Energy-Efficient Permutation Routing in Radio Networks

Koji Nakano, Member, IEEE, Stephan Olariu, Member, IEEE, and
Albert Y. Zomaya, Senior Member, IEEE

Abstract—A radio network (RN, for short) is a distributed system populated by small, hand-held commodity devices running on batteries. Since recharging batteries may not be possible while on mission, we are interested in designing protocols that are highly energy efficient. One of the most effective energy-saving strategies is to mandate that the stations go to sleep whenever they do not transmit or receive messages. It is well known that a station is expending power while its transceiver is active, that is, while transmitting or receiving a packet. It is perhaps surprising at first that a station is expending power even if it receives a packet that is not destined for it. Since, in single-hop radio networks, every station is within transmission range from every other station, the design of energy-efficient protocols is highly nontrivial. An instance of the permutation routing problem involves \( p \) stations of an RN, each storing \( \frac{1}{2} \) items. Each item has a unique destination which is the identity of the station to which the item must be routed. The goal is to route all the items to their destinations while expending as little energy as possible. Since, in the worst case, each item must be transmitted at least once, every permutation routing protocol must take \( \frac{3}{2} \) time slots. Similarly, each station must be awake for at least \( \frac{3}{2} \) time slots to transmit and/or receive packets. Our main contribution is to present an almost optimal energy-efficient permutation routing protocol for a \( k \)-channel, a \( p \)-station RN that routes \( n \) packets in at most \( (2d + 2b + 1) \frac{3}{2} + k \) time slots with no station being awake for more than \( (4d + 7b - 1) \frac{3}{2} \) time slots, where \( d = \left\lfloor \frac{b}{a} \right\rfloor \), \( b = \left\lfloor \frac{4a}{b} \right\rfloor \), and \( k \leq \frac{3}{2} \). Since, in most real-life situations, the number \( n \) of packets to route, the number \( p \) of stations in the RN, and the number \( k \) of channels available satisfy the relation \( k \ll p \ll n \), it follows that \( d \) and \( b \) are very small.

Index Terms—Radio networks, rapidly deployable networks, wireless communications, energy-efficient protocols, reservation protocols, permutation routing.

1 INTRODUCTION

In recent years, wireless and mobile communications have seen an explosive growth, both in terms of the number of services provided and the types of technologies that have become available. Indeed, cellular telephony, radio paging, cellular data, and even rudimentary multimedia services have become commonplace and the demand for enhanced capabilities will continue to grow into the foreseeable future [1], [5], [9], [10], [16], [20], [24], [30], [40], [41], [43], [47], [49]. It is anticipated that, in the not-so-distant future, mobile users will be able to access, while on the move, data and other services, such as electronic mail, video telephony, stock market news, map services, and electronic banking [10], [20], [24], [30], [40], [41], [49].

Unlike the well-studied cellular systems that assume the existence of a robust infrastructure, radio networks must be rapidly deployable, self-organizing, and capable of multimedia service support. Radio networks suit well the needs specific to disaster relief, search-and-rescue, law enforcement, collaborative computing, interactive mission planning, and other special-purpose applications [20], [30], [31], [36], [40].

1.1 Radio Networks

The types of applications supported by personal communications and mobile computing require a robust radio network that can be rapidly deployed and does not rely on preexisting infrastructure. The first such network was the PRNET—a packet radio network—developed in the 1970s [9], [20].

An RN is a distributed system with no central arbiter, consisting of \( p \) radio transceivers, henceforth referred to as stations. It is often the case that the stations of the RN are simple hand-held devices running on batteries and, therefore, saving battery power is exceedingly important as recharging batteries may not be possible while on a mission [5], [6], [14], [16], [39], [42], [45]. We assume that each of the \( p \) stations has a unique ID in the range \([1, p]\). This is a reasonable hypothesis because, as shown in [11] and [32], the task of assigning unique ID numbers to the stations of an RN can be performed very efficiently. Indeed, they have shown that, even if the stations do not have ID numbers initially, one can devise such a protocol
that terminates with high probability in $O(\log k)$ time slots where $k$ is the number of channels available.

While the computational power of the devices used in the end units is rapidly increasing, the lifetime of batteries is not expected to improve much in the foreseeable future [6, 39]. Given the mission-critical nature of radio network communications and the fact that virtually all end units are small devices that run on batteries, there is a clear need to design protocols that are both power and mobility-aware [43, 44, 49].

The focus of this work is on low-mobility single-hop RNs where every station can receive the transmission of every other station in the network. Such a network arises, for example, when several researchers meet in a conference room and wish to exchange data among themselves without the use of any preexisting infrastructure.

We note that, in a more general context, single-hop RNs are key ingredients in handling routing in multihop mobile radio networks [11, 7, 9, 30]. As a preliminary step, a multihop radio network is partitioned into several single-hop networks, commonly referred to as clusters [20, 30]. Routing is performed locally in each such single-hop network. Repeaters, or gateways, are used to route between clusters [20, 30]. It is worth noting that, in addition to helping with scalability and robustness, aggregating stations into single-hop clusters and, further, into superclusters has the additional benefits of conserving battery power, promoting spatial code reuse, and concealing the details of global network topology from individual stations.

In most practical situations, the stations of a cluster are located in close physical proximity—the diameter of a typical cluster rarely exceeds a few hundred meters. An important side benefit of clustering is that, in a cluster, every station is within transmission range of every other station, implementing, essentially, a local area ad hoc network [7, 41]. Thus, all the stations in a cluster can use the same transmitter-oriented CDMA code and, thus, avoid intercluster collisions [20, 30]. As it turns out, clusters are key ingredients in handling routing in hierarchically organized multihop ARNs [7, 9, 20, 30, 41]. In such an environment, routing of messages proceeds in stages: First, routing takes place locally within a cluster. Next, repeaters, or gateways, are used to route the message from one cluster to the next. This is then repeated until, eventually, the message is delivered to the intended destination [20, 30].

In this work, we shall refer to a $k$-channel, $p$-station, single-hop radio network as RN($p, k$). The RN($p, k$) involves $p$ radio stations, $S(1), S(2), \ldots, S(p)$, and $k$ disjointed radio frequency channels that we denote by $C(1), C(2), \ldots, C(k)$. The stations are assumed to have the computing power of a laptop computer. In particular, they all run the same protocol.

As customary, time is assumed slotted and all transmissions take place at slot boundaries [7, 9]. We assume that the stations have a local clock that keeps synchronous time by interfacing with a Global Positioning System (GPS, for short) [13, 27, 36, 38]. It is widely known that, under current technology, the commercially available GPS systems provide location information accurate to within 22 meters as well as time information accurate to within 100 nanoseconds [13]. In particular, this allows the stations to detect time slot boundaries and, thus, to synchronize.

For practical reasons, we assume that the duration of a time slot corresponds to the transmission time of one data packet. In every time slot, a station can tune to one of the $k$ channels and/or transmit a data packet on one of the channels. These two channels may or may not be distinct.

### 1.2 Energy-Efficient Protocols

It is becoming more and more apparent that battery life imposes severe constraints on the large-scale deployment of mobile computing technology. Thus, reducing power consumption is an important goal in protocol design because battery life is not expected to increase significantly in the coming years [24, 31, 39, 45].

We assume that the stations of the RN are simple, handheld devices running on batteries and, therefore, saving power is exceedingly important as recharging batteries may not be possible while on a mission [6, 23, 25]. As it turns out, significant energy is being expended by a station while its transceiver is active, that is, while transmitting or receiving a packet. As an example, the DEC Roamabout portable radio consumes about 5.76 watts during transmission, 2.88 watts during reception, and 0.35 watts while in idle mode [44]. It is, perhaps, less well known and somewhat surprising that a station is expending power even if it receives a packet that is not destined for it [6, 16, 24, 39, 42, 45, 47]. For single-hop RNs, the problem is compounded by the fact that, as discussed, every station is within transmission range of every other station. Consequently, we are interested in developing protocols that allow the stations to power off their transceiver (i.e., go to sleep) whenever they are not transmitting or receiving packets.

One of the key issues in radio networks is media access. Random access methods, such as Carrier Sense Multiple Access (CSMA) and its variants, have been effectively employed [1], [2], [3], [4], [18], [19], [48]. These schemes are simple to implement and robust. However, due to random conflicts, a fraction of the bandwidth is wasted for conflict resolution [9, 48]. More recently, a number of conflict-free multiple access schemes for radio networks have been proposed in the literature. Early solutions included the Token-Passing Access Method [9], the HYPERchannel network [19], and the CSMA/SD [18]. Along with the well-known TDMA scheme, these are particular instances of more general Demand Assignment Multiple Access (DAMA) schemes that have been proposed for transmission networks [4], [12], [17]. The main idea behind the DAMA scheme is that the stations that wish the transmission on a given channel are ordered in a logical ring according to which they are granted transmission access to the channel [17].

Recently, Sivalingam et al. [45] looked at the power consumption characteristics of several media access

---

1. It is well documented that GPS systems using military codes achieve a level of accuracy that is orders of magnitude better than their commercial counterparts [13, 27].
(MAC) protocols for single-hop RN\((p,k)\) networks. Their conclusion is that protocols that require transmission contention resolution feature a higher energy consumption than DAMA or reservation-based protocols. Since collisions of packets and the need for subsequent retransmission adversely impact the energy efficiency of protocols in radio networks, all our protocols will be reservation based.

Accordingly, all the protocols devised in this paper are reservation-based DAMA protocols, resulting in conflict-free transmissions. We also note that our protocols are not TDMA-based. The main reason for this is that TDMA is prone to wasting bandwidth when a station with an allocated time slot has no data to transmit.

In light of the previous discussion, we judge the goodness of a protocol by the following two yardsticks:

- the overall number of time slots required by the protocol to terminate and,
- for each individual station, the total number of awake time slots.

The goals of optimizing these two parameters are, of course, conflicting. It is relatively straightforward to minimize overall completion time at the expense of energy consumption. Similarly, one can minimize energy consumption at the expense of completion time. The challenge is to strike a sensible balance between the two by designing protocols that take a small number of time slots to terminate while being, at the same time, as energy efficient as possible.

### 1.3 Permutation Routing—State of the Art

Suppose that \(n\) items are stored by the \(p\) stations of an RN such that every station stores \(\frac{n}{p}\) items. Each item has a unique destination which is the identity of the station to which the item must be routed. The goal is to route all the items to their destinations. In permutation routing, every station is the destination of \(\frac{n}{p}\) items. We assume that each packet can store either an item to be routed or any data with \(O(\log n)\) bits, such as the index of a station, the number of items, etc. For all practical purposes, this prohibits large aggregates from being sent in one packet.

Permutation routing arises naturally in various contexts in distributed systems where traffic is known to be approximately uniform, with every host transmitting and receiving roughly the same number of packets. For an excellent discussion of permutation routing, see [21] where many applications are summarized.

We refer the reader to Fig. 1 for an illustration of the permutation routing problem with \(n = 27\) and \(p = 9\). For simplicity, for each packet, we only indicate its destination station. As an example, station \(S(1)\) initially stores \(\frac{n}{p} = 3\) packets destined to stations \(S(3), S(5),\) and \(S(8)\). The packets whose destination is station \(S(1)\) are stored by stations \(S(2), S(7),\) and \(S(9)\).

Since, in the worst case, each of the \(n\) packets must be transmitted at least once, the permutation routing on the RN\((p,k)\) needs at least \(\frac{n}{p}\) time slots, regardless of whether or not transmission conflicts are allowed. Concerning the energy efficiency, observe that, in the worst case, every station must be awake for at least \(\frac{n}{p}\) time slots to transmit/receive packets.

It is important to note the asymmetry inherent in permutation routing: While each station knows the exact destination of all the items it holds, no station knows the source of the items destined for it. In particular, this precludes any a priori agreement between sources and destinations concerning the time slot and/or the channel on which packets will have to be transmitted.

Recently, Nakano et al. [34], [35] presented permutation routing protocols for RNs running in \(2 \frac{n}{p} + k - 1\) time slots subject to \(k\) satisfying \(k \leq \sqrt{\frac{n}{p}}\). The protocols of [34] and [35] trade total number of slots for energy efficiency. Indeed, these protocols run extremely fast at the expense of high energy consumption since every station must be awake for \(2 \frac{n}{p} + k - 1\) time slots, which, in general, is much larger than the lower bound \(\frac{n}{p}\).

### 1.4 Our Contributions

The main contribution of this work is to present energy-efficient permutation routing protocols for single-hop, multichannel radio networks. We begin by presenting a simple permutation routing protocol on the single-channel RN running in \(n + p^2 - p\) time slots, with no station being awake for more than \(\frac{2n}{p} + 2p\) time slots. This protocol is optimal whenever \(p \leq \sqrt{n}\). We then go on to generalize this protocol to run in \(2pn\) time slots, with no station being awake for more than \(\frac{4n}{p}\) time slots and where \(d = \left\lfloor \frac{\log p}{\log 2} \right\rfloor\).

Finally, using our energy-efficient permutation routing protocol for single-channel RNs as a stepping stone, we devise an almost optimal energy-efficient permutation routing protocol for the RN\((p,k)\) that routes \(n\) items in at most \((2d + 2b + 1)k + \frac{2n}{p}\) time slots with no station being awake for more than \((4d + 7b - 1)k + \frac{2n}{p}\) time slots and where \(d = \left\lfloor \frac{\log p}{\log 2} \right\rfloor, b = \left\lfloor \frac{\log k}{\log 2} \right\rfloor,\) and \(k \leq \sqrt{\frac{n}{p}}\).

To put our results in perspective, Table 1 offers a synopsis of the performance characteristics of our energy-efficient permutation routing protocols for various values of \(p, n,\) and \(k\). In the table, \(T_s\) and \(T_a\) denote, respectively, the total number of time slots required by the protocol and the largest number of time slots during which each station is awake.

For example, when \(p = 100, n = 1,000,\) and \(k = 2,\) our protocol runs in \(T_s = 3,502\) time slots with no station being awake for more than \(T_a = 140\) time slots. Further, note that
\[ k \leq \sqrt{\frac{p}{n}} \] is not satisfied if \( p = 100 \) and \( k = 8 \). In this case, our protocol uses the first seven channels because \( p = 100 \) and \( k = 7 \) satisfies \( k \leq \sqrt{\frac{p}{n}} \). Thus, the performance is evaluated for \( k = 7 \) channels, for \( p = 100 \), and \( k \geq 8 \).

The reader will not fail to notice that, as illustrated in Table 1, the number of awake time slots per station is much smaller than the total completion time of the protocol. Thus, our protocol features a very desirable energy efficiency characteristic.

The remainder of this work is organized as follows: Section 2 addresses the permutation routing problem in the case of single-channel RNs. In Section 3, these results are extended to the case where \( k \) channels are available. Finally, Section 4 offers concluding remarks and points out directions for further investigations.

### 2 Permutation Routing on the Single-Channel RN

The main goal of this section is to propose various energy-efficient permutation routing protocols for the single-channel \( p \)-station RN. We assume that each of the \( p \) stations initially stores \( \frac{n}{p} \) items and the goal is to route these items to their destinations.

We note that in real-life applications, the number \( n \) of items, the number \( p \) of stations, and the number \( k \) of channels available satisfy the relation

\[ k \ll p \ll n. \]  \hspace{1cm} (1)

In other words, the number \( n \) of items to be routed is much larger than the number \( p \) of stations, which, in turn, is much larger than the number \( k \) of available frequency channels.

If energy efficiency is not an issue, permutation routing on the single-channel RN is very easy and has been discussed in the literature [34]. Indeed, consider an instance of the permutation routing problem involving \( n \) items stored \( \frac{n}{p} \) items per station in an RN\((p, 1)\). Taking turns, each station transmits its items, one by one, and every station tunes to the channel. In each time slot, the station whose identity matches the destination of the item being transmitted copies the item in its local memory. It should be clear that this protocol completes the permutation routing in \( n \) time slots. However, the protocol is not energy efficient because every station has to be awake to monitor the channel during \( n \) time slots.

#### 2.1 Routing in the Fine-Grain Case

If each station of an RN\((p, k)\) stores a single item, the following permutation routing protocol attains optimal performance, both in terms of the total number of time slots and energy efficiency.

Suppose that station \( S(i) \) stores an item whose final destination is station \( S(j) \). Notice the asymmetry of this situation: Station \( S(i) \) knows the destination of the item it stores, however, station \( S(j) \) does not know which station stores the item destined for it. It is intuitively clear that, in order to achieve energy efficiency, the stations \( S(i) \) and \( S(j) \) must somehow communicate. The aforementioned asymmetry seemingly makes this communication impossible since \( S(j) \) does not know who the sender of the item is.

Somewhat surprisingly, there is a simple way in which stations \( S(i) \) and \( S(j) \) can communicate. The idea is that they both have enough information to compute the ordered pair

\[ \left( \left\lfloor \frac{j}{k} \right\rfloor, (j - 1) \mod k + 1 \right). \]

We note that:

- The first component \( \left\lfloor \frac{j}{k} \right\rfloor \) of the ordered pair is an integer between 1 and \( \left\lfloor \frac{n}{k} \right\rfloor \). This represents the time slot in which station \( S(i) \) will transmit the item.
The second component \((j - 1) \mod k + 1\) is an integer between 1 and \(k\). This is the channel on which the item will be sent. Trivially, the protocol terminates in \(\lceil \frac{n}{p} \rceil\) time slots with no station being awake for more than two time slots: one for transmitting and the other for receiving an item. The correctness of the protocol follows from the fact that, for every natural number \(j\), the map 
\[
j \rightarrow \left( \left\lceil \frac{j}{k} \right\rceil, (j - 1) \mod k + 1 \right)
\]
is one-to-one and, therefore, no two ordered pairs will have the same first and second component. Put differently, no two stations will attempt to transmit on the same channel in the same time slot. Thus, the protocol is correct.

Unfortunately, if the stations store two or more items, that is, if \(\frac{n}{p} \geq 2\), the above protocol does not work. For example, if \(\frac{n}{p} = 2\), some two stations may have items with the same destination. Since these two stations do not know of each other, they either have to run a leader election protocol or else they will transmit at random with the ensuing collision probability and energy inefficiency.

In the remainder of this work, we concentrate on permutation routing for \(n\) and \(p\) satisfying \(\frac{n}{p} \geq 2\), that is, we assume that each station stores two or more items.

### 2.2 An Energy-Efficient Reservation Protocol

It is well known that the collision of items and the need for subsequent retransmission adversely impact the energy efficiency of protocols in radio networks. The purpose of this section is to discuss the details of a simple reservation protocol that will allow stations to implicitly reserve the channel for an appropriate number of time slots to complete their transmissions without any collisions.

Consider a single-channel \(p\)-station RN populated by stations \(S(1), S(2), \ldots, S(p)\) and assume that, for every \(i\) \((1 \leq i \leq p)\), station \(S(i)\) has \(n_i\) items to transmit. Of course, at the beginning of the reservation protocol, only station \(S(i)\) knows \(n_i\).

The details of the reservation stage follow: To begin, station \(S(1)\) transmits the number \(n_1\) of items it has to transmit. At this time, all the stations are asleep (i.e., they have their transceiver powered off) except for \(S(1)\), which is transmitting, and \(S(2)\), which is picking up \(n_1\). Now, \(S(2)\) computes \(n_1 + n_2\) and transmits the result to \(S(3)\). In this time slot, all the stations are asleep except for \(S(2)\) and \(S(3)\). Upon receiving \(n_1 + n_2\), \(S(3)\) computes \(n_1 + n_2 + n_3\) and transmits the result to \(S(4)\). This is continued until, at the end of \(p - 1\) time slots, \(S(p)\) receives \(n_1 + n_2 + n_3 + \cdots + n_{p-1}\). Thus, the reservation protocol runs for \(p - 1\) time slots. Importantly, no station is awake for more than two time slots. In fact, stations \(S(1)\) and \(S(p)\) are only awake for one time slot.

Notice that, at the end of the reservation protocol, each station can determine the exact number of transmissions that will be performed by stations preceding it and, consequently, will power itself off until the exact time slot when it has to transmit. More precisely, at the end of the reservation protocol, each station \(S(i)\) \((1 \leq i \leq p)\) knows \(n_1 + n_2 + \cdots + n_{i-1}\). Thus, station \(S(i)\) will wake up in time slot \(n_1 + n_2 + \cdots + n_{i-1} + 1\) and will start transmitting its \(n_i\) items after which it will go back to sleep. To summarize, we state the following result:

**Lemma 2.1.** The reservation protocol involving the \(p\) stations of an RN \((p, 1)\) can be completed in \(p - 1\) time slots with no station being awake for more than two time slots.

### 2.3 A Simple Energy-Efficient Permutation Routing Protocol

To set the stage for our first energy-efficient permutation routing protocol, let \(S(1), S(2), \ldots, S(p)\) be the stations of an RN \((p, 1)\) and let \(a(i, j)\) \((1 \leq i, j \leq p)\) be the set of items stored by station \(S(i)\) whose final destination is station \(S(j)\). Write, further,

\[
\alpha(*) = \alpha(1, 1) \cup \alpha(1, 2) \cup \cdots \cup \alpha(p, j), \quad \text{and} \quad \alpha(i, *) = \alpha(i, 1) \cup \alpha(i, 2) \cup \cdots \cup \alpha(i, p).
\]

In other words, \(\alpha(*)\) is the set of items whose final destination is station \(S(j)\). \(\alpha(i, *)\) consists of all items stored initially by station \(S(i)\). Clearly, our job is to route, for every \(j\) \((1 \leq j \leq p)\), the items in \(\alpha(*)\) to station \(S(j)\).

Fix an arbitrary \(j\) \((1 \leq j \leq p)\). The protocol involves two stages. The first stage is a reservation stage where each station \(S(i)\) \((1 \leq i \leq p)\) will implicitly reserve the channel for an appropriate number of time slots that will allow it, in the second stage, to transmit all the items destined for \(S(j)\). Interestingly, outside of this timeframe, station \(S(i)\) will be asleep, thus saving energy.

In the first stage, we use the reservation protocol discussed in Section 2.2 which terminates in \(p - 1\) time slots, with no station being awake for more than two time slots. Notice that, at the end of the reservation stage, each station can determine the exact number of transmissions that will be performed by stations preceding it and, consequently, will power itself off until the exact time slot when it has to transmit.

In the second stage, taking turns, the stations will wake up at the right time and will transmit their items to \(S(j)\). Thus, for a given value of \(j\), the task of routing all the items whose destination is \(S(j)\) takes \(\frac{n}{p} + p - 1\) time slots. Station \(S(j)\) is awake for at most \(\frac{n}{p} + 2\) time slots while every station \(S(i)\) \((i \neq j)\) is awake for at most \(|\alpha(i, j)| + 2\) time slots.

Since the above has to be repeated for every \(j\) \((1 \leq j \leq p)\) the total number of time slots needed by the routing protocol is

\[
p \left( \frac{n}{p} + p - 1 \right) - n + p^2 - p
\]

and each station \(S(i)\) \((1 \leq i \leq p)\) is awake for at most

\[
\frac{n}{p} + 2 + \sum_{j, j}^{\left|\alpha(i, j)\right| + 2} \leq \frac{n}{p} + 2p + |\alpha(i, *)| - 2 \left( \frac{n}{p} + p \right)
\]
time slots. The following result summarizes our findings:

**Lemma 2.2.** The task of permutation routing involving \(n\) items can be performed on the RN \((p, 1)\) in \(n + p^2 - p\) time slots with no station being awake for more than \(2p + 2p\) time slots.
The observant reader may have noticed small improvements to the protocol we just discussed. For starters, station $S(j)$ does not really have to participate in the reservation stage that precedes the routing of items in $\alpha(*, j)$. Thus, the reservation stage can be performed in $p - 2$ time slots.

Next, there is an asymmetry in the awake time of stations: $S(1)$ and $S(p)$ are awake for one time slot and all the others for two. We can help this by running the reservation protocol for $\alpha(*, j)$ ($1 \leq j \leq p$) in the order $S(j + 1), S(j + 2), \ldots, S(p), S(1), S(2), \ldots, S(j - 1)$.

With this trick, the entire protocol runs in $n + p^2 - 2p$ time slots and the awake time slots per station can be reduced to at most $2\frac{n}{p} + 2p - 4$. Thus, we have the following companion result to Lemma 2.2:

**Corollary 2.3.** The task of permutation routing involving $n$ items can be performed on the RN($p, 1$) in $n + p^2 - 2p$ time slots with no station being awake for more than $2\frac{n}{p} + 2p - 4$ time slots.

In spite of their intuitive appeal, the improvements in Corollary 2.3 are modest. In the next subsections, we will be discussing significant ways of improving the performance of this protocol.

### 2.4 A More Efficient Permutation Routing Protocol

The protocol discussed in Section 2.3 is time and energy efficient only when $p^2 \leq n$. Clearly, the bottleneck is the reservation stage that takes $p^2 - p$ time slots. In case $p^2 > n$, the performance of the protocol is suboptimal, both in terms of the total number of time slots and the individual awake time per station. Since every conflict-free permutation routing protocol must involve a reservation stage, any improvements will have to come from completing the reservation stage more efficiently.

Assume that $p^2 > n$. The idea that we shall exploit is to avoid running the reservation protocol for each station separately. Instead, intuition suggests partitioning the stations into several groups and running a single reservation stage per group.

To implement this idea, partition the $p$ stations into $\sqrt{p}$ equal-sized groups $G(1), G(2), \ldots, G(\sqrt{p})$ such that group $G(i)$ ($1 \leq i \leq \sqrt{p}$) contains $\sqrt{p}$ stations $S((i - 1)\sqrt{p} + 1), S((i - 1)\sqrt{p} + 2), \ldots, S(i\sqrt{p})$.

Let $\beta(i, j)$ ($1 \leq i \leq p; 1 \leq j \leq \sqrt{p}$) denote the set of items stored by station $S(i)$ whose final destination is some station in group $G(j)$. In other words,

$$\beta(i, j) = \alpha(i, (j - 1)\sqrt{p} + 1) \cup \alpha(i, (j - 1)\sqrt{p} + 2) \cup \cdots \cup \alpha(i, j\sqrt{p}).$$

Further, let

$$\beta(*, j) = \beta(1, j) \cup \beta(2, j) \cup \cdots \cup \beta(p, j)$$

denote the set of items whose final destination is some station in group $G(j)$. The details of the protocol are spelled out as follows:

**Protocol Square-root**

**Phase 0.** Partition the $p$ stations into $\sqrt{p}$ groups $G(1), G(2), \ldots, G(\sqrt{p})$ of $\sqrt{p}$ stations each such that group $G(i)$ ($1 \leq i \leq \sqrt{p}$) contains the $\sqrt{p}$ stations $S((i - 1)\sqrt{p} + 1), S((i - 1)\sqrt{p} + 2), \ldots, S(i\sqrt{p})$.

**Phase 1.** Route all the items in $\beta(*, j)$ ($1 \leq j \leq \sqrt{p}$) to the stations in group $G(j)$.

**Phase 2.** In each group $G(j)$ ($1 \leq j \leq \sqrt{p}$) route the items locally to their final destination.

We refer the reader to Fig. 2 for an illustration of protocol Square-root. Phase 0 does not take any interstation communication since each station can determine the group to which it belongs by performing a simple algebraic computation on its ID. Phase 1 involves a reservation stage followed by a data transfer stage. For an arbitrary $j$ ($1 \leq j \leq \sqrt{p}$), we run the reservation protocol discussed in Section 2.2 that terminates in $p - 1$ time slots with no station being awake for more than two time slots. At the end of the reservation stage, every station in group $G(i)$ ($1 \leq i \leq \sqrt{p}$) knows the exact time slot when it should start transmitting its $|\beta(i, j)|$ items for the stations in $G(j)$.

Every station in group $G(j)$ receives exactly $\frac{n}{\sqrt{p}}$ items from $\beta(*, j)$. Observe that the task of transmitting all items in $\beta(*, j)$ to the stations in group $G(j)$ takes $|\beta(*, j)| = \frac{n}{\sqrt{p}}$ time slots. Also, each station $S(i)$ in group $G(j)$ is awake for $\frac{n}{p} + |\beta(i, j)|$ time slots. Each station $S(i)$ outside of $G(j)$ is awake for $|\beta(i, j)|$ time slots.

To complete Phase 1 of the protocol, the above has to be performed for all $j$, $1 \leq j \leq \sqrt{p}$. Thus, altogether, Phase 1 requires at most

$$(n\sqrt{p} + p - 1) = n + p^2 - \sqrt{p}$$

time slots. Further each station $S(i)$ is awake for at most

$$\frac{n}{p} + \sum_{j=1}^{\sqrt{p}} |\beta(i, j)| + 2\sqrt{p} = \frac{n}{p} + |\beta(*, j)| + 2\sqrt{p} = 2(\frac{n}{p} + \sqrt{p})$$

time slots.
In Phase 2, we need to route the items locally within each group $G(j) \ (1 \leq j \leq \sqrt{p})$. Recall that group $G(j)$ contains $\frac{n}{p}$ stations and that each of them is equally loaded with $\frac{n}{p}$ items. By Lemma 2.2, the local routing in $G(j)$ can be performed in at most $\frac{n}{p^2} + p$ time slots with no station being awake for more than $2\left(\frac{n}{p} + \sqrt{p}\right)$ time slots. For all groups combined, Phase 2 can be performed in at most $\left(\frac{n}{p^2} + p\right) \cdot \sqrt{p} - n + p^2$ time slots. Thus, we have proven the following result:

**Lemma 2.4.** The task of permutation routing involving $n$ items can be performed on the RN($p,1$) in at most $2\left(n + p^2\right)$ time slots with no station being awake for more than $4\left(\frac{n}{p} + \sqrt{p}\right)$ time slots.

Again, some improvements are possible. First, in Phase 1, for an arbitrary $j$ (1 ≤ $j$ ≤ $\sqrt{p}$), it is not really necessary for stations in group $G(j)$ to transmit the items destined for group $G(j)$. Thus, the reservation protocol is run a bit differently.

First, a reservation protocol is run on the stations in groups

$$G(1), G(2), \ldots, G(j - 1), G(j + 1), \ldots, G(\sqrt{p}),$$

with the goal of determining the exact moment at which each such station should start transmitting its items to group $G(j)$. While this reservation protocol is being performed, the stations in group $G(j)$ are powered off. This reservation protocol takes $p - \sqrt{p} - 1$ time slots and no station is awake for more than two time slots.

Next, a local reservation protocol is run on $G(j)$ as follows: Let $z_1, z_2, \ldots, z_{\sqrt{p}}$ be the number of items destined for some station in $G(j)$ stored by the various stations in group $G(j)$. Since we wish to conserve energy at each individual station level, no station in group $G(j)$ will transmit items to group $G(j)$. Running the reservation protocol on $z_1, z_2, \ldots, z_{\sqrt{p}}$ allows each station in $G(j)$ to know exactly how many items it has to pick up so that all the stations in $G(j)$ store exactly $\frac{n}{p}$ items from $\beta(\ast, j)$. In particular, each station in $G(j)$ will be asleep until it will have to wake up to receive its share of items. This latter reservation protocol takes $\sqrt{p} - 1$ time slots and no station is awake for more than two time slots. Thus, for all the groups combined, Phase 1 runs in $n + p^2 - 2\sqrt{p}$ time slots and no station is awake for more than $\frac{n}{p^2} + 2\sqrt{p}$ time slots.

In Phase 2, routing is performed locally in each group. Now, Corollary 2.3 guarantees that, for a given group, this phase requires $\frac{n}{p^2} + p - 2\sqrt{p}$ time slots and no station is awake for more than $\frac{n}{p^2} + 2\sqrt{p} - 4$ time slots. For all the groups combined, Phase 2 takes $n + p^2 - 2p$ time slots and no station needs to be awake for more than $\frac{n}{p^2} + 2\sqrt{p} - 4$ time slots.

Phase 1 and Phase 2 combined take

$$2n + 2p^2 - 2p - 2\sqrt{p}$$
time slots and no station is awake for more than

$$4\left(\frac{n}{p} + \sqrt{p} - 1\right)$$
time slots. Thus, we have proven the following result:

**Corollary 2.5.** The task of permutation routing involving $n$ items can be performed on the RN($p,1$) in $2n + 2p^2 - 2p - 2\sqrt{p}$ time slots and no station needs to be awake for more than $4\left(\frac{n}{p} + \sqrt{p} - 1\right)$ time slots.

As before, in spite of their appeal and cleverness, the improvements obtained in Corollary 2.5 are not substantial. The goal of the next section is to generalize the idea of protocol Square-root and to show that this can lead to substantial efficiency gains.

### 2.5 A General Permutation Routing Protocol

The main purpose of this subsection is to further extend and generalize the ideas discussed in Section 2.4. Consider a single-channel $p$-station RN and let $d$ be an integer in the range 1 to $[\log p]$. We assume that there are $n (n > 2p)$ items to route to their destinations.

Instead of partitioning the set $S$ of stations into $\sqrt{p}$ groups of $\sqrt{p}$ stations each, we partition $S$ into $p^2$ groups $G(1), G(2), \ldots, G(p^2)$ of $p^{\frac{d}{2}}$ stations each such that for every $j \ (1 \leq j \leq p^2)$, group $G(j)$ involves stations $S((j - 1)p^{\frac{d}{2}} + 1), S((j - 1)p^{\frac{d}{2}} + 2), \ldots, S(jp^{\frac{d}{2}})$.

Notice that, for $d - 1$, we get the protocol of Section 2.3 while, for $d - 2$, we rediscover protocol Square-root. The new protocol is recursive, as shown below.

**Protocol PermutationRouting(A, d);**

1. if $|A| = 1$ exit the protocol;
2. partition the $|A|$ stations into $p^d$ groups $G(1), G(2), \ldots, G(p^d)$ of $\frac{\lfloor d \rfloor}{p^d}$ stations each;
3. for $j - 1$ to $p^d$ do
4. route to $G(j)$ all the items whose final destination is some station in group $G(j)$;
5. endfor;
6. for $j - 1$ to $p^d$ do
7. PermutationRouting($G(j), d$);
8. endfor;
9. end.

It is easy to confirm that the entire task of permutation routing amounts to the call $\text{Route}(S, d)$. We are now taking a closer look at this recursive call. Line 4 involves a reservation stage and a data transfer stage. Again, let $\beta(i, j)$, $(1 \leq i \leq p; 1 \leq j \leq p^d)$, denote the set of items stored by station $S(i)$ whose final destination is some station in group $G(j)$. In other words,

$$\beta(i, j) \equiv \alpha(i, (j - 1)p^{\frac{d}{2}} + 1) \cup \alpha(i, (j - 1)p^{\frac{d}{2}} + 2) \cup \cdots \cup \alpha(i, jp^{\frac{d}{2}}).$$

2. To avoid inconsequential complications, we assume that $\frac{\lfloor d \rfloor}{p^d}$ is an integer.
Further, we write
\[ \beta(i,j) = \beta(1,j) \cup \beta(2,j) \cup \ldots \cup \beta(p,j) \]
and note that
\[ |\beta(i,j)| = \frac{n}{p^3}. \]
By using the reservation protocol discussed in Section 2.2, each station in group \( G(i) \) (\( 1 \leq i \leq p^3 \)) can determine the exact time slot when it should start transmitting its items to group \( G(j) \). For a fixed \( j \), this reservation stage takes \( p - 2 \) time slots\(^3\) and no station is awake for more than two time slots.

At the end of the reservation stage:

- Every station in group \( G(i) \) (\( i \neq j \)) knows the exact time slot when it should start transmitting its \( |\beta(i,j)| \) items to the stations in \( G(j) \).
- Every station in group \( G(j) \) knows exactly how many items it has to pick up so that all the stations in \( G(j) \) store exactly \( \frac{n}{p} \) items from \( \beta(i,j) \). In particular, they will be asleep until they will have to wake up to receive their share of items.

In the data transfer stage, the task of routing the items in \( \beta(i,j) \) to the stations in group \( G(j) \) takes
\[ |\beta(i,j)| = \frac{n}{p^3} \]
time slots. Also, a generic station \( S(i) \) in group \( G(j) \) is awake for at most \( \frac{n}{p} \) time slots, while a station \( S(i) \) outside of \( G(j) \) is awake for \( |\beta(i,j)| \) time slots.

In order to complete the for loop in lines 3-5, the above has to be performed for all \( j \), \( 1 \leq j \leq p^3 \). Thus, altogether, this loop requires at most
\[ p^3 \left( \frac{n}{p} + p - 2 \right) = n + p^{1+\frac{1}{3}} - 2p^3 \]
time slots. Further, each station \( S(i) \) is awake for at most
\[ \frac{n}{p} + \sum_{j=1}^{p^3} |\beta(i,j)| + 2p^3 = \frac{2n}{p} + 2p^3 \]
time slots.

Note that, at the end of the for loop in lines 3-5, the original routing problem has been decomposed into \( p^3 \) subproblems, each local to one of the \( p^3 \) groups. The second for loop involves proceeding recursively to solve each subproblem individually. Each subgroup involves \( p^{1+\frac{1}{3}} \) stations, each storing exactly \( \frac{n}{p} \) items.

For this purpose, again we partition a generic group \( G(j) \) into \( p^3 \) subgroups and proceed to a reservation stage as above. It is easy to confirm that each group \( G(j) \) is partitioned into \( p^3 \) groups of \( p^{1+\frac{1}{3}} \) stations each.

At this point, the reader should be in a position to confirm that the recurrence describing the total number of time slots required by the protocol to terminate is:
\[ T(|A|) = \begin{cases} p^3T \left( \frac{|A|}{p^3} \right) + |A|^\frac{n}{p} + |A|^p - 2p^3 & \text{if } |A| \geq p^3 \\ 0 & \text{if } |A| = 1. \end{cases} \]
Since, in our case, \( |A| = |S| = p \), routine manipulations reveal that the solution to recurrence (2) is
\[ T(p) = d_n + dp^{\frac{n}{p}} - 2p(p - 1) \]
Now, (3) implies that
\[ T(p) < d_n + dp^{\frac{n}{p}} - 2(p - 1). \]
Similarly, it is easy to see that no station will have to be awake for more than \( 2d(p^{1+\frac{1}{3}} + p^3) \) time slots. Thus, we have proven the following important result:

**Theorem 2.6.** For every integer \( d \) \((1 \leq d \leq |\log p|)\), the task of permutation routing involving \( n \) items can be performed on the RN \((p,1)\) in fewer than \( d(n + p^{1+\frac{1}{3}}) - 2(p - 1) \) time slots with no station being awake for more than \( 2d(n + p^{1+\frac{1}{3}}) \) time slots.

Recall that we assumed \( \frac{n}{p} \geq 2 \). In particular, when writing
\[ d = \left[ \frac{\log p}{\log |\log p|} \right], \]
we have \( 1 \leq d \leq \lfloor \log p \rfloor \) and, consequently,
\[ p^3 \leq p^{\frac{n}{p} + \frac{n}{p^3}} = \frac{n}{p}. \]

Thus, we obtain the following corollary of Theorem 2.6:

**Corollary 2.7.** The task of permutation routing involving \( n \) items can be performed on the RN \((p,1)\) in fewer than \( 2dn - 2(p - 1) \) time slots with no station being awake for more than \( \frac{2dn}{p} \) time slots and where \( d = \left[ \frac{\log p}{\log |\log p|} \right] \).

### 3 Permutation Routing on the K-Channel RN

Having discussed permutation routing in the single-channel scenario, we are now in a position to generalize our results to the case where \( k \) \((k \geq 2)\) and channels \( C(1), C(2), \ldots, C(k) \) are available. In practice, the number of channels allocated to radio networks is very limited. We model this situation by insisting that
\[ k \leq \sqrt[p]{2} \]
Consider an RN \((p,k)\) populated by \( p \) stations \( S(1), S(2), \ldots, S(p) \), each storing \( \frac{n}{p} \) items. The problem is to route the items to their destination. For this purpose, we partition the \( p \) stations into \( k \) equal-sized groups \( G(1), G(2), \ldots, G(k) \) such that group \( G(i) \) \((1 \leq i \leq k)\) contains the \( \frac{\sqrt[p]{2}}{k} \) stations
\[ S \left( (i - 1) \frac{p}{k} + 1 \right), S \left( (i - 1) \frac{p}{k} + 2 \right), \ldots, S \left( i \frac{p}{k} \right). \]
Note that, since \( k \leq \sqrt[p]{2} \), each group contains at least \( \frac{\sqrt[p]{2}}{k} \geq 2k \) stations. Let \( \gamma(i,j) \) \((1 \leq i, j \leq k)\) denote the set of items
stored by the stations in group $G(i)$ destined for some
station in group $G(j)$.

Our permutation routing protocol consists of three
phases as described next:

Protocol $k$-channel-permutation-routing

**Phase 1.** For every $i$ ($1 \leq i \leq k$), assign channel $C(i)$ to
group $G(i)$ and route locally the $\frac{p}{k}$ items in
$G(i)$ such that:

1.1. Each station in $G(i)$ stores at most $\frac{p}{2n}$ items and
1.2. Each station in $G(i)$ stores items from exactly
one set $\gamma(i,j)$.

**Phase 2.** Let $H(i)$ ($1 \leq i \leq k$) denote the set of stations
storing items whose final destination is
some station in group $G(i)$. Assign
channel $C(i)$ to $H(i)$ and route the items
stored by stations in $H(i)$ to group $G(i)$
such that every station in $G(i)$ stores
exactly $\frac{p}{k}$ items.

**Phase 3.** Route the items locally within group $G(i)$ to their
final destination.

For an illustration of the various phases of protocol
$k$-channel-permutation-routing with parameters $k = 3$,$\newline$p = 18$, and $n = 54$, we refer the reader to Fig. 3. A
detailed discussion of the three phases of the protocol
follows.

Consider a generic group $G(i)$ ($1 \leq i \leq k$). Recall that
$G(i)$ contains $\frac{p}{k}$ stations that store, collectively, $\frac{p}{k}$ items. The
main task to be accomplished in Phase 1 is to route the items in

\[
\gamma(i,*) = \gamma(i,1) \cup \gamma(i,2) \cup \cdots \cup \gamma(i,k)
\]
such that the items in $\gamma(i,j)$ are stored by $\lceil \frac{\gamma(i,j)}{2n} \rceil$ stations and Conditions 1.1 and 1.2 are satisfied.

The stations that store, at the end of Phase 1, items whose
final destination is group $G(j)$ are said to be agents for
group $G(j)$. In other words, $H(j)$ is the set of agents for
group $G(j)$.

We now evaluate the number of stations in group $G(i)$
needed to implement the scheme outlined above. This
number is bounded by:

\[
\sum_{j=1}^{k} \left\lfloor \frac{\gamma(i,j)}{2n} \right\rfloor \leq \sum_{j=1}^{k} \left( \frac{\gamma(i,j)}{2n} + 1 \right)
\]
\[
= \left| \gamma(i,*) \right| \frac{p}{2n} + k
\]
\[
= \frac{p}{2k} + k \leq \frac{p}{k} \quad \text{(since, by (4), $k \leq \sqrt{\frac{p}{2}}$)}
\]

Since $G(i)$ contains $\frac{p}{k}$ stations, the scheme is feasible. To
implement our plan, we need to run $k$ reservation protocols
similar to the one discussed in Section 2.2. The goal of the
$j$th reservation protocol is to determine:

- the total number of items in $\gamma(i,j)$ as well as the
number of stations in group $G(i)$ that will act as
agents for group $G(j)$ ($1 \leq j \leq k$) and,
- for each station in $G(i)$ that stores items from $\gamma(i,j)$,
the exact time slot when it should start transmitting
its items.

The $j$th reservation protocol takes $\frac{p}{k} - 1$ time slots with no
station being awake for more than two time slots. In one
additional time slot, the last station in group $G(i)$ will
transmit on the channel the value $|\gamma(i,j)|$. Consequently,
during the $j$th reservation protocol, no station is awake for
more than three time slots.
Thus, at the end of \( k \) reservation protocols, every station knows the identity of the group for which it will act as an agent as well as the exact time slot when it should wake up to pick up its share of items. Clearly, in Phase 1, each station transmits at most \( \frac{n}{k} \) items and receives at most \( \frac{2n}{k} \) items. In addition, it will be awake for at most \( 3k \) time slots during the \( k \) reservation protocols. Thus, each station is awake for at most \( \frac{3n}{p} + 3k \) time slots. As discussed, Phase 1 requires a total of
\[
\sum_{j=1}^{k} \left( \frac{p}{k} - 1 + |\gamma(i,j)| \right) = p - k + |\gamma(i,*)| = \frac{n}{k} + p - k
\]
time slots.

The main task to be handled in Phase 2 is to route the items stored by the stations in \( H(j) \) (\( 1 \leq j \leq k \)) to the stations in \( G(j) \) such that each station will store exactly \( \frac{n}{p} \) items. To see how this is done, recall that, for every \( i \) (\( 1 \leq i \leq k \)):

- Every station in \( H(j) \cap G(i) \) knows \( |\gamma(i,j)| \);
- Every station in \( H(j) \cap G(i) \) knows its relative rank in \( H(j) \cap G(i); \) and,
- With at most one exception, all stations in \( H(j) \cap G(i) \) store exactly \( \frac{2n}{p} \) items destined to \( G(j) \).

The last station in \( H(j) \cap G(i) \) (which may store fewer than \( \frac{2n}{p} \) items) will play an important role in the first stage of Phase 2. Specifically, the subgroup of \( H(j) \), consisting of the last station in each \( H(j) \cap G(i) \) (\( 1 \leq i \leq k \)), will run a reservation protocol among themselves. The main motivation for this is for each station in \( H(j) \cap G(i) \) to determine the number of items stored collectively by the stations in \( H(j) \cap G(i), \) (\( 1 \leq t \leq i - 1 \)). In turn, this will allow the first station in \( H(j) \cap G(i) \) to determine the exact time slot when it should start transmitting on the channel. This reservation protocol takes \( k - 1 \) time slots and is run only once. Recall that, as we already pointed out, within each \( H(j) \cap G(i) \) there is a natural ordering of the stations. Moreover, all these stations, except for the last one, have exactly \( \frac{2n}{p} \) items to send. Consequently, they can easily determine the right time when they have to wake up and transmit their items.

The stations in group \( G(j) \) are responsible for picking up \( \frac{n}{p} \) items each and, consequently, they can determine the exact time slot when their turn has come to receive items.

Since each station in \( H(j) \) stores at most \( \frac{2n}{p} \) items, it must be awake for at most \( \frac{2n}{p} \) time slots to send its items. It will also be awake for an additional \( \frac{n}{p} \) time slots to receive the items for which it is responsible (as a member of some group \( G(i) \)). Therefore, Phase 2 can be completed in \( \frac{n}{p} + k - 1 \) time slots with no station being awake for more than \( \frac{2n}{p} \) time slots.

Notice that, at the end of Phase 2, all the items whose final destination is some station in group \( G(j) \) are already in group \( G(j) \), albeit not stored by their correct destination. The goal of Phase 3 of the protocol is to route the items within each group to their final destination.

Recall that a generic group \( G(j) \) (\( 1 \leq j \leq k \)) consists of \( \frac{p}{k} \) stations each storing exactly \( \frac{n}{p} \) items. At this time, we can apply a suitably modified instance of the permutation routing protocol discussed in Section 2.5. Theorem 2.6 guarantees that, for an arbitrary \( d (1 \leq d \leq \lfloor \log \frac{p}{k} \rfloor) \), this task can be completed in \( d \left( \frac{n}{p} + \frac{p}{k} \right) - 2\frac{n}{p} + 2 \) time slots with no station being awake for more than \( 2d \left( \frac{n}{p} + \frac{p}{k} \right) \) time slots. Thus, we have proven the following result:

**Lemma 3.1.** Fix an integer \( d \) (\( 1 \leq d \leq \lfloor \log \frac{p}{k} \rfloor \)). The task of permutation routing involving \( n \) items can be performed on the \( RN(p, k) \) in
\[
(d + 2) \frac{n}{k} + d \left( \frac{p}{k} \right)^{1 + \frac{d}{2}} - 2\frac{n}{p} + p + 1
\]
time slots with no station being awake for more than
\[
(2d + 6) \frac{n}{p} + 2d \left( \frac{p}{k} \right)^{1 + \frac{d}{2}} + 3k
\]
time slots, provided that \( k \leq \sqrt{\frac{n}{p}} \).

Observe that, if \( p > \frac{n}{k} \) (equivalently, \( k > \frac{n}{p} \)), the above protocol has suboptimal performance, both in terms of time and energy efficiency. To improve the performance of the protocol, we take a second look at Phase 1 and will rely on ideas similar to those used in Section 2.4. Recall that \( \gamma(i,j) \) (\( 1 \leq i, j \leq k \)) is the set of items stored initially by the stations in group \( G(i) \) and whose destination is a station in group \( G(j) \). The main goal of Phase 1 of the protocol was to route the items in
\[
\gamma(i,*) = \gamma(i,1) \cup \gamma(i,2) \cup \ldots \cup \gamma(i,k)
\]
such that the items in \( \gamma(i,j) \) are stored by \( \lceil \frac{|\gamma(i,j)|}{\frac{p}{k}} \rceil \) stations subject to the additional constraints of Conditions 1.1 and 1.2.

We partition the set \( \gamma(i,*) \) into \( \sqrt{k} \) groups as follows: For \( 1 \leq j \leq \sqrt{k} \), write
\[
\delta(i,j) = \gamma(i,(j-1)\sqrt{k} + 1) \cup \gamma(i,(j-1)\sqrt{k} + 2) \\
\cup \ldots \cup \gamma(i,j\sqrt{k}).
\]

Our plan is to move, for every \( j \) (\( 1 \leq j \leq \sqrt{k} \)), the items in \( \delta(i,j) \) to \( \lceil \frac{|\delta(i,j)|}{\frac{n}{p}} \rceil + \sqrt{k} \) stations in group \( G(i) \) in such a way that, for every \( t \), \((j-1)\sqrt{k} + 1 \leq t \leq j\sqrt{k}) \), every station contains items from one set \( \gamma(i,t) \) only. Observe that the total number of stations necessary to implement this idea is
\[
\sum_{j=1}^{\sqrt{k}} \left( \left\lceil \frac{|\delta(i,j)|}{\frac{n}{p}} \right\rceil + \sqrt{k} \right) \leq \sum_{j=1}^{\sqrt{k}} \left( |\delta(i,j)| \frac{p}{2n} + k \right)
\]
(5)
\[
= |\gamma(i,*)| \frac{p}{2n} + k = \frac{p}{2n} + k \leq \frac{n}{k}.
\]
Since group \( G(i) \) contains \( \frac{p}{k} \) stations, this is feasible. Now, to implement our plan, Phase 1 is partitioned into the following two subphases:

**Subphase 1.1.** Using channel \( C(i) \), route the items in \( \gamma(i,*) \) such that

1.1.1. For every \( j \) (\( 1 \leq j \leq \sqrt{k} \)), the items in \( \delta(i,j) \) are
stored by a subgroup $G'(i,j)$ consisting of $\| \delta(i,j) \|_{\frac{p}{2n}} + \sqrt{k}$ stations of $G(i)$,
1.1.2. Each station in group $G(i)$ stores at most $\frac{2n}{p}$ items, and
1.1.3 Each station in group $G(i)$ stores items from exactly one set $\delta(i,j)$.

**Subphase 1.2.** Route the items within each $G'(i,j)$ such that
1.2.1. Each station in $G'(i,j)$ stores at most $\frac{2n}{p}$ items and
1.2.2. Each station in $G'(i,j)$ stores items from exactly one set $\gamma(i,t)$ with $(j-1)\sqrt{k} + 1 \leq j \leq \sqrt{k}$. We now turn to a detailed description of Subphase 1 and 1.2. Subphase 1 begins by running a reservation protocol whose goal is to allow the various stations in $\{\delta(i,1), \delta(i,2), \ldots, \delta(i,\sqrt{k})\}$ to learn when to wake up in order to transmit/receive items. As discussed in Section 2.2, this task can be completed for each $\delta(i,j)$ ($1 \leq j \leq \sqrt{k}$) in $\sqrt{k} - 1$ time slots with each station being awake for at most two time slots. Thus, overall, the $\sqrt{k}$ reservation protocols can be completed in $\sqrt{k}(\sqrt{k} - 1) = |G'(i,j)|\sqrt{k} - \sqrt{k}$ time slots with no station being awake for more than $2\sqrt{k}$ time slots.

Once the reservation stage is out of the way, the items in each $\delta(i,j)$ are moved to the corresponding $\| \delta(i,j) \|_{\frac{p}{2n}} + \sqrt{k}$ stations such that no station receives more than $\frac{2n}{p}$ items. This routing operation takes $|\gamma(1,*)| = \frac{p}{k}$ time slots. Clearly, each station will be transmitting for at most $\frac{p}{k}$ time slots and will be receiving for at most $\frac{2n}{p}$ time slots. Consequently, Subphase 1 can be completed in $\frac{p}{k} + \frac{2n}{p} = \sqrt{k}$ time slots with each station being awake for at most $3\sqrt{k} + \frac{2n}{p}$ time slots.

In Subphase 1.2, the plan is to route items within each subgroup $G'(i,j)$ such that every station contains at most $\frac{2n}{p}$ items from exactly one set $\gamma(i,t)$. At this point, the reader may wonder if $G'(i,j)$ contains sufficient stations to implement this plan. To show that our plan is feasible to implement, recall that $G'(i,j)$ contains

$$\| \delta(i,j) \|_{\frac{p}{2n}} + \sqrt{k}$$

stations and let $a_q$ ($1 \leq q \leq \sqrt{k}$) denote $|\gamma(i,j) - 1|\sqrt{k} + \sqrt{k}$. In this notation, the number of stations in $G'(i,j)$ required to implement our plan is

$$\sqrt{k} \left( \sum_{q=1}^{\sqrt{k}} a_q \right) \leq \sqrt{k} \left( \sum_{q=1}^{\sqrt{k}} \left( \frac{a_q}{2n} \right) + 1 \right) \leq \sum_{q=1}^{\sqrt{k}} \left( \frac{a_q}{2n} \right) + \sqrt{k}$$

$$= \left\| \delta(i,j) \right\|_{\frac{p}{2n}} + \sqrt{k},$$

confirming that there are enough stations in $G'(i,j)$ for our plan to work.

Next, to determine the schedule of transmissions, a reservation protocol is run for each of the sets $\gamma(i,1), \gamma(i,2), \ldots, \gamma(i,\sqrt{k})$, independently. Each such reservation protocol runs in $|G'(i,j)| - 1$ time slots with each station being awake for at most two time slots. Thus, overall, the $\sqrt{k}$ reservation protocols can be completed in $\sqrt{k} |G'(i,j)| - 1 = |G'(i,j)|\sqrt{k} - \sqrt{k}$ time slots with no station being awake for more than $2\sqrt{k}$ time slots.

Once the reservation protocols are completed, the data routing stage is completed in the obvious way. Hence, Subphase 1.2 can be completed in

$$\sum_{j=1}^{\sqrt{k}} \left( \sqrt{k} |G'(i,j)| - 1 + |\delta(i,j)| \right)$$

$$\leq \frac{n}{k} + \sqrt{k} \left( \sum_{j=1}^{\sqrt{k}} |G'(i,j)| \leq \frac{p}{k} \right)$$

$$= \frac{n}{k} + \frac{p}{\sqrt{k}} - \sqrt{k}$$

time slots and no station is awake for more than $\frac{4n}{p} + 2\sqrt{k}$ time slots.

Consequently, in this improved implementation, Phase 1 (i.e., the combination of Subphases 1 and 2) can be completed in $2\left( \frac{p}{k} + \frac{2n}{p} \right) = \sqrt{k}$ time slots with no station being awake for more than $\frac{4n}{p} + 4\sqrt{k}$ time slots. Thus, we have proven the following result:

**Lemma 3.2.** For a fixed integer $d$ ($1 \leq d \leq \log \frac{p}{k}$), an instance of permutation routing involving $n$ items can be performed on the RN $(p,k)$ in at most

$$(d + 3) \frac{n}{k} + d \left( \frac{p}{k} \right)^{\frac{1}{d}} + 2 \frac{p}{\sqrt{k}} + k$$

time slots with no station being awake for more than

$$(2d + 10) \frac{n}{p} + 2d \left( \frac{p}{k} \right)^{\frac{1}{d}} + 4\sqrt{k}$$

time slots, provided that $k \leq \sqrt{p}$.

To further improve the performance of the protocol, we borrow the technique of Section 2.5. Instead of partitioning $\gamma(i,*)$ into $\sqrt{k}$ groups, we proceed to partition $\gamma(i,*)$ into $k^2$ groups $\delta(i,1), \delta(i,2), \ldots, \delta(i,k^2)$ for some $b \geq 2$ and route the items such that:

- The items in $\delta(i,j)$ will be stored by a group of $\| \delta(i,j) \|_{\frac{p}{2n}} + k^2 - b$ stations,
- No station stores more than $\frac{2n}{p}$ items, and
- No station stores items from more than one $\delta(i,j)$.

The reservation protocols needed for conflict-free scheduling of transmissions can be performed in

$$k^2 \cdot \frac{p}{k^2 - b}$$
time slots with each station being awake for at most $3k^2$ time slots. The routing of items can be completed in $\frac{p}{k^2}$ time slots and each station will transmit at most $\frac{2n}{p}$ items and will receive at most $\frac{2n}{p}$ items. Thus, Subphase 1.1 can be completed in

$$\frac{n}{k} + \frac{p}{k^2 - b}$$
time slots with no station being awake for more than $3k^2 + \frac{3b}{p}$ time slots.

Next, we partition each of the sets

$$\delta(i, 1), \delta(i, 2), \ldots, \delta(i, k^2)$$

into $k^2$ subgroups (i.e., altogether, $k^2$ subgroups) and route the items locally within each subgroup. The reader should have no difficulty confirming that this takes

$$\frac{n}{k} + \frac{p}{k^{1-\frac{1}{2}}}$$

time slots with no station being awake for more than $3k^2 + \frac{3b}{p}$ time slots. This partition is then repeated until each station stores items from exactly one set $\gamma(i, j)$. As a result, Phase 1 can be implemented to run in

$$b\left(\frac{n}{k} + \frac{p}{k^{1-\frac{1}{2}}}\right)$$

time slots with no station being awake for more than

$$3k^2 + (4b - 1)\frac{n}{p}$$

time slots. Thus, we have the following result:

**Theorem 3.3.** Given an arbitrary integer $d \ (1 \leq d \leq \log \frac{p}{k})$ and an integer $b, \ (1 \leq b \leq \log k)$, the instance of permutation routing involving $n$ items can be performed on the RN($p, k$) in

$$(d + b + 1)\frac{n}{k} + d\left(\frac{p}{k}\right)^{\frac{1}{2} + 1} + \frac{bp}{k^{1-\frac{1}{2}}} + k$$

time slots with no station being awake for more than

$$(2d + 4b - 1)\frac{n}{p} + 2d\left(\frac{p}{k}\right)^{\frac{1}{2} + 1} + 3bk^2$$

time slots, provided that $k \leq \sqrt{\frac{p}{n}}$.

By choosing

$$d = \left\lfloor \frac{\log \frac{p}{k}}{\log \frac{p}{n}} \right\rfloor,$$

we have

$$\left(\frac{p}{k}\right)^{\frac{1}{2} + 1} \leq \frac{n}{p}$$

and, consequently,

$$\left(\frac{p}{k}\right)^{1 + \frac{1}{2}} \leq \frac{n}{k}.$$

Likewise, by choosing $b = \left\lfloor \frac{\log k}{\log n} \right\rfloor$, we have $k^2 \leq \frac{n}{p}$ and

$$\frac{p}{k^{1-\frac{1}{2}}} \leq \frac{n}{k}.$$

Thus, we obtain the following corollary of Theorem 3.3:

**Corollary 3.4.** The task of permutation routing involving $n$ items can be performed on the RN($p, k$) in at most

$$(2d + 2b + 1)\frac{n}{k} + k$$

time slots with no station being awake for more than $(4d + 7b - 1)\frac{n}{p}$ time slots, provided that $k \leq \sqrt{\frac{p}{n}}$.

4 Conclusions and Open Problems

It is becoming more and more apparent that battery life imposes severe constraints on the large-scale deployment of mobile computing technology. Reducing power consumption is an important goal in protocol design because battery life is not expected to increase significantly in the coming years [24], [31], [39], [42], [45].

As it turns out, a station is expending a significant amount of power while its transceiver is active, that is, while transmitting or receiving items. It is, perhaps, less well known and somewhat surprising that a station is expending power even if it receives an item that is not destined for it [6], [16], [24], [39], [42], [45], [47]. In single-hop networks, the problem is compounded by the fact that every station is within transmission range of every other station. The main measure of energy efficiency adopted in this paper is the extent to which the stations can power their transceiver off.

Since collisions of packets and the need for subsequent retransmission adversely impact the energy efficiency of protocols in radio networks, all our protocols will be reservation-based. Accordingly, all the protocols devised in this paper are reservation-based DAMA protocols resulting in conflict-free transmissions. We also note that our protocols are not TDMA-based. The main reason for this is that TDMA is prone to wasting bandwidth when a station with an allocated time slot has no data to transmit.

Our main contribution was to present an energy-efficient permutation routing protocol on the $k$-channel $p$-station RN that routes $n$ items in at most $(2d + 2b + 1)\frac{n}{k} + k$ time slots with no station being awake for more than $(4d + 7b - 1)\frac{n}{p}$ time slots where, $d = \left\lfloor \frac{\log p}{\log n} \right\rfloor$, $b = \left\lfloor \frac{\log k}{\log n} \right\rfloor$ and $k \leq \sqrt{\frac{p}{n}}$. We note that, in the vast majority of real-life situations, the number $n$ of items, the number $p$ of stations, and the number $k$ of channels available satisfy the relation $k \ll p \ll n$. This implies that both $d$ and $b$ are very small numbers.

A large number of problems remain open. First, it would certainly be of interest to see to what extent randomization can help with permutation routing [8]. Second, it is very important—and a highly nontrivial task—to extend our routing protocols to the multihop case. This promises to be an important and challenging step.

Additional power is being consumed by the display and by spinning hard disks. Thus, one of the main concerns related to the need to conserve battery power motivates the design of stations that have small hard disks since, as discussed, spinning a hard disk is one of the most energy-consuming tasks. We have not looked at the problem of limiting the size of the hard disks and/or spinning down the hard disks. Likewise, we have not addressed the problem of stations having different transmission power and various disk capacities. All these issues will be addressed in a number of upcoming papers.
ACKNOWLEDGMENTS

The authors wish to express their gratitude to three anonymous referees for their constructive comments and candid criticism that resulted in a much better end result. This work was supported in part by US Office of Naval Research grant N00014-97-1-0526 and by Australian Research Council grant 04/1542/001.

REFERENCES


Koji Nakano received the BE, ME, and PhD degrees from Osaka University, Japan, in 1987, 1989, and 1992, respectively. From 1992 to 1995, he was a research scientist at Advanced Research Laboratory, Hitachi Ltd. Since 1995, he has worked at Nagoya Institute of
Technology, Japan, where he is currently an associate professor with the Department of Electrical and Computer Engineering. His research interests include parallel algorithms and architectures, computational complexity, and graph theory. He is a member of the IEEE.

Stephan Olariu received the MSc and PhD degrees in computer science from McGill University, Montreal, in 1983 and 1986, respectively. In 1986, he joined the Computer Science Department at Old Dominion University where he is now a professor. He has published extensively in various journals, book chapters, and conference proceedings. His research interests include wireless networks and mobile computing, parallel and distributed systems, performance evaluation, and medical image processing. He serves on the editorial board of several archival journals, including the IEEE Transactions on Parallel and Distributed Systems, Journal of Parallel and Distributed Computing, International Journal of Foundations of Computer Science, International Journal of Computer Mathematics, VLSI Design, and Parallel Algorithms and Applications. Dr. Olariu is a member of the IEEE.

Albert Y. Zomaya received the PhD degree from the Department of Automatic Control and Systems Engineering, Sheffield University, United Kingdom. Also, he held visiting positions at Waterloo University (Canada) and the University of Missouri-Rolla. He is a professor and deputy head of the Department of Electrical and Electronic Engineering at the University of Western Australia where he also leads the Parallel Computing Research Laboratory. He is the author/coauthor of five books, more than 100 publications in technical journals, collaborative books, and conferences, and the editor of three volumes and three conference proceedings. He is currently an associate editor for the IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Systems, Man, and Cybernetics (Parts A, B, and C), Journal of Parallel Algorithms and Applications, Journal of Interconnection Networks, The International Journal on Parallel and Distributed Systems and Networks, Journal of Future Generation of Computer Systems, and The International Journal of Foundations of Computer Science. He previously served on the editorial board of the International Journal in Computer Simulation and The IFAC Control Engineering Practice Journal. He is also the founding editor of the Wiley Book Series on Parallel and Distributed Computing. He is the editor-in-chief of The Parallel and Distributed Computing Handbook (McGraw-Hill, 1996). He was elected in July 1999 to chair the IEEE Technical Committee on Parallel Processing. He is also a board member of the International Federation of Automatic Control (IFAC) Committee on Algorithms and Architectures for Real-Time Control and serves on the executive board of the IEEE Task Force on Cluster Computing. He served on the steering and program committees of several national and international conferences. He is the founding cochair of the Workshop on Biologically Inspired Solutions to Parallel Processing Problems (Florida, 1998; San Juan, 1999; Mexico, 2000). He is the stream chair of the 10th International Conference on Computing and Information (Kuwait, 2000), program vice chair of the International Conference on Parallel Processing (Toronto, 2000), general chair of the International Symposium on Parallel Architectures, Algorithms, and Networks (Perth, 1999), the program vice chair of the Second International Conference on Parallel and Distributed Computing and Networks (Brisbane, 1998), vice chair of the 11th International Conference on Parallel and Distributed Computing Systems (Chicago, 1998), and vice chair of the High Performance Computing Conference (Bangalore, 1997). He received the 1997 Edgeworth David Medal from the Royal Society of New South Wales for outstanding contributions to Australian Science. His research interests are in the areas of high performance computing, parallel and distributed algorithms, scheduling and load-balancing, computational machine learning, scientific computing, adaptive computing systems, mobile computing, data mining, and cluster, network, and megacomputing. Dr. Zomaya is a chartered engineer and a senior member of the IEEE, member of the ACM, the IEE (U.K.), the New York Academy of Sciences, the Association for the Advancement of Science, and Sigma Xi.