Stochastic Routing in *Ad-Hoc* Networks

Christopher Lott and Demosthenis Teneketzis, *Fellow, IEEE*

***Abstract*—We investigate a network routing problem where a probabilistic local broadcast transmission model is used to determine routing. We discuss this model's key features, and note that the local broadcast transmission model can be viewed as soft handoff for an *ad-hoc* network. We present results showing that an index policy is optimal for the routing problem. We extend the network model to allow for control of transmission type, and prove that the index nature of the optimal routing policy remains unchanged. We present three distributed algorithms which compute an optimal routing policy, discuss their convergence properties, and demonstrate their performance through simulation.**

I. Introduction

The term *ad-hoc* is applied to networks in which there is no central network controller, each node can itself act as a store-and-forward router, and in which the connection topology is time-varying (e.g., see [8]). Such a network is in contrast to, for example, a cellular network where each cell has a central base station through which all cell data is transmitted. An *ad-hoc* network contains a number of packets, each of which is destined for some set of destination nodes. The general routing problem is to define a policy which, given the trajectory histories of all the packets, chooses which nodes should next transmit which packet. It is usually desirable that this policy be implemented in a distributed form, so transmission decisions can be decided locally without knowledge of other parts of the network.

Many approaches are possible for network routing optimization. A typical one is maximizing the overall network throughput. However, in many cases, other considerations are at least as important. In the case of a wireless network the energy source is locally stored in a battery at each node, and a major design goal is often to achieve satisfactory communication while using up as little energy as possible. This low-energy requirement takes an extreme form in *sensor networks*, where large arrays of mobile information-gathering devices must communicate under severe energy limitations [2].

This paper explores some design issues in network routing algorithms for *ad-hoc* networks, and provides a novel system model which allows for optimal design in a number of realistic situations. It is organized as follows. In the remainder of Section I we present briefly a discussion of the literature available on routing in *ad-hoc* wireless networks, and a summary of the contributions of this paper. In Section II, we define and solve a stochastic routing problem. In Section III, we extend this routing problem to include choice of transmission type. In Sections IV–VIII, we present three distributed algorithms which compute an optimal routing policy, and discuss their convergence properties and performance.

Manuscript received October 7, 2004; revised March 25, 2005 and June 8, 2005. Recommended by Associate Editor R. Srikant. This work was supported in part by the Army Research Office under Grant DAAH04-96-1-0377, by the Air Force Office of Scientific Research under Grants F49620-96-1-0028 and F49620-98-1-0370, by the National Science Foundation under Grants ECS-9 979 347 and CCR-0082784, and by the Office of Naval Research under Grant N00014-03-1-0232.

C. Lott is with Qualcomm, Inc., San Diego, CA 92121 USA (e-mail: clott@qualcomm.com).

D. Teneketzis is with the Department of Electrical Engineering and Computer Science, the University of Michigan, Ann Arbor, MI 48109 USA (e-mail: teneketzis@eecs.umich.edu).

Digital Object Identifier 10.1109/TAC.2005.860280

A. Ad-Hoc Network Routing Literature

There has been much recent research activity on routing for *ad-hoc* networks. This work can perhaps most conveniently be categorized as either *on-demand* or *route maintenance* (also sometimes referred to as *reactive* and *proactive* [19]).

1) On-Demand Protocols: On-demand protocols are the algorithms which only update routing tables when new packets arrive. On-demand protocols usually store old routing information and use it as long as packets still get to their destination. When a given path fails, there is some mechanism for probing the network to find a new path, which often involves flooding the network. Link detection used to decide path failure generally uses information from the medium access control (MAC) at the link layer. Usually the path found is not optimal in any sense. The value of an on-demand protocol is that the communication requirement to set up a route is minimized. The idea is that in general it would cost more to find a better path than the improved route would reduce cost. Hence, these protocols can be valuable when network dynamics are rapid compared to packet transmission frequency.

A number of on-demand routing protocols have been proposed for *ad-hoc* networks. The temporally ordered routing algorithm (TORA) [18] is based on work in [7]. TORA tries to quickly establish routes on demand using a path reversal method which is guaranteed to find some route, with no criteria for the quality of the route. Dynamic source routing (DSR) [15] uses source routing, where each packet header contains the entire route for the packet. Existing routes are used without update until one fails. When one fails, a Route Discovery is executed, during which Route Request packets flood the network until any route is found. The *ad-hoc* on-demand distance vector (AODV) protocol [20] is a kind of hybrid of on-demand and route maintenance protocols. The link status between a node and its neighbors is maintained through periodic beacon signals between nodes. However, routes are only updated when a packet routing fails, and then this link status information is used. The route chosen by AODV is the one with the minimum number of hops. The zone routing protocol (ZRP) [19] is another hybrid which allows trading off of route update and route maintenance overheads. The network nodes are partitioned into zones. As in route maintenance, each node maintains a continuous knowledge of the topology of all the nodes within its own zone. However, routes to nodes outside a node's zone are determined on-demand in a route discovery process. Independent zone routing (IZR) is an update to ZRP which allows zone size to vary per node [23].

2) Route Maintenance Protocols: As the name implies, route maintenance protocols periodically expend the energy and bandwidth necessary to maintain updated routing tables under all traffic conditions. Given their wide acceptance, the benchmarks for route maintenance protocols are often the TCP/IP standards routing information protocol (RIP) [14] and open shortest path first (OSPF) [17]. However, the standard Internet implementations of these algorithms are not well-suited to the *ad-hoc* network environment. In highly mobile environments, communication overhead to implement a full link-state algorithm is prohibitively costly. Alternatively, much research has been performed to adapt distance vector type algorithms to the mobile environment. In these approaches, it is common to use a distributed Bellman–Ford (DBF) type algorithm [4] to update the routes and cost estimates. A common theme in the research, then, is to address the weaknesses of this type of algorithm in a highly mobile and uncertain environment.

An early effort along these lines is [16], which endeavors to provide loop-free routing in a quasistatic environment. In least resistance routing (LRR) [21], the link quality between nodes and the current buffer length of the receiving node are used to define a link resistance value. A DBF type algorithm is then used to find a least resistance path for such a network. An interesting approach to the multicast problem is developed in [25]. A deterministic local broadcast model for the link is used to set up a problem to find the minimum energy to span a given set of destination nodes, and heuristic routing algorithms are studied using simulation. This circuit-switched multicast problem with deterministic local broadcast model is quite different from the problems we define in this paper.

A detailed comparison via simulation of a few different routing protocols for *ad-hoc* networks is described in [5]. To truly understand the tradeoffs involved among the wide variety of available algorithms, more such work is necessary. Even better would be progress in our conceptual understanding, leading to better methods for critiquing the different approaches. For a useful guide to the large and growing literature on *ad-hoc* networks, see [8] and the references therein.

B. Contribution

In this work, we present what is, to the best of our knowledge, the first network routing protocol which uses a probabilistic local broadcast model for transmission. This model: i) provides an intimate coupling between lower network layers through careful modeling of the key channel characteristics, and ii) allows for routing decisions to be made based on immediate feedback from each transmission. In most network models in the literature, nodes are connected by links, and the routing algorithm dictates which set of links each packet is to traverse. This is true even when through local broadcast a packet can traverse multiple links in parallel. Typically at each point in time, through an estimate of its SINR, a link is designated as deterministically up or down, with SINR sometimes used to determine the sustainable link rate. The stochastic transmission nature is subsumed in a link cost or delay, or in a success probability for the entire link layer transmission protocol. In contrast, our protocol detects and reacts to the stochastic result of each node's local broadcast transmission, and constructs an optimal route based on this immediate feedback. In consequence, the actual route taken by a packet depends on random system events, may vary among packets heading to a common destination, and is fully determined only via actual transmission. This reacting to individual transmission results can be interpreted as the *ad-hoc* network analog of soft handoff in a cellular system, as discussed further in the following. We show that such a protocol is optimal for the stochastic local broadcast model. We also show the protocol is well suited to a distributed implementation, as it is based only on information local to a node.

We extend the model to allow for control of transmission type at each node, and show how the fundamental nature of this optimal protocol doesn't change. Varying transmission type to optimize routing decisions couples physical layer considerations with the network layer routing function. For example, in the particular case of power control, an optimal protocol effectively resolves the tradeoffs between fewer long hops vs. more short hops.

We present three distinct algorithms which compute an optimal index policy for our stochastic routing problem in an asynchronous distributed fashion, so that at each computation only information local to each node is used. The algorithms are all similar in the asynchronous nature of their value updates and transmissions, with the fundamental difference being how the update value is computed. The algorithm presented in Section V uses an update motivated by the stochastic Dijkstra algorithm we introduce in Section II. The algorithm presented in Section VI implements DDP, as defined in [3], and this leads to an alternate update computation. Finally, the algorithm presented in Section VII, which we call the rank algorithm, is a third way to implement the update. For each algorithm we prove the same fundamental convergence result: 1) Asymptotic convergence of the value function; 2) finite time convergence to an optimal policy; and 3) finite time convergence to an optimal index policy under the condition that all nodes have distinct value functions. We end by comparing the performance of these three algorithms with other approaches in a realistic simulation environment.

There is an interesting precedent for the stochastic local broadcast model used in this paper. In CDMA cellular systems [24], soft handoff is used to improve network efficiency. On the reverse link, even though the mobile is under transmit power control, random channel variation makes it uncertain which base station will successfully decode each mobile packet. Efficient operation is achieved by having each base station attempt to decode each packet, and the packet is successful if any of them succeed. In effect, each base station is a destination node and the random result of each mobile local broadcast is used to decide on packet success. Our stochastic routing problem can be viewed as a generalization of soft handoff to the case of an *ad-hoc* network, where intermediate store-and-forward nodes may also receive each packet.

In CDMA cellular system reverse links [6], [11], a mobile's transmit power is controlled to achieve a fixed packet error rate (PER). Implicitly this relies on the fact that the recent interference level seen by mobile transmissions is a good estimator of future interference. No further attempt is made to coordinate mobile transmissions across the network (interference cancellation is a separate issue). In our approach here, rather than controlling mobile transmit power to achieve a fixed PER, we assume a given set of transmit power levels, and estimate the resulting packet success probabilities to neighbors. By taking this approach, we also implicitly rely on some persistence in interference (and hence transmission success) statistics. In deference to the inherently distributed nature of an *ad-hoc* network, we also do not attempt any further coordination of mobile transmissions, so that each mobile views other mobile transmissions solely as random sources of interference.

Stochastic routing also can be viewed as a means to achieve dynamic network load balancing. In effect, each mobile transmission is a parallel attempt across neighbor channels, and packet success depends on the concurrent loading. Packets thus are naturally routed toward less loaded portions of the network, with minimal retransmissions. For longer term loading imbalances, the transmission probability estimates themselves reflect interference-limited regions of the network. These aspects of stochastic routing are important for the network simulation results presented in Section VIII.

II. Stochastic Routing Problem

A. Notation and Preliminaries

We begin by briefly summarizing notation and definitions for the system model under consideration, which we refer to as Model $(\mathbf{M})$. In Model $(\mathbf{M})$ control is centralized, meaning the controller has access to all information in the network. Also, Model $(\mathbf{M})$ is probabilistic, with transitions described by $P(S \mid i)$.

- N is the number of nodes in the network.
- $\Omega = \{1, \ldots, N\}$ the set of all nodes. So $|\Omega| = N$.
- $S \subseteq \Omega$ refers to a state of the system, defined as the set of nodes which have received the packet.
- S_t refers to the state at time t.
- $R : 2^{\Omega} \to \mathbb{R}$ is the reward function, and $R_i := R(\{i\})$. Also $R_{\max} := \max_{i \in \Omega} R_i$.
- π is a Markov policy. We write $\pi(S) = i$ to indicate policy π transmits at node i when in state S.

We write $\pi(S) = r$ to indicate policy π retires and receives reward $R(S)$ when in state S. For convenience we write $\pi(S) = r_i$ as shorthand that policy π retires and receives $R_i, i \in S$. In this case, we say that policy π retires and *receives the reward* of node i.

By $\pi(S) \neq i, r_i$, we mean both $\pi(S) \neq i$ and $\pi(S) \neq r_i$.

By $\pi(S) = \tilde{\pi}(S)$, we mean either $\pi(S) = \tilde{\pi}(S) = i$, or $\pi(S) = \tilde{\pi}(S) = r_i$, for some i.

$V^{\pi}(S)$ is the expected reward when starting in state S under policy π. We often write V_i^{π} for $V^{\pi}(\{i\})$. When π is optimal, we use $V^{\pi}(\cdot)$ to indicate the optimal value function.

We write $P^i(S' \mid S)$ to indicate the probability of reaching state S' from state S when choosing i for transmission, $i \in S$. We write $P^i(S \mid i)$ as shorthand for $P^i(S \mid \{i\})$.

We define $P_{ij} := \sum_{S: i, j \in S} P^i(S \mid i)$. j is called a *neighbor* of i if $P_{ij} > 0$. $\mathcal{N}(i)$ is the set of all neighbors of i, together with i itself. Note that $P_{ij} \neq P_{ji}$ is permitted.

By $\operatorname{argmax}_{i \in S} f(i)$, we mean the *set* of values of i from the finite set S which maximizes $f(i)$.

Definition 2.1 (Increasing Property): Model $(\mathbf{M})$ is said to be *increasing* if for any system realization under any policy we have $S_{t_2} \supseteq S_{t_1}, \forall t_1, \forall t_2 > t_1$.

Definition 2.2 (Decoupling Property): Model $(\mathbf{M})$ is said to be *decoupled* if $\forall S_1, S_2 \subseteq \Omega, S_3 \subset S_1, i \in S_1 - S_3$ we have

$$P^i(S_2 \mid S_1) = \sum_{S \subseteq S_3} P^i(S_2 - S \mid S_1 - S_3).$$

The meaning of this definition is that transmission success to a set of neighbors from a node at a given time is unaffected by which other nodes already have the packet.

Definition 2.3: A function $f : 2^{\Omega} \to \mathbb{R}$ is an *index function* on Ω if f satisfies

$$f(S) = \max_{i \in S} f(\{i\}) \qquad \forall S \subseteq \Omega \tag{1}$$

We next formulate the centralized version of the stochastic routing problem.

B. Statement of Problem

Problem $(\mathbf{P_1})$

We consider the transmission of a single packet, from a given initial state S_o (i.e., a given set of nodes) to a set of destination states, in a wireless *ad-hoc* network of N nodes described by Model $(\mathbf{M})$. Transmission instances occur at discrete time points. Each transmission from a given node i incurs a fixed cost $c_i > 0$. According to Model $(\mathbf{M})$: i) at each transmission instance the transmitting node is chosen by a central controller that always knows the current state of the system (i.e., the set of nodes that have the packet); ii) node transmissions are local broadcasts, that is, multiple neighbor nodes may all simultaneously receive the packet; iii) given the node chosen to transmit, the probability that a given set of nodes receives the packet is known and fixed; iv) The central controller is informed, without any cost, as to which nodes receive the packet. In general, control information flow between the nodes and the controller is considered free in energy and instantaneous in time; and v) each transmission event is assumed independent of those before and after. We assume Model $(\mathbf{M})$ is increasing and decoupled. A reward function R is specified, where R is an index function. At each instance, the central controller chooses either to terminate the transmission process or to continue transmitting. The objective is to choose: i) the node to transmit at each transmission instance, and ii) the instance to terminate the transmission process, to maximize

$$E\left\{R(S_f) - \sum_{t=1}^{\tau-1} c_{i(t)}\right\} \tag{2}$$

where τ is the time when the transmission process is terminated, S_f is the state at τ, and $i(t)$ is the node chosen by the transmission policy at time t.

C. Analysis of Problem $(\mathbf{P_1})$

We analyze Problem $(\mathbf{P_1})$ and discuss the character of an optimal policy π. The system of Problem $(\mathbf{P_1})$ is a time-homogeneous Markov chain, hence we are faced with a finite-state Markovian decision problem with perfect information. We can thus restrict attention to Markov policies on 2^{Ω}, and we are guaranteed that such an optimal Markov policy exists (cf. [22, Ch. 3, p. 51]). We seek an optimal Markov policy $\pi : 2^{\Omega} \to (1, \ldots, N)$ which minimizes (2).

To solve Problem $(\mathbf{P_1})$, we could directly apply stochastic dynamic programming. However, since the number of states is 2^N, the complexity of such an approach is at least $O(2^N)$, and generally higher. Instead, we use the special structure of this problem to find a better algorithm.

1) Structure of an Optimal Policy for Problem $(\mathbf{P_1})$*:* We begin with some definitions.

Definition 2.4: A Markov policy π is a *priority* policy if there is a strict priority ordering of the nodes s.t. $\forall i \in \Omega$ we have $\pi(S \cup \{i\}) = \pi(\{i\}) = i$ or $r_i, \forall S \subseteq \Omega_i$, where Ω_i is the set of nodes of priority lower than i.

Definition 2.5: For priority policy π, we write $i >_{\pi} j$ when i has higher priority than j under π.

Definition 2.6: A priority policy π is called an *index* policy if $V^{\pi}(\cdot)$ is an index function on Ω.

Note that a priority policy need not be an index policy.

Our main goal in this section is to prove that there exists an index policy which is an optimal Markov policy for Problem $(\mathbf{P_1})$. We state this result in the following theorem.

Theorem 2.1 (Index Policy): There is an optimal Markov policy π for Problem $(\mathbf{P_1})$ which is an index policy.

We develop a series of lemmas which are used to prove Theorem 2.1. In the first lemma we show that the definition of an index function is equivalent to requiring two properties on f.

Lemma 2.1: Function f is an index function on Ω if and only if the following two properties (3) and (4) both hold:

$$f(S \cup \{i\}) \geq f(S) \qquad \forall S \subset \Omega, \quad i \in \Omega \tag{3}$$

$$f(S) \neq f(\{i\}) \Longrightarrow f(S) = f(S - \{i\}) \qquad \forall S \subseteq \Omega, \quad i \in S \tag{4}$$

Proof: Assume f is an index function on 2^{Ω}. Then f can be written in the form (1). We have $\forall S \subseteq \Omega, i \in \Omega$

$$f(S \cup \{i\}) = \max_{j \in S \cup \{i\}} f(\{j\}) \geq \max_{j \in S} f(\{j\}) = f(S) \tag{5}$$

and (5) establishes (3). To establish (4), assume we have an $i \in S$ where $f(S) \neq f(\{i\})$. Then

$$f(S) = \max_{j \in S} f(\{j\}) = \max_{j \in S - \{i\}} f(\{j\}) = f(S - \{i\}) \tag{6}$$

Together (5) and (6) establish that if f is an index function, then (3) and (4) both must hold.

Conversely, assume (3) and (4) hold. We proceed by induction on the number of elements in S. When $|S| = 1$, for any S it is clear that $f(\{i\}) = \max_{j \in \{i\}} f(\{j\})$, so the induction base step is established. Now assume $f(S)$ can be written in the form of (1) $\forall S \subset \Omega$ s.t. $|S| = K$. Let $S' \subseteq \Omega$ be s.t. $|S'| = K + 1$. We consider two cases.

- Case 1): $f(\{i\}) = f(\{j\}) \forall i, j \in S'$
  Assume there is an $i \in S'$ s.t. $f(S') \neq f(\{i\})$. Then by (4) and using the fact that $|S' - \{i\}| = K$ with the inductive hypothesis, we have $f(S') = f(S' - \{i\}) = \max_{j \in S' : j \neq i} f(\{j\}) = f(\{i\})$, which is a contradiction. So we must have $f(S') = f(\{i\}) = \max_{j \in S'} f(\{j\})$, and (1) holds for this case.
- Case 2): $\exists i, j \in S'$ s.t. $f(\{i\}) < f(\{j\})$
  By (3), $f(S') \geq f(\{j\}) > f(\{i\})$. Then by (4), and using the fact that $|S' - \{i\}| = K$ with the inductive hypothesis, we have

$$f(S') = f(S' - \{i\}) = \max_{j \in S' : j \neq i} f(\{j\}) = \max_{j \in S'} f(\{j\}) \tag{7}$$

  and (1) holds for this case.

In both cases, f has the form of (1) for S', and this completes the induction step. So by induction f must be an index function on 2^{Ω}. ■

Next, we use the decoupling and increasing properties of Problem $(\mathbf{P_1})$ to show that the optimal value function for Problem $(\mathbf{P_1})$ possesses a monotonicity property.

Lemma 2.2 (Monotonicity): In Problem $(\mathbf{P_1})$, let π be an optimal Markov policy, and let $S_1, S_2 \subseteq \Omega$ and $S_2 \subseteq S_1$. Then, $V^{\pi}(S_2) \leq V^{\pi}(S_1)$.

Proof: Given π and $S_2 \subseteq S_1$, we define a new policy $\hat{\pi}$ acting on state S_1 as follows. Let $i = \pi(S_2)$, and suppose S_4 is the state resulting if at first π were to choose i. At the first step, $\hat{\pi}$ chooses i from S_1, which is possible since $S_2 \subseteq S_1$. $\hat{\pi}$ learns the result of the transmission, and hence knows the new actual state of the system, which we call S_3. Furthermore, since $\hat{\pi}$ knows which nodes receive the packet even if they already have it, it also knows what the new state would be if the previous state were S_2 instead of S_1. By the decoupling property, this new state is S_4. This fact together with the increasing property also imply that $S_4 \subseteq S_3$, since $S_2 \subseteq S_1$.

At the next step $\hat{\pi}$ acts on S_3 by choosing the same node as π would use on S_4; this is possible because $\hat{\pi}$ knows S_4, and $S_4 \subseteq S_3$. The process continues in this way until $\hat{\pi}$ retires at the same time at which π would retire. Policy $\hat{\pi}$ knows π's retirement time because it knows the state π "sees" at each time. Let S_{f1} and S_{f2} be the states at retirement for $\hat{\pi}$ and π, respectively. By the above argument, we know $S_{f2} \subseteq S_{f1}$. At retirement, $\hat{\pi}$ has incurred the same cost as π, since π and $\hat{\pi}$ use the same nodes to transmit. Because R is an index function, by Lemma 2.1 (3) we have

$$V^{\hat{\pi}}(S_1) - V^{\pi}(S_2) = R(S_{f1}) - R(S_{f2}) \geq 0. \tag{8}$$

Because π is optimal and $\hat{\pi}$ is suboptimal, we conclude from (8) that

$$V^{\pi}(S_1) \geq V^{\hat{\pi}}(S_1) \geq V^{\pi}(S_2). \tag{9}$$

This completes the proof. ■

In the next lemma, we use the increasing property of Problem $(\mathbf{P_1})$ to show that a Markov policy which is optimal for all states that are a superset of some $S_1 \subset \Omega$, and that takes an optimal action when in S_1, is also optimal when in state S_1.

Lemma 2.3: Let $\tilde{\pi}$ be an optimal Markov policy for Problem $(\mathbf{P_1})$. Suppose we are given S_1, and let π be a Markov policy which has the following two properties:

$$V^{\pi}(S) = V^{\tilde{\pi}}(S) \qquad \forall S \supset S_1 \tag{10}$$

$$\pi(S_1) = \tilde{\pi}(S_1). \tag{11}$$

Then

$$V^{\pi}(S_1) = V^{\tilde{\pi}}(S_1). \tag{12}$$

Proof: If $\pi(S_1) = \tilde{\pi}(S_1) = r_i$, for some $i \in S_1$, then (12) holds.

Suppose $\pi(S_1) = \tilde{\pi}(S_1) \neq r$. We compare π and $\tilde{\pi}$ when both transmit in state S_1. Let $\tilde{S}_2$ and S_2 be the state after transmitting when in S_1 for $\tilde{\pi}$ and π, respectively. Due to (11), we have

$$\tilde{S}_2 = S_2 \tag{13}$$

Due to the increasing property, we have

$$\tilde{S}_2 = S_2 \supseteq S_1. \tag{14}$$

By (10), we have

$$V^{\tilde{\pi}}(\tilde{S}_2) = V^{\pi}(S_2) \quad \tilde{S}_2 = S_2 \supset S_1. \tag{15}$$

Equations (13)–(15) mean that $\tilde{\pi}$ and π choose the same node from S_1 for transmission, and either reach the same state $S_2 \supset S_1$, which has the same value function for both policies, or both stay in $S_2 = S_1$, at which point they again both play the same node for transmission. Hence, (12) follows. ■

Next, we construct a Markov policy for Problem $(\mathbf{P_1})$ which has many characteristics necessary for an index policy, and then use the lemmas presented above to show that this policy is optimal. The result of this lemma is instrumental in proving Theorem 2.1.

Lemma 2.4: Let $\tilde{\pi}$ be an optimal Markov policy for Problem $(\mathbf{P_1})$. Then there exists a Markov policy π which has the following three properties.

1) For all $S \subseteq \Omega$ where $|S| \geq 2$

$$\pi(S) = i \Longrightarrow \pi(S - \{j\}) = i \qquad \forall j \in S, \quad j \neq i \tag{16}$$

$$\pi(S) = r_i \Longrightarrow \pi(S - \{j\}) = r_i \qquad \forall j \in S, \quad j \neq i. \tag{17}$$

2) For all $S \subseteq \Omega$ where $|S| \geq 2$, and $\pi(S) = i, r_i$

$$V^{\pi}(S - \{j\}) = V^{\pi}(S) = V^{\tilde{\pi}}(S) = V^{\tilde{\pi}}(S - \{j\}) \qquad \forall j \in S, \quad j \neq i. \tag{18}$$

3) π is an optimal Markov policy.

Proof: We define π using the following rules:

$$\pi(\Omega) = \tilde{\pi}(\Omega) \tag{19}$$

$$\pi(S - \{j\}) = \pi(S) \qquad \forall S \subseteq \Omega, \quad \forall j : \pi(S) \neq j, r_j \tag{20}$$

$$\pi(S - \{j\}) = \tilde{\pi}(S - \{j\}) \qquad \forall S \subseteq \Omega \quad j : \pi(S) = j, r_j. \tag{21}$$

If $|\Omega| = N = 1$, by (19) the lemma is true. Assume $|\Omega| = N \geq 2$.

It follows directly from (19)–(21) that π satisfies (16) and (17).

We prove (18) by backward induction on the cardinality of S. By the optimality of $\tilde{\pi}$, we know that $\tilde{\pi}(\Omega) = r_i$ for some $i \in \mathrm{argmax}_{j \in \Omega} R_j$. By (19) and (20) we have $\pi(\Omega) = \pi(\Omega - \{j\}) = r_i, \forall j \in \Omega, j \neq i$. That is, π acting on both Ω and $\Omega - \{j\}$ immediately retires and receives reward R_i. We also have $V^{\tilde{\pi}}(\Omega) = R_i = V^{\tilde{\pi}}(\Omega - \{j\})$, because $\tilde{\pi}$ is optimal and i is available for retirement in $\Omega - \{j\}$. Hence, when $\tilde{\pi}(\Omega) = r_i$, we have $\forall j \in \Omega, j \neq i$

$$V^{\tilde{\pi}}(\Omega - \{j\}) = V^{\tilde{\pi}}(\Omega) = V^{\pi}(\Omega) = V^{\pi}(\Omega - \{j\}) = R_i. \tag{22}$$

Equality (22) proves (18) for π when $S = \Omega$, and the basis for induction is established.

If $N = 2$, the argument of (22) completes the proof of (18). We now assume $N > 2$ and prove the induction step. Assume (18) is true for any state S where $|S| \geq L + 1, 2 \leq L < N$. Consider any state S_1 where $|S_1| = L$.

We first prove that

$$V^{\pi}(S_1) = V^{\tilde{\pi}}(S_1). \tag{23}$$

If there exists $j \in \Omega - S_1$ such that $\pi(S_1 \cup \{j\}) \neq j, r_j$, then by (20) we have $\pi(S_1) = \pi(S_1 \cup \{j\})$. By the induction hypothesis, (18) is true for $S = S_1 \cup \{j\}$, because $|S_1 \cup \{j\}| = L + 1$. We thus have

$$V^{\pi}(S_1 \cup \{j\} - \{j\}) = V^{\tilde{\pi}}(S_1 \cup \{j\} - \{j\}). \tag{24}$$

Equation (23) follows from (24).

If no such j exists, then by (21) we have $\pi(S_1) = \tilde{\pi}(S_1)$. Because $\pi(S_1) = \tilde{\pi}(S_1)$, and $V^{\pi}(S) = V^{\tilde{\pi}}(S), \forall S \supset S_1$ (because of the induction hypothesis), conditions (10) and (11) of Lemma 2.3 are satisfied by π and $\tilde{\pi}$ for S_1. Hence

$$V^{\pi}(S_1) = V^{\tilde{\pi}}(S_1). \tag{25}$$

We have shown that (23) holds for any S_1 where $|S_1| = L$. We use (23) to show that (18) holds for all S_1 where $|S_1| = L$. For the remainder of the proof, assume that either $\pi(S_1) = i$ or $\pi(S_1) = r_i$, and let $j \in S_1, j \neq i$.

Consider first the case where $\pi(S_1) = r_i$. By (20), $\pi(S_1 - \{j\}) = r_i$, so that

$$V^{\pi}(S_1 - \{j\}) = V^{\pi}(S_1) = R_i. \tag{26}$$

By Lemma 2.2 and the optimality of $\tilde{\pi}$, we have

$$V^{\tilde{\pi}}(S_1 - \{j\}) \leq V^{\tilde{\pi}}(S_1). \quad (27)$$

By (23)

$$V^{\tilde{\pi}}(S_1) = V^{\pi}(S_1) = R_i. \quad (28)$$

However, $i \in S_1 - \{j\}$, and $\tilde{\pi}$ is an optimal policy, so

$$V^{\tilde{\pi}}(S_1 - \{j\}) \geq R_i. \quad (29)$$

Relations (27)–(29) together imply that

$$V^{\tilde{\pi}}(S_1 - \{j\}) = V^{\tilde{\pi}}(S_1) = R_i. \quad (30)$$

Equations (23), (26), and (30) imply that

$$\begin{aligned} V^{\tilde{\pi}}(S_1 - \{j\}) &= V^{\tilde{\pi}}(S_1) = V^{\pi}(S_1) \\ &= V^{\pi}(S_1 - \{j\}) = V^{\tilde{\pi}}(S_1 - \{j\}) \quad (31) \end{aligned}$$

and the induction step for (18) is proved when $\pi(S_1) = r_i$.

Now, consider the case where

$$\pi(S_1) = i. \quad (32)$$

We claim that

$$\pi(S) \neq j, r_j \qquad \forall S \supseteq S_1 - \{j\}. \quad (33)$$

We prove (33) as follows. Let S be any state where $S \supseteq S_1 - \{j\}$. If $j \notin S$, then $\pi(S) \neq j, r_j$, and (33) follows. Assume $j \in S$. Then $S \supseteq S_1$ and $|S| \geq L$. If $|S| = L$, then $S = S_1$ and $\pi(S) = i \neq j, r_j$. Assume $|S| > L$. If $\pi(S) = j$, then by successively removing nodes $k \in \Omega - S_1$ and using (20) we obtain $\pi(S_1) = j$, which contradicts (32). If $\pi(S) = r_j$, then by successively removing nodes $k \in \Omega - S_1$ and using (20), we obtain $\pi(S_1) = r_j$, which contradicts (32). Hence, (33) is true in all cases.

By the decoupling property and (33), we have $V^{\pi}(S_1 - \{j\}) = V^{\pi}(S_1)$. Using $V^{\pi}(S_1 - \{j\}) = V^{\pi}(S_1)$ with (23), Lemma 2.2, and noting the optimality of $\tilde{\pi}$, we have

$$\begin{aligned} V^{\pi}(S_1 - \{j\}) &= V^{\pi}(S_1) = V^{\tilde{\pi}}(S_1) \\ &\geq V^{\tilde{\pi}}(S_1 - \{j\}) \geq V^{\pi}(S_1 - \{j\}). \quad (34) \end{aligned}$$

Relation (34) proves the induction step for (18) for $\pi(S_1) = i$.

This completes the induction step for (18). By induction, we have proved that (18) is true for π for all $S \subseteq \Omega$ where $|S| \geq 2$.

Finally, we prove that π is an optimal policy. First, note that by (18),

$$V^{\pi}(S) = V^{\tilde{\pi}}(S) \qquad \forall S, |S| \geq 2 \quad (35)$$

Relation (18) also implies that for some $i \in \Omega$

$$V^{\pi}(\{j\}) = V^{\tilde{\pi}}(\{j\}) \qquad \forall j \in \Omega \text{ s.t. } \pi(\{i\} \cup \{j\}) = j \quad (36)$$

We are left to consider $V^{\pi}(\{j\})$ when no such i as in (36) exists. By (21) we have $\pi(\{j\}) = \tilde{\pi}(\{j\})$ when $\pi(\{j\} \cup \{i\}) = i, \forall i \in \Omega$. Under this condition and (18), and identifying $S_1 = \{j\}, \pi$, and $\tilde{\pi}$ satisfy the requirements of Lemma 2.3 (10) and (11). Lemma 2.3 (12) then implies that

$$\begin{aligned} &V^{\pi}(\{j\}) = V^{\tilde{\pi}}(\{j\}) \\ &\forall j \in \Omega \text{ s.t. } \pi(\{j\} \cup \{i\}) = i \qquad \forall i \in \Omega \quad (37) \end{aligned}$$

Equations (35), (36), and (37) prove the optimality of π. The proof of the lemma is complete. ■

Besides Lemma 2.4, we need one more result to prove Theorem 2.1, which is that the value function for Problem $(\mathbf{P_1})$ is always an index function. We present this lemma next.

Lemma 2.5: For any optimal Markov policy $\tilde{\pi}, V^{\tilde{\pi}}(\,\cdot\,)$ is an index function on Ω.

Proof: First note that Lemma 2.4 implies Lemma 2.1 (3) is satisfied for $V^{\tilde{\pi}}(\,\cdot\,)$ on Ω.

Next, let π be the Markov policy satisfying (16) and (17) as constructed in Lemma 2.4. Consider any state S and $i \in S$ s.t. $V^{\pi}(S) \neq V^{\pi}(\{i\})$. If $\pi(S) = i$, then by removing all nodes except i from S via repeated application of Lemma 2.4 we would get $V^{\pi}(S) = V^{\pi}(\{i\})$, a contradiction. Hence, $\pi(S) \neq i$. So by Lemma 2.4 (18), we have $V^{\tilde{\pi}}(S) = V^{\tilde{\pi}}(S - \{i\})$. Thus the requirement (4) of Lemma 2.1 is satisfied for $V^{\tilde{\pi}}(\,\cdot\,)$ on Ω.

Since both requirements of Lemma 2.1 are satisfied, we have shown that $V^{\tilde{\pi}}(\,\cdot\,)$ is an index function on Ω. ■

We now use Lemmas 2.1–2.5 to prove Theorem 2.1.

Proof: **[of Theorem 2.1]**
Let π be the Markov policy satisfying (16) and (17) as constructed in Lemma 2.4. By Lemma 2.4 (3.), π is an optimal Markov policy. Hence, Lemma 2.5 indicates that $V^{\pi}(\,\cdot\,)$ is an index function.

We next show that π is a priority policy. By (16) and (17), we have

$$\pi(S) = i \Longrightarrow \pi(S') = i \qquad \forall S' \subseteq S, \quad i \in S' \quad (38)$$

$$\pi(S) = r_i \Longrightarrow \pi(S') = r_i \qquad \forall S' \subseteq S, \quad i \in S' \quad (39)$$

Properties (38) and (39) show that π is a is a priority policy (cf. Definition 2.4), with node priority as follows. For any S where $\pi(S) = i$ or $\pi(S) = r_i$, i has priority higher than all other nodes of S.

We have shown that π is a priority policy with index function $V^{\pi}(\,\cdot\,)$. Hence, π satisfies Definition 2.6, and is an index policy. ■

Note that (38) and (39) imply that $\pi(S_1) = i$ and $\pi(S_2) = r_i$ cannot both occur for a given π for any S_1, S_2 with $i \in S_1$ and $j \in S_2$. That is, for a given system for Problem $(\mathbf{P_1})$ and an optimal index policy π, if there is a state where π transmits from node i, then there is no state where π retires and receives the reward from i. Similarly, if there is a state where π retires and receives the reward from node i, then there is no state where π transmits at i. We henceforth use this fact when we write the node priority list for an index policy π, where exactly one of i and r_i is listed for each $i \in \Omega$. When r_i occurs in the list, this means that for states which include i and no nodes of higher priority than i, the optimal action is is to retire. The reward received at retirement will be R_i.

It is interesting to note that the total number of stationary Markov policies for Problem $(\mathbf{P_1})$ with N nodes is $N^{(2^N)}$, whereas the total number of priority policies is $N!$. Based on the result of Theorem 2.1, we develop an algorithm which is able to compute an optimal index policy for Problem $(\mathbf{P_1})$ with computational complexity of only $O(N^2)$.

2) Description of Centralized Algorithm: As stated in Section II.A, we use the notation $V^{\pi_i} := V^{\pi}(\{i\})$.

Algorithm 1: (A Dijkstra-Type Algorithm for an Index Policy)

Define the sets $\mathcal{A}$ and $\mathcal{X}$ as follows.

Initially: $\mathcal{A}$ contains the nodes of highest retirement reward $R_{\max}$ in arbitrary order (there must be at least one such node). The action taken by the optimal index policy π on any node $i \in \mathcal{A}$ is r_i and the reward received is $R_{\max}$. $\mathcal{X}$ is the unordered complement (w.r.t. Ω) of $\mathcal{A}$.

During the Construction of Optimal Policy π: $\mathcal{A}$ contains a priority list of a set S of nodes, $S \subset \Omega$, together with the action specified by π on each node in S. $\mathcal{X}$ is the unordered complement (w.r.t. Ω) of $\mathcal{A}$.

The algorithm proceeds as follows.

1) For each $i \in \mathcal{X}$, let π_i be an index policy with the same priority list as π for the nodes of $\mathcal{A}$, with i as the next highest priority node after $\mathcal{A}$, and with the priority of the nodes $\mathcal{X} - \{i\}$ arbitrary, but lower than i. Compute $V_i^{\pi_i}$ from

$$V_i^{\pi_i} = \max\left\{\frac{-c_i + \sum_{S\supset\{i\}:\pi_i(S)\neq i} P^i(S\,|\,i)V^{\pi_i}_{\pi_i(S)}}{\sum_{S\supset\{i\}:\pi_i(S)\neq i} P^i(S\,|\,i)}, R_i\right\} \tag{40}$$

2) Choose $i \in \mathcal{X}$ with the highest value of $V_i^{\pi_i}$, with ties broken arbitrarily. Append this node to the list $\mathcal{A}$ as the next priority node, together with the action specified by (40). Remove i from $\mathcal{X}$.
3) If $\mathcal{X}$ is empty, stop. If not, go to step 1).

Remark: In Step 1 the right-hand side of (40) computes the best expected reward for node i, assuming i is the node of next highest index in π. This computation is feasible because π is a priority policy.

We now establish a relation between (40) and the optimality equation for Problem $(\mathbf{P_1})$. Such a relation allows us to prove that Algorithm 1 indeed computes an optimal index policy.

Theorem 2.1 states that there is an optimal index policy for Problem $(\mathbf{P_1})$. Hence, the DP equation for Problem $(\mathbf{P_1})$ can be written $\forall i \in \Omega$

$$V_i^{\pi} = \max\left\{\max_{\tilde{\pi}}\left\{-c_i + \sum_{S\supseteq\{i\}} P^i(S\,|\,i)V^{\tilde{\pi}}_{\tilde{\pi}(S)}\right\}, R_i\right\} \tag{41}$$

where the inner maximum is taken over all index policies $\tilde{\pi}$. In the following lemma, we use the existence of optimal index policy π to put the computation of V_i^{π} into a more convenient form.

Lemma 2.6: Assume π is an optimal index policy for Problem $(\mathbf{P_1})$. Then, $\forall i \in \Omega$, (42), as shown at the bottom of the page, holds, where the inner maximum is taken over all index policies $\tilde{\pi}$.

Proof: The proof of Lemma 2.6 is in [13]. ■

In the following corollary, we show an important property of the update (40) and its relation to (41).

Corollary 2.1: Assume π is an optimal index policy for Problem $(\mathbf{P_1})$. Let π_i be as defined in Algorithm 1, that is, π_i is the same as π for the set of highest priority nodes $\mathcal{A}$ of π, and node i is the node of highest priority in $\mathcal{X}$ according to π_i. Assume i is also the node of highest priority in $\mathcal{X}$ according to π. If $V_i^{\pi_i}$ is computed as in (40), then

$$V_i^{\pi_i} = V_i^{\pi}. \tag{43}$$

Proof: Corollary 2.1 follows from the fact that (40) computes (42) with the policy $\tilde{\pi} = \pi_i$, which is optimal for all nodes of priority i or higher. ■

Algorithm 1 also resembles Klimov's algorithm [9] and has the following feature.

Theorem 2.2: For Problem $(\mathbf{P_1})$, Algorithm 1 produces an optimal index policy.

Proof: We prove the theorem by induction on the number of nodes in the set $\mathcal{A}$ defined in the description of Algorithm 1. Recall that for a Markov decision problem, the optimal policy maximizes the value function for each state.

Let π be an optimal index policy. Suppose Algorithm 1 has run to the point that $|\mathcal{A}| = L$. Let $i \in \mathcal{X}$ be the node with the actual $(L+1)$th highest priority according to π, whether retiring or not. Let $j \in \mathcal{X}, j \neq i$. Let π_j denote the priority policy that has the same priority as π in the first L nodes, gives j the $(L+1)$th node priority, retiring or not as optimal, and arbitrarily gives priority lower than $L+1$ to the remaining nodes.

Then, we claim that

$$V_i^{\pi_i} = V_i^{\pi} \geq V_j^{\pi} \geq V_j^{\pi_j}. \tag{44}$$

$$V_i^{\pi} = \max\left\{\max_{\tilde{\pi}}\left\{\frac{-c_i + \sum_{\mathcal{N}(i)\supseteq S\supset\{i\}:\tilde{\pi}(S)\neq i} P^i(S|i)V^{\tilde{\pi}}_{\tilde{\pi}(S)}}{\sum_{\mathcal{N}(i)\supseteq S\supset\{i\}:\tilde{\pi}(S)\neq i} P^i(S|i)}\right\}, R_i\right\} \tag{42}$$

Equality (44) follows from Corollary 2.1, because i is assumed to be the actual $(L+1)$th priority node of π. The first inequality of (44) follows because by assumption i is higher priority than j in π. The second inequality follows because π is an optimal policy.

Relation (44) implies that any node $j \in \mathcal{X}$ maximizing $V_j^{\pi_j}$ may optimally be made the $(L+1)$th priority node. Note that this j is not necessarily unique. This is the procedure used to find the node of next highest priority in step 2) of Algorithm 1.

This completes the proof of the induction step. Hence, by induction Algorithm 1 produces an optimal index policy. ■

3) Remarks:

1) In typical network models the network layer requests point-to-point communication from the link layer. However, the stochastic local broadcast nature of our link model requires a control which describes alternative courses of action depending on feedback from random transmission events. In light of Theorem 2.1, we can state precisely what this means for Problem $(\mathbf{P_1})$. The instructions for transmission at a node consist of a priority list of neighbors and the transmitting node itself. The node transmits until a node of higher priority successfully receives the packet. Note that there is no *one* route, as the actual route a packet takes between source and destination is sample path dependent.
2) The index policy result of Theorem 2.1 is very general. It says that no matter what the actual values of $(\mathbf{P}, c, R)_i$ are at each node, an index policy is always one possible optimal policy *structure*. There might be errors in knowledge of $(\mathbf{P}, c, R)_i$, and this affects the indexes of the computed optimal index policy. However, the fact that *some* index policy is optimal is always true.
3) We have shown that Algorithm 1 computes an optimal index policy for Problem $(\mathbf{P_1})$. It is important to note that Algorithm 1 uses all of the network parameters, meaning the $(\mathbf{P}, c, R)_i$ values at each node. To run this algorithm, all of this information must be available at the same location. In this sense, Algorithm 1 is a centralized algorithm, like the Dijkstra algorithm used in OSPF [10, Ch. 4]. The number of updates of (40) in Algorithm 1 is in the worst case $O(N^2)$.

4) Distributed Implementation of an Optimal Index Policy: We note an interesting feature of an index policy used for Problem $(\mathbf{P_1})$. The indexes of the network nodes are fixed, and at each transmission a node of highest index which has the packet is chosen to transmit. This leads to the following property.

Property 2.1: In an index policy for Problem $(\mathbf{P_1})$, the only nodes of index higher than the transmitting node which can receive the packet are neighbors of the transmitting node.

Property 2.1 allows for a natural disributed implementation of the index policy, as follows. Imagine there is a token associated with the packet that begins with the packet at the node of origin. The token indicates which node is to transmit next. After a node transmits, it passes the token to a neighbor (that depends on the outcome of the transmission), or keeps the token for retransmission. By Property 2.1, an optimal index policy can be implemented in this way, as the optimal next node is always a neighbor. Note that there is no central control of this token passing mechanism. All decisions are made locally, and involve only neighboring nodes.

It is important to distinguish distributed optimal routing policy implementation from distributed computation of an optimal index policy itself. For policy implementation, it is assumed that the index policy has already been determined. We consider the problem of distributed index policy computation starting in Section IV.

III. Stochastic Routing Problem With Transmission Control

In this section, we extend the model of Problem $(\mathbf{P_1})$ to allow for control of transmission type at each node. That is, at each time step the controller may choose a node for transmission, and also a type of transmission at that node. Transmission type may be used to model various physical layer features, such as multiple transmission power levels, modulation/coding scheme, antenna directionality, and destination addressing. We begin with some notation and definitions.

A. Notation and Definitions

Definition 3.1: W_i refers to the number of transmission types available at node i.

Definition 3.2: We write $\pi(S) = (i,k)$ to mean that when in state S, policy π chooses node i and transmission type k, $i \in \Omega, k \in \{1 \dots W_i\}$. The expression $\pi(S) = (i,*)$ means policy π chooses i at some unspecified transmission type. The notation for retirement $\pi(S) = r_i$ is retained unchanged.

Definition 3.3: When in state S for which $\pi(S) = (i,k)$, a cost $c_{(i,k)}$ is incurred, and the transition probabilities are written $P^{(i,k)}(S'|S)$.

B. Statement of Problem

We consider Problem $(\mathbf{P_1})$, with the following addition. At each time step the central controller chooses a node for transmission, from among the nodes with the packet, and a transmission *type* from among the allowable types for that node. To each node and transmission type is associated a transmission cost and a probability that a given set of nodes receives the packet. We seek a policy which maximizes (2) under the conditions of Problem $(\mathbf{P_1})$ and the above addition.

C. Analysis of Problem $(\mathbf{P_2})$

The time-homogenous Markov nature of Problem $(\mathbf{P_1})$ is not altered by adding in a choice of transmission type. We find that with an appropriate mapping to a new larger space (each transmission type becomes a node), we can apply the result of Problem $(\mathbf{P_1})$ directly. To demonstrate this, we define a new problem, Problem $(\mathbf{P_3})$, show its relation to Problem $(\mathbf{P_2})$, and then show that it is equivalent to Problem $(\mathbf{P_1})$.

1) Notation and Definitions for Problem $(\mathbf{P_3})$*:* We define a new space of nodes Ω_P as follows. As in Definition 3.2, we list possible control choices for Problem $(\mathbf{P_2})$ as (i,k), where

$i \in \Omega$ and $k \in \{1 \ldots W_i\}$. Every node of Ω_P corresponds to exactly one such control choice of Problem $(\mathbf{P_2})$. Defining N_P to be the cardinality of Ω_P, we then have

$$N_P := |\Omega_P| = \sum_{i=1}^{N} W_i \tag{45}$$

where as before $N := |\Omega|$. We write $\mathsf{j} \in \Omega_P$ to refer to a node from Ω_P. We can also refer to the (i,k) pair of a node $\mathsf{j} \in \Omega_P$, which is the associated i and transmission type $k \in \{1 \ldots W_i\}$ in the original space Ω. We say that the group of nodes in Ω_P which corresponds to $i \in \Omega$ is the *family* of i, so that the family of i has W_i members. The family of $S \subseteq \Omega$ is the set of all nodes in Ω_P which are in the family of any one of the nodes of S.

The cost to transmit from a node j with (i,k) pair is defined to be $c_{(i,k)}$, the cost for transmission type k at i in Problem $(\mathbf{P_2})$.

Transmissions in the Ω_P space are based on the corresponding events in the Ω space, as follows. Say a node j with (i,k) pair is chosen for transmission. This incurs a cost $c_{(i,k)}$. On Ω, transmission from i with transmission type k leads to a set of nodes, say $S_1 \subseteq \Omega$, receiving the packet. Correspondingly, on Ω_P by definition a node $\mathsf{j} \in \Omega_P$ with (l,k) pair receives the packet if and only if l receives the packet on $\Omega, l \in S_1$. That is, nodes in Ω_P in the family of l receive the packet precisely when l does. Note that this means that packet reception for nodes of Ω_P in the same family are deterministically coupled, in that they all receive or all do not receive it. This strongly restricts the kinds of transitions that can occur in Ω_P.

Each member of the family of $i \in \Omega$ gets the same reward as i. That is, let R_P be the reward function on Ω_P, an index function on Ω_P. If $\mathsf{j} \in \Omega_P$ is in the family of i, then $R_P(\{\mathsf{j}\}) = R(\{i\})$. Because R_P is an index function, this fully defines R_P on Ω_P.

2) Formulation of Problem $(\mathbf{P_3})$*:* Problem $(\mathbf{P_3})$ is a formulation of Problem $(\mathbf{P_1})$ for the space Ω_P, with the transition probabilities, cost, and reward described above. The starting state is the family of S_o, the initial state for Problem $(\mathbf{P_2})$.

At each time point the controller either chooses a node from Ω_P for transmission, or retires. We seek a policy which maximizes (2) under the conditions of Problem $(\mathbf{P_1})$ defined on Ω_P.

3) Relation of Problem $(\mathbf{P_3})$ *to Problem* $(\mathbf{P_2})$*:* We show the relation of Problem $(\mathbf{P_3})$ to Problem $(\mathbf{P_2})$. There is a one-to-one mapping from the states of Ω-space to the states of Ω_P-space, as follows. Let $S_1 \subseteq \Omega$ be a state of Problem $(\mathbf{P_2})$. The state of Problem $(\mathbf{P_3})$ corresponding to S_1 is the set of all nodes $\mathsf{j} \in \Omega_P$ such that j is in the family of some node $i \in S_1$. There are states of Problem $(\mathbf{P_3})$ which do not correspond to any state in Problem $(\mathbf{P_2})$. To see this, let $\mathsf{j}, \mathsf{k} \in \Omega_P$ both be in the family of $i \in \Omega$. Consider a state $S \subset \Omega_P$ where $\mathsf{j} \in S, \mathsf{k} \notin S$; this state S is not the mapping from any state $S_1 \subseteq \Omega$. The one-to-one mapping from Ω to Ω_P is not in general an *onto* mapping, because $|\Omega_P| > |\Omega|$. Those states $S \subseteq \Omega_P$ that are not the image of any state $S \subseteq \Omega$ under the above-mentioned mapping are not reachable under the state transition mechanism of Problem $(\mathbf{P_3})$, and play no role in the analysis.

A control action for Problem $(\mathbf{P_3})$ has a a corresponding action in Problem $(\mathbf{P_2})$, as follows. A node $\mathsf{j} \in \Omega_P$, with associated (i,k), chosen for transmission in Problem $(\mathbf{P_3})$ corresponds to transmission at node i of type k in Problem $(\mathbf{P_2})$. The cost $c_{(i,k)}$ incurred for this transmission is the same in both problems, and the state reached by the transmission in Ω maps to the state reached in Ω_P. When retirement is chosen for Problem $(\mathbf{P_3})$, this corresponds to retirement for Problem $(\mathbf{P_2})$, and the same reward is received, because corresponding states have the same reward.

Hence, Problem $(\mathbf{P_2})$ and Problem $(\mathbf{P_3})$ are entirely equivalent, in that to each decision policy for Problem $(\mathbf{P_3})$ there is a corresponding decision policy for Problem $(\mathbf{P_2})$ which results in exactly the same behavior, and hence expected reward, for both systems. We can thus solve Problem $(\mathbf{P_2})$ by finding an optimal policy for Problem $(\mathbf{P_3})$. We proceed to solve Problem $(\mathbf{P_3})$.

4) Analysis of Problem $(\mathbf{P_3})$*:* We show that the system of Problem $(\mathbf{P_3})$ satisfies the requirements of Problem $(\mathbf{P_1})$, and hence is a special case of that system. Specifically, we show that the increasing and decoupling properties (cf. Definitions 2.1 and 2.2) are satisfied, and that the reward function is an index function.

The increasing property on Ω_P follows directly from the fact that the increasing property holds on the underlying space Ω, together with the way states of Ω map to Ω_P. That is, transmissions only lead to an increasing state in Ω, leading to an increasing state in Ω_P.

The decoupling property holds for Problem $(\mathbf{P_3})$ because it holds for Problem $(\mathbf{P_2})$. The nodes that receive transmissions in Problem $(\mathbf{P_2})$ are unaffected by what other nodes have the packet.

Note that transmission events in Ω_P can be highly correlated, in that nodes of Ω_P in the same family either all have the packet, or none have it. However, such event correlations are allowed by Model $(\mathbf{M})$.

Finally, R_P satisfies the definition of an index function on Ω_P, because R is an index function on Ω and R_P gives the same reward for the associated states of Ω_P.

Hence, an index policy is optimal for Problem $(\mathbf{P_3})$, and Algorithm 1 can be used to determine such a policy. This policy is also optimal for Problem $(\mathbf{P_2})$. Thus, Algorithm 1 is effectively used to solve Problem $(\mathbf{P_2})$.

Note that in the resulting priority list of an optimal index policy π, nodes from the same family in Ω_P will appear in some relative order. Since nodes in a given family either all have the packet or none have it, it is clear that only one node from each family will ever be used for transmission by the algorithm. Hence, all the nodes that are not of the highest priority within their family can be removed from the priority list for simplicity. In the terms of Problem $(\mathbf{P_2})$, this means that only one transmission type at each node is ever used for transmission.

Algorithm 1 when applied to Problem $(\mathbf{P_2})$ in this manner is $O(N_P^2)$ in complexity, that is, of complexity $O((\sum_{i=1}^{N} W_i)^2)$.

Comment: The results in this section can be used to show that when a choice exists among deterministic transmissions to single neighbor nodes, then Algorithm 1 reverts to the well-known Dikstra's Algorithm [10, Ch. 4]. In this sense, Algorithm 1 is a generalization of Dikstra's algorithm to the case of stochastic local broadcast transmissions.

IV. Notation Used in Distributed Algorithms for Problem $(\mathbf{P_1})$

A. New Notation and Definitions

We summarize the notation and definitions we need for the distributed algorithms in this paper.

A general Markov policy can be written

$$\{\pi_1\pi_2\ldots\pi_t\ldots\} \tag{46}$$

where the subscript of π indexes time. Since Problem $(\mathbf{P_1})$ is a time-homogeneous Markov decision problem, we know there exists an optimal stationary policy of the form

$$\{\pi\pi\ldots\pi\ldots\}. \tag{47}$$

Further, from Theorem 2.1 we know that there exists a stationary optimal policy of the form (47) where π is an index policy.

When describing distributed algorithms to compute an optimal policy for Problem $(\mathbf{P_1})$, we need to consider a more general type of policy.

Definition 4.1: A *local index policy* for node i at t is written π_t^i, and defines a node ordering of $\mathcal{N}(i), i \in \Omega$, with retirement for node i indicated by r_i if desired. A *distributed index policy at* t is written $\mathbf{\Pi}_t = \{\pi_t^1\pi_t^2\ldots\pi_t^N\}$, where π_t^i is a local index policy for node i. A *distributed index policy* is written $\mathbf{\Pi} = \{\mathbf{\Pi}_1\mathbf{\Pi}_2\ldots\mathbf{\Pi}_t\ldots\}$.

A distributed index policy functions by transmitting at the current node, say i, at t, then using the node ordering π_t^i to choose the next node for transmission.

For local index policy π_t^i, we write $j >_{\pi_t^i} k$ when j has higher priority than k under π_t^i, $j, k \in \mathcal{N}(i)$.

Definition 4.2: Consider nodes i and j, local index policies π_t^i and π_t^j, and let $k, l \in \mathcal{N}(i) \cap \mathcal{N}(j)$. If either 1) $k >_{\pi_t^i} l$ and $k >_{\pi_t^j} l$ or 2) $k <_{\pi_t^i} l$ and $k <_{\pi_t^j} l$, then we say π_t^i and π_t^j *match* on k and l.

Definition 4.3: If π_t^i and π_t^j match on k and l $\forall i, j \in \Omega, \forall k, l \in \mathcal{N}(i) \cap \mathcal{N}(j)$, we say distributed index policy $\mathbf{\Pi}_t$ is *uniform* at t. When $\mathbf{\Pi}_t$ is uniform at t, a global order of the nodes is induced. We call the index policy which has this global node ordering the *associated index policy* of $\mathbf{\Pi}_t$.

We write $P^i(S \mid i)$ to indicate the transmission probability as known at node i.

V. Distributed Algorithm for Problem $(\mathbf{P_1})$

We present our first distributed algorithm (Algorithm 2 described below) which computes the optimal solution for Problem $(\mathbf{P_1})$, and has the characteristic that computations at each node use only information directly from neighbor nodes. This property is critical to the distributed implementation of an optimal policy in an *ad-hoc* wireless network. We claim convergence of the algorithm to the global node ordering and value function consistent with the optimal index policy under the following constraints.

1) Each node i keeps a current estimate, denoted by V_i, of its own optimal expected reward value, with initial value $0 \leq V_i^0 \leq R_{\max}$. Each node i also stores the most recently received estimate of each of its neighbors' optimal expected reward values, denoted $V_{i,j}$, where $j \in \mathcal{N}(i)$, with initial value $0 \leq V_{i,j}^0 \leq R_{\max}$.
2) Information transfer among neighboring nodes consists only of the current V_i value of the transmitting node.
3) Each node's V_i information is transmitted asynchronously.
4) A node's V_i update, defined below, is also asynchronous.
5) It is assumed that each node has knowledge of its own $P^i(S|i)$ update structure. For example, node i may estimate $P^i(S|i)$ based on all its communications, both control signals and packets.
6) The energy required to run the algorithm is not included in finding the optimal solution for Problem $(\mathbf{P_1})$.

The algorithm is as follows.

Algorithm 2: An event time n is when one or more of the following two events occurs. Any number of events may occur at an event time.

- Event 1): A node i receives V_j from a neighbor j and stores it as $V_{i,j}^n$, $j \in \mathcal{N}(i)$.
- Event 2): A node i recomputes V_i using the current $V_{i,j}$ values, as shown in (48) at the bottom of the page.

The maximization in (48) is over all local priority policies $\tilde{\pi}$ of i. It is assumed that events 1 and 2 occur infinitely often at each node i. Note that $P^i(S|i) > 0$ and $\tilde{\pi}(S) \neq i$ imply that $S \supset i$, and for conciseness in (48) we write simply $S \subseteq \mathcal{N}(i)$.

We require no *a priori* time ordering on the above events, nor on the nodes where they are occurring. At times when neither of the above events is taking place, the system is in a frozen state, with all system parameters remaining unchanged. An event which occurs at some event time can have no effect on other events at the same time. Hence, we can choose an arbitrary order for all events occurring at a given time without affecting the outcome. In this way, we can talk sensibly about the nth event in the system since the start, and we use this convention hereafter.

The local policy which optimizes (48) for node i at event n is a local index policy at i, $\pi_{t_n}^i$. For convenience we notate this as π_n^i, where the context prevents ambiguity. The distributed index policy after event n will be denoted by $\mathbf{\Pi}_n$.

Let V_i^n be the expected reward for node i just after event n in the previous system, so that V_i^0 is this value at the start of the

$$V_i^n = \max\left\{\max_{\tilde{\pi}}\left\{\frac{-c_i + \sum_{S\subseteq\mathcal{N}(i):\tilde{\pi}(S)\neq i} P^i(S \mid i)V_{i,\tilde{\pi}(S)}^n}{\sum_{S\subseteq\mathcal{N}(i):\tilde{\pi}(S)\neq i} P^i(S \mid i)}\right\}, R_i\right\} \tag{48}$$

algorithm, where the allowed range is $0 \le V_i^0 \le R_{\max}$. Let $V_{i,j}^n$ be the value after event n of the last transmitted V_j^m received at i, where $m < n$ is the event index of this transmission. At the start, the $V_{i,j}^0$ values do not need to match the neighbor's V_j^0 values, we only require that $0 \le V_{i,j}^0 \le R_{\max}$.

The computation of (48) is based on update (40) of Algorithm 1. Given the nature of this update equation, finding the maximum in (48) is easier than it might first appear, and a simple efficient procedure based on Algorithm 1 and of worst case complexity $O(|\mathcal{N}(i)|^2)$ is given in [13].

We proceed with the analysis of Algorithm 2. The following theorem is our main result concerning Algorithm 2, summarizing its convergence properties.

Theorem 5.1: For Algorithm 2 with any initial state s.t. $0 \le V_i^0 \le R_{\max}$ and $0 \le V_{i,j}^0 \le R_{\max}$ for all $i, j \in \Omega$, we have the following.

1)
$$\lim_{n\to\infty} V_i^n = V_i^\pi \qquad \forall i \in \Omega. \tag{49}$$

2) There exists an event $n_p < \infty$ s.t. $\mathbf{\Pi} = \{\mathbf{\Pi}_{n_p}\mathbf{\Pi}_{n_p+1}\mathbf{\Pi}_{n_p+2}\ldots\}$ is an optimal distributed policy.

3) If
$$V_i^\pi \ne V_j^\pi \qquad \forall i \in \Omega, \qquad j \in \mathcal{N}(i). \tag{50}$$

a) $V_i^n = V_i^\pi, \forall n \ge n_p \qquad \forall i \in \Omega$.

b) $\mathbf{\Pi} = \{\mathbf{\Pi}_{n_p}\mathbf{\Pi}_{n_p+1}\mathbf{\Pi}_{n_p+2}\ldots\}$ has an associated index policy that is optimal.

We outline the proof of Theorem 5.1 by stating without proof a series of lemmas. Full details are available in [13]. Our method of proof of Theorem 5.1 is influenced by the proof of the asynchronous distributed Bellman–Ford algorithm in [4]. We begin with a statement of an important property of the node update procedure which is fundamental for the analysis.

Lemma 5.1 (Update Monotonicity): Consider two cases of a node recomputation for i at event n using (48). In case 1, the neighbor values are $\tilde{V}_{i,j}^n$ and the updated node value is $\tilde{V}_i^n$. In case 2 the neighbor values are $\hat{V}_{i,j}^n$ and the updated node value is $\hat{V}_i^n$. Assume $\tilde{V}_{i,j}^n \ge \hat{V}_{i,j}^n, \forall j \in \mathcal{N}(i)$. Then

$$\tilde{V}_i^n \ge \hat{V}_i^n. \tag{51}$$

We say that the *monotonicity property* holds for a node update due to the result of Lemma 5.1.

We now define two random sequences, $\underline{D}_i^n$ and $\overline{D}_i^n$, which we use to provide bounds on node updates. Noting that at the start of Algorithm 2 each node has a V value between 0 and $R_{\max}$, and aiming at a kind of worst case initial state in light of the monotonicity property just demonstrated, we define $\underline{D}_i^n$ and $\overline{D}_i^n$ as follows.

Definition 5.1: We define $\forall i \in \Omega$ such that $R_i \ne R_{\max}$

$$\underline{D}_i^n := \text{Computed value for } i \text{ after event } n \quad \text{when } V_i^0 = V_{i,j}^0 = 0$$
$$\overline{D}_i^n := \text{Computed value for } i \text{ after event } n \quad \text{when } V_i^0 = V_{i,j}^0 = R_{\max}$$

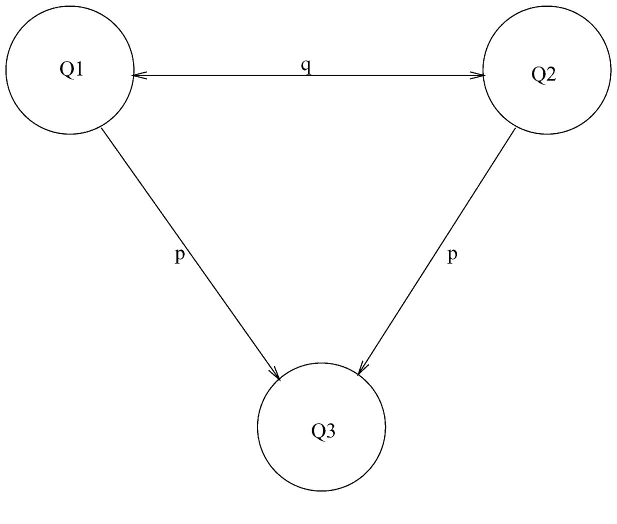

Fig. 1. System which takes infinite time to converge.

For i such that $R_i = R_{\max}$, define

$$\underline{D}_i^n = \overline{D}_i^n := R_{\max} \qquad \forall n.$$

This last definition results from the fact that a node which can retire and receive $R_{\max}$ should always do so.

In the following, we assume π is an optimal policy, and that V_i^π is the optimal value function for i.

The next lemma indicates that $\underline{D}_i^n$ is a monotonically nondecreasing sequence which lower bounds V_i^n, and in turn is upper bounded by V_i^π. Similarly, $\overline{D}_i^n$ is a monotonically nonincreasing sequence which upper bounds V_i^n, and in turn is lower bounded by V_i^π. Given these facts, it remains to show that $\underline{D}_i^n$ converges to V_i^π from below, and that $\overline{D}_i^n$ converges to V_i^π from above, to obtain covergence of V_i^n to V_i^π.

Lemma 5.2: We have

1) $\underline{D}_i^m \le \underline{D}_i^n \le V_i^n \le \overline{D}_i^n \le \overline{D}_i^m \qquad \forall i, n, m \le n;$

2) $\underline{D}_i^n \le V_i^\pi \le \overline{D}_i^n \qquad \forall i, n.$

Comment: In Lemma 5.2, the first part follows directly from the monotonicity property, and in fact is true for any function that is computed at each node that has this property. The second part further requires the property that when all neighbors are correct, the correct value function gets computed.

Corollary 5.1: We have

1) $\underline{D}_i^n = V_i^\pi \Longrightarrow \underline{D}_i^m = V_i^\pi \qquad \forall m \ge n;$

2) $\overline{D}_i^n = V_i^\pi \Longrightarrow \overline{D}_i^m = V_i^\pi \qquad \forall m \ge n.$

The following lemma demonstrates convergence of $\underline{D}_i^n$ to the optimal value V_i^π in a finite number of steps (we remind the reader that π denotes an optimal policy).

Lemma 5.3: There exists $n_p < \infty$ s.t. $\forall n \ge n_p, \underline{D}_i^n = V_i^\pi, \forall i \in \Omega$.

Unfortunately, finite time convergence does not hold in general for $\overline{D}_i^n$, as the following example demonstrates.

Example: Consider the system of Fig. 1 with parameters $0 < p < 1, 0 < q \le 1, c_1 = c_2 = 1$, and $R_1 = R_2 = 0, R_3 = R$ (Q_3 is the destination node). Assume that $R > (1/p)$, so the system has a nontrivial optimal policy. Transmission success from either Q_1 or Q_2 to the other two nodes is independent, with probabilities p and q respectively. Assume Algorithm 2 begins with $V_i^0 = V_{i,j}^0 = R, \forall i, j$.

As Algorithm 2 runs, the node value computations of Q_1 and Q_2 ping-pong back and forth, as the update of one is transmitted to be used in the other's update. Letting V_1^n denote the update value at Q_1 at the nth such update, it can be shown that for $n \geq 1$

$$V_1^n = \left(R - \frac{B}{1-A}\right) A^n + \frac{B}{1-A} \tag{52}$$

where $0 < A < 1$ and $(B)/(1-A) = R - (1/p)$. A similar equation holds for V_2^n. Hence $\lim_{n\to\infty} V_1^n = R - (1/p)$, which is the correct value function. But because $A > 0$, this value is never reached in finite time. □

In what follows, we discuss the sense in which an optimal distributed policy *is* reached in finite time in the above example.

Comment: The key fact in the preceding example is that nodes 1 and 2 have identical value functions. As a result, the V_i^n values of these two nodes converge to the same value, and hence interact in each computation at each event n, preventing convergence in finite time. When limit values are different at each node, a result similar to Lemma 5.3 for the $\overline{D}_i^n$ sequence follows easily, as shown in the following lemma.

Lemma 5.4: For Algorithm 2

1) $\forall i$, there exists $W_i \in \mathbb{R}$ s.t. $\overline{D}_i^n \downarrow W_i$;
2) Assume $W_i \neq W_j, \forall i, j$ with $j \neq i$. Then $\exists n_p < \infty$ s.t. $\forall n \geq n_p, \overline{D}_i^n = V_i^\pi, \forall i$.

We are in fact able to prove asymptotic convergence for the $\overline{D}_i^n$ sequence in general, and we outline our approach here (details are in [13]). We define a new system, which we refer to as the *round-robin* (r-r) system, where node update and transmission events follow a fixed predefined order. Using the monotonicity property (Lemma 5.1), we are able to prove that $\overline{D}_i^n$ is bounded above by a corresponding value in the r-r system and then show that they both must converge to V_i^π. The proof strategy is to define a mapping for the r-r system $T : \mathbb{R}^N \to \mathbb{R}^N$ in the form $\mathbf{W}^{n+1} = T(\mathbf{W}^n)$ and demonstrate that the mapping T is component-wise continuous for each output, that $\mathbf{V}^\pi = T(\mathbf{V}^\pi)$ and there are no other fixed points of T, and hence that the r-r system converges to the optimal fixed point. Asymptotic convergence of $\overline{D}_i^n$ follows.

We give an example where Algorithm 2 does not converge to an optimal index policy, but still converges to an optimal distributed policy.

Example: Consider the system of Fig. 2 with parameters $0 < p < 1, 0 < q \leq 1, c_1 = c_2 = c_4 = 1$, and $R_1 = R_2 = R_4 = 0, R_3 = R$. Assume node recomputations and successful value transmissions occur in the order $[1, 4, 2, 4, 1, 4, 2, 4, \ldots]$. Then node 1 and 2 updates are still represented by (52). The policy at node 1 is fixed at $\pi_t^1 = (3, 2, 1), \forall t$, and the policy at node 2 is fixed at $\pi_t^2 = (3, 1, 2), \forall t$. However, at each update of node 4, the policy changes, as the ranking of the values of nodes 1 and 2 alternate. That is, the policy computed at node 4 is

$$\pi_t^4 = \{(1, 2, 4)(2, 1, 4)(1, 2, 4)(2, 1, 4)(1, 2, 4)\ldots\}. \tag{53}$$

Of course, $\pi_t^3 = r_3$. The overall policy is $\mathbf{\Pi}_t = (\pi_t^1 \pi_t^2 \pi_t^3 \pi_t^4)$, which is not a stationary policy, due to π_t^4. It is, however, an optimal distributed policy.

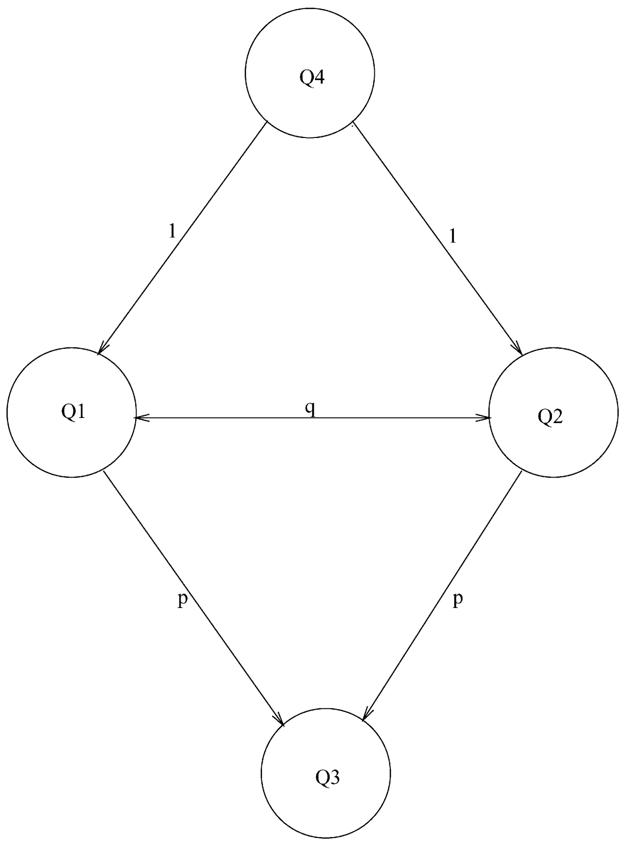

Fig. 2. System with optimal distributed policy.

VI. Distributed Dynamic Programming Formulation

We develop a distributed algorithm, different from but related to Algorithm 2, using the methodology of DDP [3]. DDP is a technique for solving dynamic programming problems using distributed computation. The technique may only be used for problems formulated so that a standard dynamic programming equation applies, and for which a suitable partitioning of the state space among processors ("computation centers") can be made. See [3] for a description of DDP, and [12] for further details on applying DDP to Problem $(\mathbf{P_1})$.

A. Solution of Problem $(\mathbf{P_1})$ *Using DDP*

The model of Problem $(\mathbf{P_1})$ is a standard controlled Markov chain with finite state space 2^Ω and action space $i \in \Omega$. Dynamic programming can be directly applied to Problem $(\mathbf{P_1})$ on state space 2^Ω, but this approach is inefficient and does not lead to a direct application of DDP. This is because it is not possible to define useful computation centers as in [3] through partition of $S = 2^\Omega$ (on this state space, to define computation centers in a way that leads to the application of DDP, we would need to know *a priori* the optimal priority list of nodes as dictated by Theorem 2.1). Our approach is to use the index structure of an optimal policy demonstrated in Theorem 2.1 to define a new state space on which DDP can be applied.

To solve Problem $(\mathbf{P_1})$ using DDP, we proceed in three steps.

1) Formulate a new problem, Problem $(\mathbf{F_1})$.

2) Show that an optimal policy for Problem $(\mathbf{F_1})$ can be mapped to an optimal distributed policy for Problem $(\mathbf{P_1})$.
3) Apply DDP to Problem $(\mathbf{F_1})$.

Here, we present the details of each of the aforementioned steps.

1) Formulation of Problem $(\mathbf{F_1})$*:* Problem $(\mathbf{F_1})$ The state space is Ω, the set of nodes. The policy space is the set of all local index policies π^i for all $i \in \Omega$. When transmitting in state i, a cost c_i is incurred and transition to a new state $j \in \Omega$ occurs. Define $P_{ij}^{\pi^i}$ to be the probability of transition from state i to state j, $j \in \mathcal{N}(i)$, under policy π^i. Then

$$P_{ij}^{\pi^i} = \sum_{S:\pi^i(S)=j,r_j} P^i(S|i). \tag{54}$$

When we retire in state i, the process terminates and a reward R_i is received. The objective is to choose for each state i the local index policy π^i to maximize

$$E\left\{R_{i(\tau)} - \sum_{t=1}^{\tau-1} c_{i(t)}\right\} \tag{55}$$

where τ is the time when the transmission process is terminated, and $i(t)$ is the state at time t.

2) Mapping of Optimal Policy From Problem $(\mathbf{F_1})$ *to Problem* $(\mathbf{P_1})$*:* We define a mapping of policies for Problem $(\mathbf{F_1})$ to policies for Problem $(\mathbf{P_1})$.

Mapping 1: A general (possibly time-varying) policy for Problem $(\mathbf{F_1})$ consists of a sequence of local index policies for each $i \in \Omega$. For each t, we define

$$\mathbf{\Pi}_t = \left\{\pi_t^1 \pi_t^2 \dots \pi_t^N\right\}. \tag{56}$$

A policy for Problem $(\mathbf{F_1})$ is then specified as

$$\mathbf{\Pi} = \{\mathbf{\Pi}_1 \mathbf{\Pi}_2 \dots \mathbf{\Pi}_t \dots\}. \tag{57}$$

We map a policy for Problem $(\mathbf{F_1})$ represented as (57) into a distributed index policy for Problem $(\mathbf{P_1})$ (cf. Definition 4.1) in the obvious way. That is, at each time t the local index policy π_t^i at each node $i \in \Omega$ is the same for Problem $(\mathbf{P_1})$ as for Problem $(\mathbf{F_1})$.

We show that mapping an optimal policy of Problem $(\mathbf{F_1})$ by Mapping 1 leads to an optimal policy for Problem $(\mathbf{P_1})$.

Lemma 6.1: If $\mathbf{\Pi}$ is an optimal policy for Problem $(\mathbf{F_1})$, then $\mathbf{\Pi}$ mapped to a policy for Problem $(\mathbf{P_1})$ using Mapping 1 is an optimal distributed policy for Problem $(\mathbf{P_1})$.

Proof: The assertion of Lemma 6.1 follows from Mapping 1 and the fact that an optimal policy for Problem $(\mathbf{P_1})$ is of the index type. ■

3) A Distributed Dynamic Programming Implementation for Problem $(\mathbf{F_1})$*:* We formulate a DDP solution of Problem $(\mathbf{F_1})$ by translating the notation of [3] into our notation. Each node i in our model is a computation center for itself alone, so that $S_i = i, \forall i \in \Omega$. We associate

$$\begin{aligned} S &= \Omega \\ C &= \{\text{Set of all priority orderings of } \mathcal{N}(i), \forall i \in \Omega\} \\ U(i) &= \{\text{Set of all priority orderings of } \mathcal{N}(i)\} \\ \bar{F} &= (-\infty, R_{\max}]. \end{aligned}$$

We also note the notational correspondences $V \leftrightarrow J$, and $V_i^n \leftrightarrow J_i^t$, which are equivalent assuming event n occurs at time t.

A neighbor of a node in the sense of [3] corresponds to our notion of neighbor. However, the notation for the neighbors of i used in [3], $N(i)$, does not include i, whereas for our notation $i \in \mathcal{N}(i)$.

For Problem $(\mathbf{F_1})$ we define the H function of [3] $\forall i \in \Omega, \pi^i \in U(i)$ as

$$H(i,\pi^i,V) = -\max\left\{-c_i + \sum_{S\subseteq\mathcal{N}(i)} P(S|i)V(\pi^i(S)), R_i\right\}. \tag{58}$$

The negative sign on the right-hand side of (58) is used to conform to the convention of [3] that the goal is to minimize cost. Thus, the dynamic programming equation for Problem $(\mathbf{F_1})$ is

$$V_i^\pi = \min_{\pi^i} H(i,\pi^i,V^\pi). \tag{59}$$

Note that H is monotone in V in the sense of [3].

The update for state $i \in \Omega$ by DDP at event n is

$$V_i^n = \min_{\pi^i} H(i,\pi^i,V^n). \tag{60}$$

We suppose that Assumption 1 of [3] is true for node updates and transmissions. The functions

$$\underline{V}_i := -R_{\max} \qquad \forall i \in \Omega \tag{61}$$

$$\bar{V}_i := 0 \tag{62}$$

satisfy part i) of *Assumption 2* of [3] due to (58) and (60), and part ii) of *Assumption 2* due to (59) and the standard result on value iteration for a dynamic program. We require that, when running DDP for Problem $(\mathbf{F_1})$, we start with a value between $\underline{V}$ and $\bar{V}$ at each node.

This completes the DDP formulation of Problem $(\mathbf{F_1})$. We now briefly state the results from [3] which apply to Problem $(\mathbf{F_1})$. Assumptions 1 and 2, and the requirements of [3, Prop. 1] are satisfied. Reference [3, Prop. 1], translating into our notation, then implies that

$$\lim_{n\to\infty} V_i^n = V_i^\pi \qquad \forall i \in \Omega. \tag{63}$$

Because for fixed π^i update (58) is continuous in the components of V, the requirements of Proposition 3 of [3] are also satisfied. Translating to our notation, we obtain from Proposition 3 that there exists a $\bar{n}_i > 0$ such that for all $n \geq \bar{n}_i$, if a local index policy π_n^i satisfies

$$H\left(i,\pi_n^i,V_i^n\right) = \min_{\pi^i} H\left(i,\pi^i,V_i^n\right) \qquad \forall i \in \Omega \tag{64}$$

then

$$H\left(i,\pi_n^i,V^\pi\right) = \min_{\pi^i} H(i,\pi^i,V^\pi). \tag{65}$$

Equation (65) states that π_n^i takes an optimal action for state i at event n.

Thus, we have shown how Problem $(\mathbf{F_1})$ can be solved with the DDP methodology of [3] using update (60).

B. Relation Between DDP Solution to Problem $(\mathbf{F_1})$ and Algorithm 2

Event 1 of Algorithm 2 corresponds precisely to the transmit state, which is update 1) of [3]. Event 2 of Algorithm 2 is related to update 2) of [3]. However, the update (48) for Algorithm 2 and (60) for DDP are not the same. For an update at node $i \in \Omega$ at event $n+1$, the key difference is that in (60) the value V_i^n affects the updated value V_i^{n+1}, whereas in (48) it does not. Another difference, relatively minor, between the DDP formulation and Algorithm 2 is that where Algorithm 2 assumes Events 1 and 2 occur infinitely often, the DDP formulation of Problem $(\mathbf{F_1})$ requires the somewhat more restrictive *Assumption 1*.

Though Algorithm 2 and the DDP formulation differ in the ways just mentioned, the results proved for each algorithm are quite similar. As remarked in (63), [3, Prop. 1] implies that

$$\lim_{n\to\infty} V_i^n = V_i^\pi \qquad \forall i \in \Omega. \tag{66}$$

Equation (66) shows asymptotic convergence to the value function for Problem $(\mathbf{F_1})$ and, hence, for Problem $(\mathbf{P_1})$, and is similar to result Theorem 5.1, (1.). The result of Proposition 3 (65) is similar to our Theorem 5.1, (2.). No results similar to Theorem 5.1, (3.) for Problem $(\mathbf{P_1})$ under the DDP formulation follow directly from a result of [3].

As these results show, once Theorem 2.1 is proved for Problem $(\mathbf{P_1})$, application of Distributed Dynamic Programming on an appropriate state space provides a new distributed algorithm for Problem $(\mathbf{P_1})$ with certain properties nearly equivalent to those of Algorithm 2.

VII. Distributed Rank Method

We define a third distributed algorithm for Problem $(\mathbf{P_1})$, which is similar in spirit to Algorithm 2, except that a different method is used to update the node value function. We define the following.

Algorithm 3: At each event time, any number of the following two events can occur.

- **Event 1):** A node i receives V_j from a neighbor $j, j \in \mathcal{N}(i)$.
- **Event 2):** A node i recomputes V_i using the current $V_{i,j}$ values, as follows.

1) The set of $V_{i,j}$ and V_i values are ranked, high-to-low. A local index policy $\tilde{\pi}$ for i is created which uses this ranking (ties are broken arbitrarily). If $V_i \geq V_{i,j}\ \forall j \in \mathcal{N}(i)$, then $\tilde{\pi}$ uses *any* ranking of the neighbors, so long as i is given a ranking below at least one neighbor.
2) The following is computed.

$$V_i^n = \max\left\{\frac{-c_i + \sum_{S\subseteq\mathcal{N}(i):\tilde{\pi}(S)\neq i} P^i(S\,|\,i)V_{i,\tilde{\pi}(S)}^n}{\sum_{S\subseteq\mathcal{N}(i):\tilde{\pi}(S)\neq i} P^i(S\,|\,i)}, R_i\right\} \tag{67}$$

We assume events 1 and 2 occur infinitely often at each node i. We state our main results for Algorithm 3 in the following theorem.

Theorem 7.1: For Algorithm 3 with any initial state s.t. $0 \leq V_i^0 \leq R_{\max}$ and $0 \leq V_{i,j}^0 \leq R_{\max}$ for all $i, j \in \Omega$, we have

1) $\lim_{n\to\infty} V_i^n = V_i^\pi, \forall i \in \Omega$;
2) there exists an event $n_p < \infty$ s.t. $\mathbf{\Pi} = \{\mathbf{\Pi}_{n_p}\mathbf{\Pi}_{n_p+1}\mathbf{\Pi}_{n_p+2}\ldots\}$ is an optimal distributed policy;
3) if

$$V_i^\pi \neq V_j^\pi \qquad \forall i \in \Omega, \qquad j \in \mathcal{N}(i) \tag{68}$$

then there exists an event $n_p < \infty$ s.t.

a) $V_i^n = V_i^\pi, \forall n \geq n_p \qquad \forall i \in \Omega$;
b) $\mathbf{\Pi} = \{\mathbf{\Pi}_{n_p}\mathbf{\Pi}_{n_p+1}\mathbf{\Pi}_{n_p+2}\ldots\}$ has an associated index policy that is optimal.

We state without proof the following Lemma 7.1 which we use to prove Theorem 7.1 (detailed proofs are in [13]). First we note that the monotonicity property in the sense of Lemma 5.1 does not hold in general for the update of Algorithm 3 (a concrete counterexample is given in [13]), and hence we must take a different approach to proving convergence than what we used for Algorithm 2. Nevertheless, we are able to prove the following lemma.

Lemma 7.1: For Algorithm 3 we have $\lim_{n\to\infty} V_i^n = V_i^\pi, \forall i \in \Omega$

We are now ready to present the Proof of Theorem 7.1.

Proof: **[of Theorem 7.1]**

Proof of 1: Relation 1 follows directly from Lemma 7.1.

Proof of 2: Because of 1. there exists n_p after which the node values at each node i and its $\mathcal{N}(i)$ are in the order of some local index policy with an optimal associated index policy. At each event $n \geq n_p$ this local ordering can change, but it always corresponds to some optimal associated index policy. Hence, at each node i an optimal action is taken at each event $n \geq n_p$. Hence, the distributed policy is optimal.

Proof of 3: Because of 1. and Lemma 7.1, there exists n_p after which node values for each node i and its $\mathcal{N}(i)$ are in the order of the unique optimal local index policy and there are no subsequent changes in this order. Subsequent to n_p, once a full round of node updates occurs, each node has its correct V_i^π and its correct local index policy, which subsequently do not change. ■

Discussion: It is interesting to note the relation of Algorithm 3 to the results in [7]. Major differences include the on-going nature of the node updates in Algorithm 3, and that node values are interpreted as the expected reward at a given node, whose estimates are constantly being updated by (67). The direction flipping action of [7] corresponds to resetting in Algorithm 3. The Proof of Theorem 7.1 is more difficult than the analogous result in [7], due to the fact that in Algorithm 3 node values can change not just when being reset but also when normally computing updated estimates.

VIII. Algorithm Comparison and Simulation Results

For ease of reference, we summarize the update functions of this paper's three algorithms in Table I. All three algorithms run in a similar fashion, with the main difference being this update function. In terms of computational complexity of the update, Alg. 2 and the DDP algorithm are roughly comparable, while the Rank algorithm is somewhat less complex. In Theorem 5.1, the DDP results, and Theorem 7.1 we have shown finite-time convergence of each algorithm to an optimal policy in a static network. There remain important questions about dynamic performance of each algorithm, such as convergence rate, which have not been addressed in this paper.

TABLE I
UPDATE FUNCTION FOR THREE DISTRIBUTED ALGORITHMS

Alg.	V_i^n *Update*
Alg. 2	$\max\left\{\max_\pi\left\{\frac{-c_i+\sum_{S\subseteq\mathcal{N}(i):\pi(S)\neq i} P^i(S\vert i)V^{n-1}_{i,\pi(S)}}{\sum_{S\subseteq\mathcal{N}(i):\pi(S)\neq i} P^i(S\vert i)}\right\}, R_i\right\}$
DDP	$\max\left\{\max_\pi\left\{-c_i+\sum_{S\subseteq\mathcal{N}(i)} P^i(S\vert i)V^{n-1}_{i,\pi(S)}\right\}, R_i\right\}$
Alg. 3 (Rank)	π set to rank order of $V^{n-1}_{i,j}$ $\max\left\{\frac{-c_i+\sum_{S\subseteq\mathcal{N}(i):\pi(S)\neq i} P^i(S\vert i)V^{n-1}_{i,\pi(S)}}{\sum_{S\subseteq\mathcal{N}(i):\pi(S)\neq i} P^i(S\vert i)},\ R_i\right\}$

An *ad-hoc* network simulation has been conducted to compare performance of the three distributed algorithms of Table I, along with the performance of other algorithms in the literature. The simulation is setup so that the frequency of packet arrival justifies a route-maintenance approach. The density of mobiles in the simulated network is chosen so that the local broadcast nature of transmission is beneficial. The results presented here are not conclusive in comparing the different approaches, but they do represent an important class of networks for which our approach performs very well.

The network consists of 12 mobiles on a field 2 km $\times$ 2 km. Each mobile moves at the same constant speed, which can be varied over different simulation runs. Each mobile chooses a waypoint which is a random point in the field of operation, moves toward this waypoint at a fixed speed, and then chooses a new random waypoint once the previous one is reached. Messages arrive at a fixed rate at each mobile, and they all have the same destination node, which is also moving. Mobiles transmit each packet at fixed power, and take statistics on which other mobiles receive the packet. Mobiles are assumed to be transmitting pilot signals from which the active set of neighbors can be determined for both control signaling and decoding attempts. Transmission instances are assumed synchronous across all mobiles. Each mobile has a single omnidirectional antenna for both transmitting and receiving.

Transmission channels and interference are modeled using the method from the 3GPP2 1xEV-DO Evaluation Methodology [1] for CDMA transmission. In the present simulation, only Channel A (a 3 kmph single-path Rayleigh fading channel) is used, and short-term curves (PER versus Eb/No) generated for this channel are used to determine packet success. When variation in velocity is indicated in the simulation results, this indicates the rate of network topology change, not of the fading process. Signal propagation loss is determined using a power of 3.5 dropoff in distance. Path loss to each neighboring mobile consists of a propagation loss, time-varying shadowing, and short-term fading. The propagation loss is based on distance, the time-varying shadowing is assumed 0.5 correlated across neighbors, and the short-term fading is assumed independent across neighbors. Each time instant, the power received at each node is determined and an interference term is used in the Eb/No computation for each mobile packet transmission. We model a carrier frequency of 1.9 GHz, a chip rate of 1.25 Mcps, and a spreading gain of 20, which implies an approximate 60 kbps transmission rate. We assume each packet is 50 ms in duration.

Each transmission is assumed to cost a fixed energy of 1 unit and occur at a fixed power. After each transmission from a mobile, the routing protocol uses the set of neighbor mobiles successfully receiving the packet to determine whether to retransmit the packet or to pass control of the packet to a neighbor mobile. The energy cost of the packet reception and control signaling is assumed negligible in these simulations.

As the mobiles move toward their waypoints, and as shadowing and fading vary with time, transmission success probability to each neighbor dynamically changes. Transmission probabilities are estimated for all algorithms using a moving window average of recent success frequency. These successes are kept individually for each neighbor and assumed independent. Though this is not exactly true physically (nor in the simulation transmission model itself), the effect of fading over packet transmission times often means this model is a reasonable approximation and useful for making routing decisions. Optimality of these algorithms does not depend on such independence, we assume it solely for computational simplicity.

We simulate the performance of six algorithms. The first three are those presented in this paper and summarized in Table I. The remaining three are versions of well-known algorithms: Distributed Bellman–Ford (DBF) [4] with a hop-count metric (DBF-HC), DBF with an expected energy cost metric (DBF-EN), and the method of Gafni–Bertsekas (GB) [7], which has also been adapted for use in the TORA algorithm [18]. In DBF-HC, the path with fewest links to the destination is used. In DBF-EN, a metric for each link is determined based on the expected energy to transmit across the link, which in our model is the inverse of the success probability, and the path of smallest cumulative expected energy is used. In GB, the same path is used until a link in the path is broken, at which time a distributed algorithm runs which is guaranteed to determine a new path to the destination. In the DBF-HC and GB algorithms, a strict binary decision is made as to whether or not each link is connected. In our simulation, we use a fixed probability threshold to make this determination.

We simulate two cases, that of a *loaded* network, and that of a *partially loaded* network. In the case of a loaded network, it is assumed that each mobile is transmitting and generating interference nearly all of the time, even when the mobile does not currently have any packet destined for the destination being simulated. This accounts for the fact that though we simulate packets for just one destination, in the actual network there may be many such destinations and their packets are constantly flowing in the network. The loaded network is the case where transmission opportunities are close to saturation. In the partially loaded network, we assume that each mobile transmits only a fraction of the time, with these times chosen randomly when generating neighbor interference.

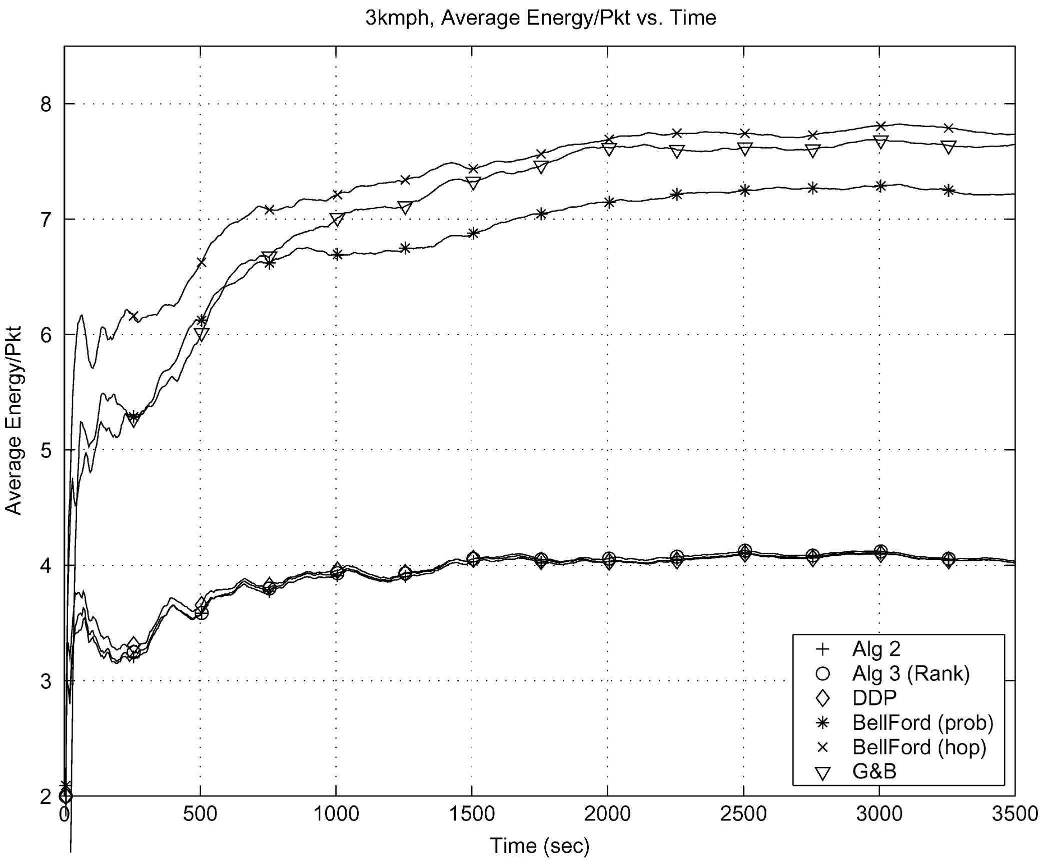

Fig. 3. Convergence of average energy per packet for six distributed routing algorithms, 3 kmph, loaded network.

We first show the nature of convergence of the average energy expended per packet (EPP) to reach its destination. Because each packet transmission requires one unit of energy, average EPP also indicates the number of times on average a transmission attempt was made per packet before it finally reaches the destination. A running total of this average is recorded in the simulation each time step. The average EPP versus time value is plotted in Fig. 3 for the 3 kmph case for all six algorithms for a loaded network. The overhead signaling for each of these algorithms was not accounted for (e.g., for GB upon occasion the entire path must be updated with extensive control signaling, but we ignore the cost of that here). Because the energy cost of a packet is only recorded once it reaches the destination, early measurements of EPP are biased toward good packets, and hence are optimistic. As the simulation progresses, the average settles to a steady-state value indicating the correct overall EPP.

The three algorithms using stochastic routing outperform the three algorithms that do not by nearly 3 dB. We note in passing that this is also the order of improvement claimed for soft handoff in cellular systems [24]. Interestingly, the Alg2, DDP, and Rank algorithms all converge to the same value. It is important to recognize that, though we have proved that these three algorithms converge to the optimal routing algorithm in the steady-state, their dynamic performance is potentially different in this more realistic nonstationary context with accurate channel models. As expected, DBF-EN moderately outperforms DBF-HC, while GB converges to a value slightly better than DBF-HC. It is difficult to predict in general how DBF-HC and GB will compare to each other. There is no guarantee that the hop-count metric is actually better than the path in use by GB, because, though the GB path is chosen in a somewhat arbitrary manner, the actual cost of the hop-count path may also vary widely based on the link channel quality. The DBF-EN algorithm takes the link quality into account, and so should be the best of the algorithms which update a fixed path.

Fig. 4 shows average EPP for the six algorithms as a function of mobile velocity, which is used to update network topology. Note that in general the performance of each algorithm suffers as the velocity increases. The basic effect is that the estimate of transmission probability from past events becomes less relevant to future transmission success as the topology changes more and more rapidly. For the three stochastic routing algorithms and for DBF-EN, the inaccuracy in probability estimation translates directly into routing inefficiency. For DBF-HC and GB, which use a channel quality threshold, the effect is to postpone when link state change detection occurs, and the quality of the path in use suffers accordingly. Note also that Alg 2 very slightly outperforms DDP and Rank for thise case at higher velocity.

Fig. 5 also shows average EPP for the six algorithms as a function of mobile velocity, but for the case of a partially loaded network. Note that the average EPP is about half that of Fig. 4, but the ratio of each algorithms performance is roughly comparable. This indicates that the advantages of stochastic routing extend to the case of multiple loading levels of a network. One question to pursue about this plot is why the DBF-HC, DBF-EN, and GB algorithms actually improve at the highest velocity.

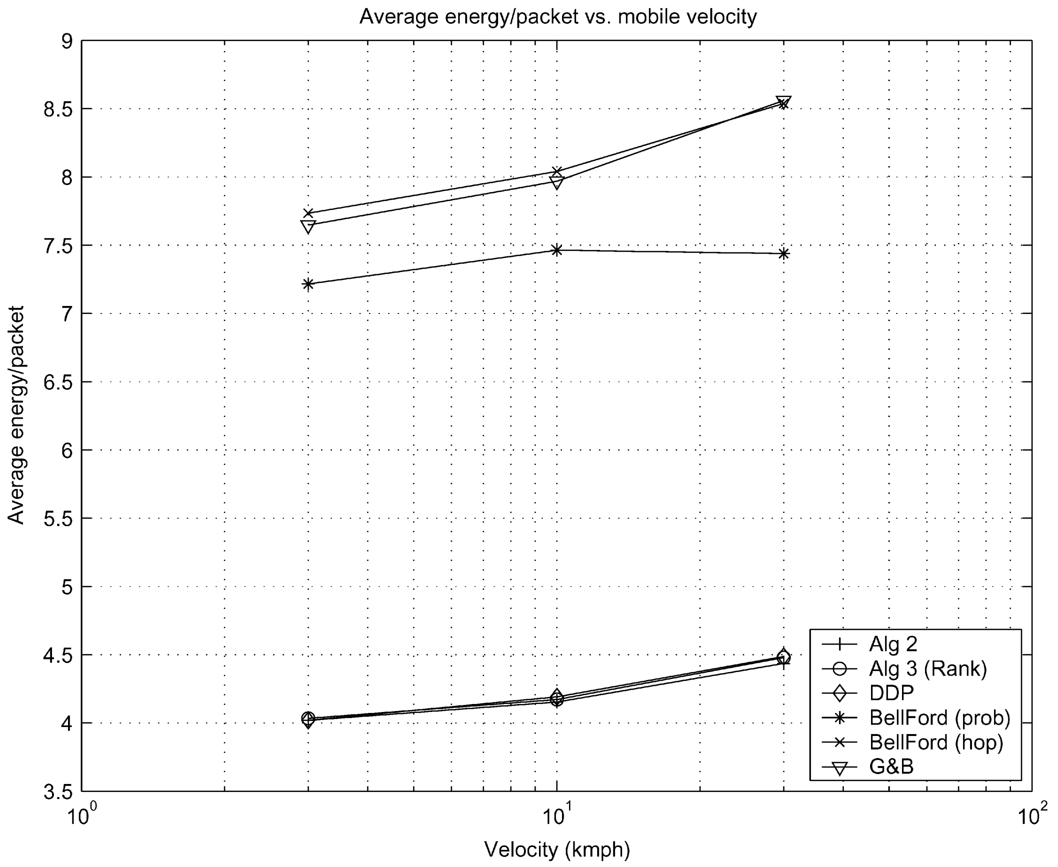

Fig. 4. Average energy per packet versus velocity for six distributed routing algorithms, loaded network.

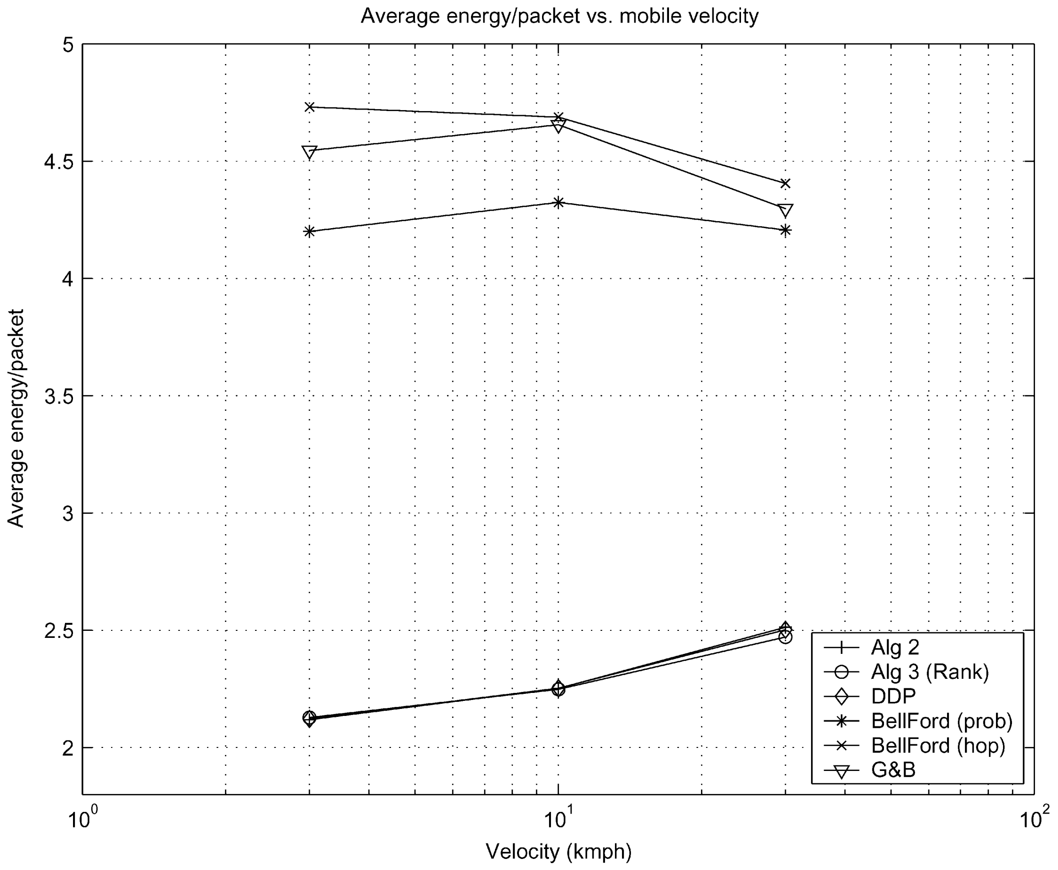

Fig. 5. Average energy per packet for six distributed routing algorithms, partially loaded network.

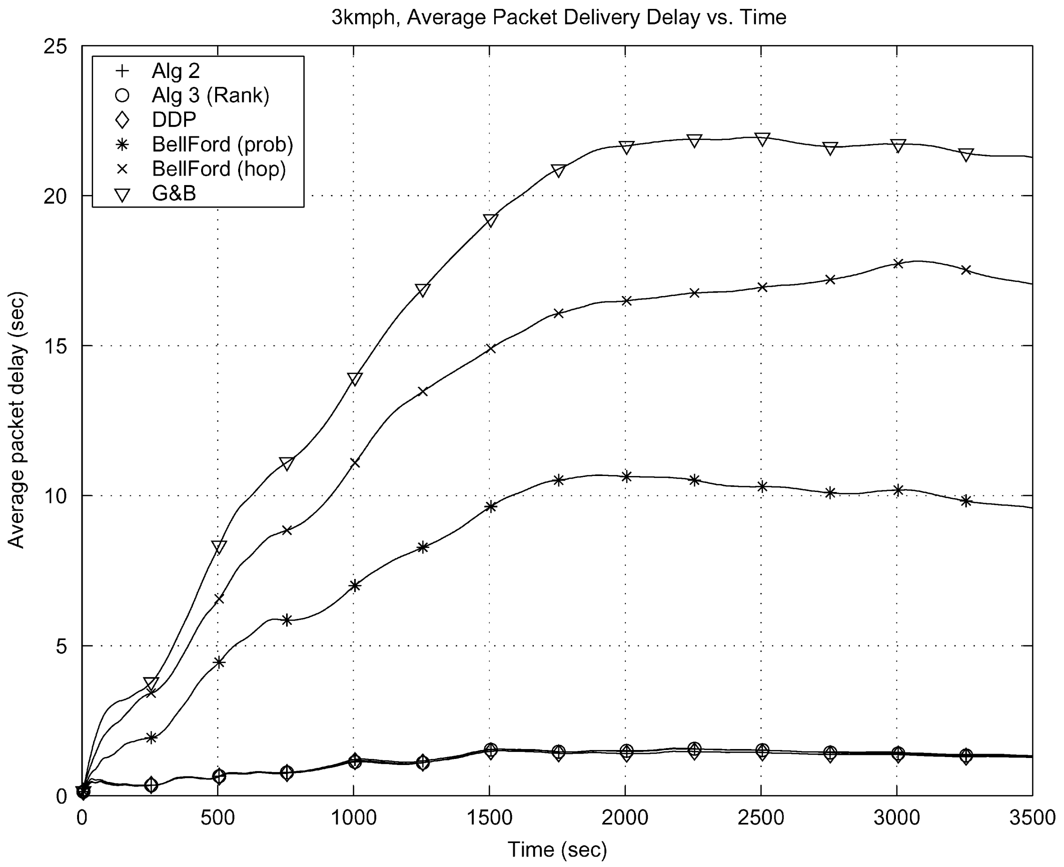

Fig. 6. Convergence of delivery delay per packet for six distributed routing algorithms, 3 kmph, loaded network.

Along these lines note that though higher velocity has the effect of impeding probability estimation, it can also be beneficial in keeping the network sufficiently mixing to keep fixed paths from clogging up.

Finally, we consider delivery delay per packet. Delivery delay is different from EPP because packets can also incur queuing delay at each mobile. This queuing delay is generally worse for the fixed-path algorithms, because packets locally tend to take the same path. Note that if load-balancing were implemented to improve delay, then EPP would generally increase, at least for DBF-EN. In contrast, the stochastic routing approach naturally leads to more spreading out of packet transmissions among mobiles, greatly reducing the need for an explicit load balancing algorithm.

Fig. 6 shows the time trace of average delivery delay per packet for each algorithm in the loaded network for the case of 3 kmph. The stochastic routing delay performance is substantially better than that of the other algorithms. Also, DBF-EN outperforms DBF-HC and GB more significantly in delay. This is because the expected energy metric leads to more variation in the path used, which leads to less congestion. Though none of these algorithms has been optimized for delay, the delay statistic is still of interest in understanding algorithm performance. It is useful when an algorithm achieves both better energy and delay performance, as stochastic routing does in these examples.

References

[1] *1xEV-DO Evaluation Methodology (v1.4)*, 2003.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[3] D. Bertsekas, "Distributed dynamic programming," *IEEE Trans. Autom. Control*, vol. AC-27, no. 3, pp. 610–616, Jun. 1982.

[4] D. Bertsekas and R. Gallager, *Data Networks*. Upper Saddle River, NJ: Prentice-Hall, 1992.

[5] J. Broch, D. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Proc. 4th Annu. ACM/IEEE Int. Conf. Mobile Computing and Networking*, Oct. 25–30, 1998.

[6] M. Fan *et al.*, "On the reverse link performance of cdma2000 1xEV-DO revision a system," in *Proc. ICC 2005*, Seoul, South Korea.

[7] E. Gafni and D. Bertsekas, "Distributed algorithms for generating loop-free routes in networks with frequently changing topology," *IEEE Trans. Commun.*, vol. COM-29, no. 1, pp. 11–18, Jan. 1981.

[8] Z. Haas, "Guest editorial, wireless ad hoc networks," *IEEE J. Select. Areas Commun.*, vol. 17, no. 8, pp. 1329–1330, Aug. 1999.

[9] G. Klimov, "Time sharing service systems I," *Theory Probab. Appl.*, vol. 19, pp. 532–551, 1974.

[10] J. Kurose and K. Ross, *Computer Networking*. Reading, MA: Addison-Wesley, 2000.

[11] C. Lott *et al.*, *Reverse Traffic Channel Design of cdma2000 1xEV-DO Revision A System*. Stockholm, Sweden: VTC, Jun. 2005.

[12] C. Lott and D. Teneketzis, "Stochastic routing in ad hoc wireless networks," Univ. Michigan Control Group, Rep. CGR 01-01, Feb. 2001.

[13] ——, (2005, May) Stochastic routing in ad hoc networks. Communications and Signal Processing Lab., Dept. EECS, Univ. Michigan, Ann Arbor, MI. [Online]. Available: www.eecs.umich.edu/systems/TechReportsList.html, Rep. TR-362

[14] G. S. Malkin, "RIP version 2: Carrying additional information," in *RFC 1388*, Jan. 1993.

[15] D. Maltz, J. Broch, J. Jetcheva, and D. Johnson, "The effects of on-demand behavior in routing protocols for multihop wireless ad hoc networks," *IEEE J. Select. Areas Commun.*, vol. 17, no. 8, pp. 1439–1453, Aug. 1999.

[16] P. Merlin and A. Segall, "A failsafe distributed routing protocol," *IEEE Trans. Commun.*, vol. 27, no. 9, pp. 1280–1287, Sep. 1979.

[17] J. Moy, "OSPF Version 2," *RFC 1247*, July 1991.

[18] V. Park and M. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proc. INFOCOM'97*, Apr. 1997, pp. 1405–1413.

[19] M. Pearlman and Z. Haas, "Determining the optimal configuration for the zone routing protocol," *IEEE J. Select. Areas Commun.*, vol. 17, no. 8, pp. 1395–1414, Aug. 1999.

[20] C. Perkins, E. Royer, and S. Das, *Ad Hoc On-Demand Distance Vector (AODV) Routing*: IETF MANET Working Group, Mar. 2000.

[21] M. Pursley, H. Russell, and P. Staples, "Routing for multimedia traffic in wireless frequency-hop communication networks," *IEEE J. Select. Areas Comm.*, vol. 17, no. 5, pp. 784–792, May 1999.

[22] S. Ross, *Introduction to Stochastic Dynamic Programming*. Orlando, FL: Academic, 1983.

[23] P. Samar, M. Pearlman, and Z. Haas, "Independent zone routing: An adaptive hybrid routing framework for ad hoc wireless networks," in *IEEE/ACM Trans. Networking*, vol. 12, Aug. 2004, pp. 595–608.

[24] A. Viterbi, *CDMA: Principles of Spread Spectrum Communications*. Reading, MA: Addison-Wesley, 1995.

[25] J. Wieselthier, G. Nguyen, and A. Ephremides, "Algorithms for bandwidth-limited energy-efficient wireless broadcasting and multicasting," in *Proc. 2000 IEEE Military Communications Conf.*, Los Angeles, CA, Oct. 2000.

Christopher Lott received the Ph.D. degree in electrical engineering and computer science from the University of Michigan, Ann Arbor, in 2001.

He is currently with the Corporate R&D Department at Qualcomm, Inc., San Diego, CA. His interests include stochastic systems, resource allocation, distributed algorithms, communication theory, dynamical systems, and wireless networks. At Qualcomm, he is a System Designer for 1xEV-DO, with a focus on MAC design.

Demosthenis Teneketzis (M'87–SM'97–F'00) received the diploma in electrical engineering from the University of Patras, Patras, Greece, and the M.S., E.E., and Ph.D. degrees, all in electrical engineering, from the Massachusetts Institute of Technology, Cambridge, in 1974, 1976, 1977, and 1979, respectively.

He is currently Professor of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor. In winter and spring 1992, he was a Visiting Professor at the Swiss Federal Institute of Technology (ETH), Zurich, Switzerland. Prior to joining the University of Michigan, he worked for Systems Control, Inc., Palo Alto, CA, and Alphatech, Inc., Burlington, MA. His research interests are in stochastic control, decentralized systems, queueing and communication networks, stochastic scheduling and resource allocation problems, mathematical economics, and discrete-event systems.