorithm–which minimizes the expected number of transmissions under the perfect ACK assumption. We showed how this assumption can be relaxed in practice to account for unreliable and asymmetric link qualities. Through extensive simulations using the MIT Roofnet dataset, we demonstrated that in more than 90% cases the MTS algorithm outperforms the forwarder selection scheme used in ExOR, with performance gains up to 32% in terms of the (average) number of transmissions and up to 41% in terms of throughput.

References

[1] MIT roofnet. *http://pdos.csail.mit.edu/roofnet/*.

[2] S. Biswas and R. Morris. Exor: Opportunistic routing in multi-hop wireless networks. In *Proceedings of the ACM SIGCOMM '05 Conference*, Philadelphia, Pennsylvania, August 2005.

[3] R. R. Choudhury and N. H. Vaidya. Mac-layer anycasting in ad hoc networks. *SIGCOMM Comput. Commun. Rev.*, 34(1):75–80, 2004.

[4] H. Dubois-Ferriere, M.Grossglauser, and M. Vetterli. Least-cost opportunistic routing. In *Allerton Conference on Communication, Control, and Computing*, 2007.

[5] K. Fall and K. Varadhan. *The ns-2 Manual*. The VINT Project, UC Berkeley, LBL, and Xerox PARC, 2003,.

[6] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*. Kluwer Academic Publishers, 1996.

[7] C. Perkins and E. Royer. Ad hoc on demand distance vector routing, mobile computing systems and applications. In *In Proceedings of WMCSA 99*, 1999.

[8] K. Zeng, W. Luo, and H. Zhai. On end-to-end throughput of opportunistic routingin multirate and multihop wireless networks. In *IEEE INFOCOM'08 Conference*, 2008.

[9] Z. Zhong and S. Nelakuditi. On the efficacy of opportunistic routing. *SECON '07. 4th Annual IEEE Communications Society Conference on*, pages 441–450, 18-21 June 2007.