Topography-Independent Transmission scheduling algorithms in Multihop Packet Radio Networks* 

Jong-Hoon Youn and Bella Bose
Dept. of Computer Science
Oregon State University
Corvallis, OR 97331, U.S.A.
{jhyun, bose}@cs.orst.edu
TEL: 541-737-5573
FAX: 541-737-3014

Abstract: In this paper, based on coding theory concepts, new time scheduling algorithms for multihop packet radio networks are described. Each mobile host is assigned a word from an appropriate constant weight code of length $n$, distance $d$ and weight $w$. The host can send a message at the $j^{th}$ slot provided the assigned code has a 1 in this $j^{th}$ bit. The proposed algorithms are better than the previously known algorithms in terms of minimum system throughput and/or delay bound. The algorithms also preserve other desired properties, such as topology independence, guaranteed minimum throughput, bounded maximum delay, and fair transmission policy. In the simulation, we measure the average system throughput of transmission scheduling algorithms. The simulation results show that the proposed algorithms outperform the previously known algorithms in terms of mean system throughput.

* This work is supported by the NSF grant MIP-9705738
1. Introduction

A mobile ad hoc network (i.e. multi-hop packet radio network) consists of a cluster of mobile hosts without fixed base station or any wired backbone infrastructure. Because an ad-hoc networks does not rely on existing infrastructure and is a self-configurable network, it can be rapidly deployed in applications such as tactical communication for military, search and rescue mission, disaster recovery, etc. In mobile ad hoc networks, two mobile hosts can communicate directly with each other if they are located within their radio range. If the destination is located outside of the sender's radio range, packets are relayed via intermediate nodes located between the two nodes.

Since the wireless medium is a broadcast medium, multiple simultaneous transmission can result in collision. A medium access control (MAC) protocol defines rules that allow the mobile hosts to share medium in efficient and fair manner. Therefore, MAC protocol is a key issue that determines the performance of a packet radio network.

Wireless MAC protocols have been studied extensively since 1970s. The first MAC protocol used in multihop packet radio networks was the Carrier Sense Multiple Access (CSMA) protocol. Although the CSMA protocol prevents collision by sensing the carrier before transmission, a packet sent by a host, A, can collide at the receiver with another packet sent by a different host, B, which is outside the range of A; the host B is referred to as the hidden terminal [12]. To solve a hidden terminal problem, the Multiple Access with Collision Avoidance (MACA) protocol was proposed by Karn [11]. In MACA, a mobile host first transmits a Request To Send (RTS) control message whenever it wants to transmit a packet. When the receiver receives a RTS message, it responds with a Clear To Send (CTS) control message. On receiving the CTS message, the sender begins transmitting the data packet. Several modifications of MACA have been proposed in [2] [8] [16].

Another class of MAC protocol used in multihop wireless networks is a time-division multiple-access (TDMA) transmission schedule. In TDMA, the frequency band is split into a number of channels, which are stacked into short time units, so that a single frequency can support multiple, simultaneous data channels. The design of time scheduling in multihop mobile radio networks has been an active research topic. Most of
Previous research have focused on designing a fair collision-free scheduling algorithm which maximizes the system throughput. In [7], it is proved that the problem of scheduling the time slots with optimal throughput is NP-Complete. Also, the optimal scheduling algorithms mostly require complete topology information. Recently, an interesting scheduling algorithm which does not require any topology information, has been proposed in [4]. This algorithm is topology transparent, and guarantees that each node has at least one collision-free slot in each frame. Some variations of this scheduling scheme are introduced in [5][6], and the attempt to optimize the system throughput of the algorithm has been made in [10]. We briefly explain these topology-transparent algorithms in the next section.

The rapid growth of the real-time and multimedia applications have created a need to have constant and predictable data delivery service. In this paper, we introduce new time scheduling algorithms that guarantee minimum data throughput and bounded maximum delay. Although our approach is similar to the topology-transparent algorithms in [4][10], it is much better than the previously known topology-transparent algorithms in terms of minimum throughput and/or delay bound.

The remainder of the paper is organized as follows. In the following section, we describe some preliminaries and the previously known topology-transparent algorithms. Section 3 elaborates on the proposed transmission scheduling algorithms, and compares the minimum system throughput of the proposed algorithms to that of the previously known algorithms. Section 4 discusses the simulation results. Finally, Section 5 presents the conclusion.

2. Preliminaries and previous works

A multihop packet radio network can be represented as an undirected graph $G=(V, E)$, where $V$ is a set of vertices denoting the mobile hosts in the radio network, and $E$ is a set of edges between vertices. For any two vertices $u, v \in V$, $(u, v) \in E$ if and only if they can hear each other transmission. In this case, the vertices $u$ and $v$ are called adjacent. The degree of a node $v$, $d(v)$, is defined as the number of nodes which are adjacent to it. Each
node uses an omnidirectional antenna for communication, and the network works in half-duplex mode, which means that a node cannot transmit and receive simultaneously.

Assume there are $N$ mobile hosts in the network, i.e. $|V| = N$, and the maximum degree in the network is $D_{\text{max}} = \max d(v)$, where $v \in V$. Each node in $V$ has a transmission scheduling vector (TSV) which is defined as follows.

**Definition 1**: TSV’s are binary vectors of length $n$. The TSV of a node $A \in V$, denoted by $\text{TSV}_A$, is $a_1 \ a_2 \ a_3 \ a_4 \ ... \ a_{n-2} \ a_{n-1} \ a_n$, where

$a_i = 1$ if node $A$ has a permission to transmit in the $i^{th}$ slot, and

$a_i = 0$ otherwise, for $1 \leq i \leq n$.

The problem of transmission scheduling is equivalent to finding a schedule, $S$, which is a set of transmission scheduling vectors (TSV’s) satisfying the conditions described below. Given $N$ and $D_{\text{max}}$, for any given TSV and any other $D_{\text{max}}$ of TSV’s $\{\text{TSV}_1, \text{TSV}_2, \text{TSV}_3, ... , \text{TSV}_{D_{\text{max}}}\}$, there exists a position $j$, for $1 \leq j \leq n$, such that TSV has 1 in this position, and all other TSV’s, for $1 \leq i \leq D_{\text{max}}$, have 0 in this same position.

Since it is important to have fair bandwidth sharing, it is assumed that each host has the same number of transmission slots. Thus, the transmission scheduling problem can be formalized as follows.

**Problem 1**: Find a set of TSV’s, $S$, which satisfies the following properties.

1. $S$ has at least $N$ number of TSV’s; $|S| \geq N$.
   (This is because each host must have a unique TSV.)
2. There are exactly $q$ 1’s in a TSV.
   (This property guarantees fair bandwidth sharing.)
3. For any two TSV’s, $\text{TSV}_i$ and $\text{TSV}_j$ in $S$, there are at most $k$ positions in which both $\text{TSV}_i$ and $\text{TSV}_j$ are 1’s. In other words, there must be at least $2(q-k)$ positions in which $\text{TSV}_i$ and $\text{TSV}_j$ differ.
4. $q > D_{\text{max}} \cdot k$.
   (The properties 3 and 4 guarantee that there is at least one collision free slot in each frame.)
Chlamtac and Farago solve the transmission scheduling problem by designing a topology-transparent transmission scheduling method called as ‘Galois Radio Network Design (GRAND)’ algorithm [4]. The GRAND algorithm assumes that the number of mobile hosts \((N)\) and the maximum degree \((D_{\text{max}})\) are known in advance. This algorithm constructs TSV’s of length \(q^2\), where \(q\) is a prime or a prime power. The constructed schedules satisfy the properties specified in Problem 1. In this algorithm, a distinct polynomials of degree at most \(k\) over GF\((q)\) is assigned to each mobile host. Since the number of polynomial over GF\((q)\) of degree at most \(k\) is \(q^{k+1}\), the number of mobile hosts \((N)\) should be less than or equal to \(q^{k+1}\). Using the uniquely assigned polynomial, each mobile host calculates its transmission scheduling vector as follows. Assume \(p_i(t) = a_0 + a_1 t + a_2 t^2 + \ldots + a_{k-2} t^{k-2} + a_{k-1} t^{k-1}\) is the assigned polynomial to the node \(i\). Let the elements in GF\((q)\) be \(\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{q-1}\).

\[
\text{TSV}_i = \left[ \phi(p_i(\alpha_0)) \phi(p_i(\alpha_1)) \phi(p_i(\alpha_2)) \ldots \phi(p_i(\alpha_{q-1})) \right],
\]

where \(p_i(t) = (a_0 + a_1 t + a_2 t^2 + \ldots + a_{k-2} t^{k-2} + a_{k-1} t^{k-1})\) for \(t \in \text{GF}(q)\), and \(\phi\) is a function which maps a \(q\)-ary vector to a binary vector of length \(q\):

\[
\begin{align*}
\phi(\alpha_0) &= 1000 \ldots 00 \\
\phi(\alpha_1) &= 0100 \ldots 00 \\
\phi(\alpha_2) &= 0010 \ldots 00 \\
&\vdots \\
\phi(\alpha_{q-1}) &= 0000 \ldots 01
\end{align*}
\]

While calculating \(p_i(t)\), note that the multiplication and addition operations must be done in GF\((q)\).

Since the degree of each assigned polynomial is at most \(k\), the maximum number of slots in which \(i\) can collide with one of its neighbors is at most \(k\); this is because two distinct polynomials of degree at most \(k\) can’t have more than \(k\) common intersection points. More precisely, the maximum number of collisions for any two nodes is exactly the same
as the degree of their difference polynomial. Furthermore, if \( q > D_{\text{max}} \cdot k \), each node has at least one collision free slot in each frame.

Later, Ju and Li have given an improved topology-transparent scheduling algorithm by maximizing the minimum system throughput of the GRAND algorithm [10]. In their scheduling algorithm, the length of TSV (i.e. the length of frame) is restricted to \( p^2 \), where \( p \) is a prime. Their analysis shows that the best value of \( k \) (the degree of the polynomial) is 1 unless the network size is extremely large. More precisely, the best value of \( k \) is 1 if \( D_{\text{max}} > 0.1464 N^{1/2} \). It is also shown that for a given \( k \), the minimum system throughput is maximized when \( p = 2k D_{\text{max}} \), provided \( N^{1/(k+1)} \leq 2k D_{\text{max}} \), and \( p = N^{1/(k+1)} \), otherwise.

**Example 1:** Assume there are 35 mobile hosts in the network, and the maximum degree in the network is 5. First, the GRAND algorithm selects proper \( q \) and \( k \) values. Since \( q^{k+1} \) should be greater than or equal to 35, and \( q \) should be greater than \( 5k \), the selected \( q \) and \( k \) are 7 and 1 respectively. Then, a distinct polynomial over GF(7) of degree 1 is assigned to each mobile host. This polynomial is used to construct a TSV of length 49. Assume the polynomial \( p_A(x) = 2x + 5 \), and \( p_B(x) = 5x + 3 \) are assigned to hosts \( A \) and \( B \) respectively. Then the hosts \( A \) and \( B \) calculate their TSV’s as follows.

**A:** \( \text{TSV}_A = [\phi(p_A(0)) \phi(p_A(1)) \phi(p_A(2)) \phi(p_A(3)) \phi(p_A(4)) \phi(p_A(5)) \phi(p_A(6))] \)

\[ = [\phi(2\cdot0+5) \phi(2\cdot1+5) \phi(2\cdot2+5) \phi(2\cdot3+5) \phi(2\cdot4+5) \phi(2\cdot5+5) \phi(2\cdot6+5)] \]

\[ = [\phi(5) \phi(0) \phi(2) \phi(4) \phi(6) \phi(1) \phi(3)] \]

\[ = [0000010 1000000 0000000 0000000 0000000 0100000 0001000] \]

**B:** \( \text{TSV}_B = [\phi(p_B(0)) \phi(p_B(1)) \phi(p_B(2)) \phi(p_B(3)) \phi(p_B(4)) \phi(p_B(5)) \phi(p_B(6))] \)

\[ = [\phi(5\cdot0+3) \phi(5\cdot1+3) \phi(5\cdot2+3) \phi(5\cdot3+3) \phi(5\cdot4+3) \phi(5\cdot5+3) \phi(5\cdot6+3)] \]

\[ = [\phi(3) \phi(1) \phi(6) \phi(4) \phi(2) \phi(0) \phi(5)] \]

\[ = [0001000 0100000 0000000 0000000 0000000 0000000 1000000] \]

Since the degree of \( (p_A - p_B) = (2x+5 - 5x-3) = (-3)x+2 = 4x+2 \) is 1, there is exactly one position at which both TSV\(_A\) and TSV\(_B\) have value 1; this is because \( 4x+2 \) has only one root in GF(7). When \( x = 3 \), \( 4x+2 = 14 = 0; \phi(p_A(3)) = \phi(p_B(3)) \). Thus, there is a collision in the fourth subframe.
Ju and Li’s improved algorithm uses a different frame length. Since $5 > 0.1464 \cdot 35^{\frac{1}{2}}$, the best value of $k$ is 1. Since $35^{\frac{1}{2}} \leq 2 \cdot 5$, the minimum system throughput is maximized when $p = 2k \cdot D_{\text{max}} = 10$. However, $p$ should be a prime, and so $p = 11$. Then, the GRAND algorithm is used to construct TSV’s of length 121.

3. Transmission scheduling using constant weight codes

Since our approach uses constant weight codes to design an efficient scheduling algorithm, let us redefine Problem 1 using coding theory concepts. Before that some definitions are given.

**Definition 3:** The weight of a binary vector $X = (x_1, x_2, \ldots, x_n)$ where $x_i \in \{0,1\}$, denoted by $\omega(X)$, is the number of 1’s in $X$.

**Definition 4:** The Hamming distance between two vectors $X$ and $Y$, denoted by $H_d(X, Y)$, is the number of positions at which $X$ and $Y$ differ.

**Definition 5:** An $(n, d, w)$ constant weight binary code is a set of binary vectors of length $n$, Hamming distance at least $d$ apart, and weight $w$.

**Definition 6:** $A(n, d, w)$ denotes the maximum number of binary vectors of length $n$, Hamming distance at least $d$ apart, and constant weight $w$.

From the above definitions, it can be seen that the problem of transmission scheduling is equivalent to the problem of constructing a constant weight code that satisfies the properties mentioned in Problem 2 below.

**Problem 2:** Construct a $(n, 2(w-k), w)$ binary constant weight code $C$ such that
1. $A(n, d, w) \geq N$.
   
   (Since each host must have a distinct vector, there must be at least $N$ codewords.)
2. $\omega(X) = w$, where $X \in C$. 
(This property guarantees that the fair bandwidth sharing.)

3. For \(X, Y \in C\), \(H_d(X, Y) \geq d = 2(w - k)\), where \(k \geq 0\).

4. \(w > D_{\text{max}} \cdot k\).

(The properties 3 and 4 guarantee that there is at least one collision free slot in each frame)

The proposed algorithm is based on the constant weight code which satisfies the properties mentioned in Problem 2. The new algorithms have similar properties to that of the previously known topology-transparent algorithms such as the GRAND and Ju and Li’s algorithms: topology independence, guaranteed minimum throughput, bounded maximum delay, and fair transmission policy. In the previous topology-transparent algorithms, the length of the frame should be \(q^2\), where \(q\) is a prime or a prime power. In our algorithms, the length of the frame depends on the length of the codewords, \(n\). The only restriction on the length of the frame is that the selected constant weight code should have at least \(N\) codewords; \(i.e., A(n, d, w) \geq N\).

**A. Maximizing the minimum system throughput**

Before describing the new algorithm, let us define the minimum system throughput \((T_{\text{min}})\).

**Definition 7:** The minimum system throughput \((T_{\text{min}})\) is defined as the ratio of the number of guaranteed collision-free slots in a single frame to the length of the frame.

\[
T_{\text{min}} = \frac{\text{the number of guaranteed collision-free slots}}{\text{the length of the frame}} = \frac{(w - D_{\text{max}} \cdot k)}{n}
\]

We can improve the minimum system throughput \((T_{\text{min}})\) by decreasing the length of the frame \((n)\) while preserving the number of guaranteed collision-free slots. The proposed algorithm tries to maximize the minimum system throughput by choosing proper constant weight codes. The algorithm is described below; the first and second steps of our algorithm are based on Ju and Li’s analysis in [10].
Scheduling algorithm A (MaxThrou)

(Input: the number of mobile hosts (N),
    the maximum degree in the network (D_max),
Output: The transmission scheduling vectors.)

1. Set $k$ to 1 if $D_{\text{max}} > 0.1464 N^{1/2}$; Set $k$ to 2, otherwise.
2. Set $w$ to $2k \cdot D_{\text{max}}$ if $N^{1/(k+1)} \leq 2k D_{\text{max}}$; Set $w$ to $N^{1/(k+1)}$, otherwise.
3. Set $d$ to $2 \cdot (w - k)$.
4. By looking up the table of constants weight codes in [13], select the smallest $n$ such that $A(n, d, w) \geq N$.
5. Calculate $T_{\text{min}} = (w - D_{\text{max}} \cdot k) / n$.
6. Using the method denoted in [13], construct a $(n, d, w)$ binary constant weight code.
7. Assign a unique constant weight code to each host.

B. Minimizing the nodal delay bound

In this subsection, we focus on designing a transmission scheduling which minimizes the nodal delay bound. The nodal delay is the amount of time between when a packet is ready and when it is transmitted. To reduce the nodal delay, the proposed algorithm chooses a constant weight code such that $n$ is as small as possible.

Scheduling algorithm B (MinDelay)

(Input: the number of mobile hosts (N),
    the maximum degree in the network (D_max),
Output: The transmission scheduling vectors.)

1. Set $w$ to $D_{\text{max}} + 1$.
2. Set $d$ to $2 \cdot (w - 1)$.
3. By looking up the table of constants weight codes in [13], select the smallest $n$ such that $A(n, d, w) \geq N$.
4. Using the method denoted in [13], construct a $(n, d, w)$ binary constant weight code.
5. Assign a unique constant weight code to each host.
The fundamental difference between the proposed algorithm and the GRAND algorithm is the length of the frame. In GRAND algorithm, the frame length should be $q^2$, where $q$ is a prime or a prime power. But our algorithm can flexibly select the frame length based on the number of mobile hosts ($N$). Thus, the proposed algorithm can minimize the frame length while preserving the number of collision-free slots. This minimization of the frame length results in the improved minimum system throughput.

**Example 2:** Assume there are 12 mobile hosts in the network, and the maximum degree in the network is 3.

The *MaxThrou* scheduling algorithm performs the following steps.

1. Since $D_{\text{max}} > 0.1464 N^{\frac{1}{2}}$, set $k$ to 1.
2. Since $N^{\frac{k}{k+1}} = 12^{\frac{1}{2}} \leq 2D_{\text{max}}$, $w = 2k \cdot D_{\text{max}} = 2 \cdot 1 \cdot 3 = 6$.
3. $d = 2 \cdot (w - k) = 2 \cdot (6 - 1) = 10$
4. By looking up the table in [13], select the proper $n$. Since $A(26, 10, 6) = 13$, $n = 26$.
5. Calculate the $T_{\text{min}} = (w - D_{\text{max}} \cdot k) / n = (6 - 3 \cdot 1) / 26 = 3 / 26 = 0.115$.
6. Using the method denoted in [13], construct a constant weight code of length 26, distance 10 and weight 6:

   0 : 0000000011001000010100001
   1 : 0000000110010000101000010
   2 : 0000000110010000101000010
   3 : 0000001100100000101000010
   4 : 0000110010000010100001000
   5 : 0001100100000010100001000
   6 : 0011001000000010000100001
   7 : 0110010000000010000100001
   8 : 1100100000000001000010000
   9 : 10010000000001000100001001
  10 : 0100000000011001000010000
  11 : 0100000000011001000010000
  12 : 100000000011001000001010000
7. Assign a unique codeword to each mobile host. Since there are 13 codewords, we can randomly select any 12 codewords.

The steps of the \textit{MinDelay} scheduling algorithm is shown below.
1. \( w = D_{\text{max}} + 1 = 3 + 1 = 4. \)
2. \( d = 2 \cdot (w - 1) = 2 \cdot (4 - 1) = 6. \)
3. By looking up the table in [13], select the smallest \( n \) such that \( A(n, 6, 4) \geq 12 \). Since \( A(13, 6, 4) = 13 \), \( n = 13 \).
4. Using the method denoted in [13], construct a constant weight code of length 13, distance 6 and weight 4;

\[
\begin{align*}
0 & : 0000010110001 \\
1 & : 0000101100010 \\
2 & : 0001011000100 \\
3 & : 0010110001000 \\
4 & : 0101100010000 \\
5 & : 1011000100000 \\
6 & : 0110001000001 \\
7 & : 1100010000010 \\
8 & : 1000100000101 \\
9 & : 0001000001011 \\
10 & : 0010000101100 \\
11 & : 0100001011000 \\
12 & : 1000001011000
\end{align*}
\]
5. Assign a unique codeword to each mobile host.

Sometimes, the selected constant weight code may not maximize the minimum system throughput. For example, if we use a (34, 12, 7) constant weight code, then the \( T_{\text{min}} \) is equal to 0.118; since \( A(34, 12, 7) = 12 \), we can also use this code. This code is slightly better than the (26, 10, 6) code in terms of the minimum throughput. Thus, we might want to check the minimum throughputs of other constant weight codes whose weights are
close to the current code weight. To do this, we may increase or decrease the weight by 1, and perform the Step 3 to Step 5 of the proposed algorithm until $T_{min}$ is not improved.

**Example 3:** In this example, we compare the minimum throughput of the scheduling algorithm $A$ to that of previously known algorithms (the conventional TDMA, Chlamtac and Farago’s GRAND algorithm [4], and Ju and Li’s algorithm [10]). Assume there are 12 mobile hosts in the network, and the maximum degree in the network is 3.

- For the conventional TDMA, unique collision-free slots are assigned to each mobile. Thus, the minimum throughput ($T_{min}$) is $1/12 = 0.083$.
- For the GRAND algorithm, there are 16 slots in each frame, and every host has at least one collision-free slot. Thus, the minimum throughput ($T_{min}$) is $1/16 = 0.062$.
- For the Ju and Li’s algorithm, there are 49 slots in each frame, and every host has at least 4 collision-free slots. Thus, the minimum throughput ($T_{min}$) is $4/49 = 0.082$.
- In the MaxThrou scheduling algorithm, there are 26 slots (or 34 slots) in each frame, and every host has at least 3 (or 4) collision-free slots. Thus, the minimum throughput using the proposed algorithm is $3/26$ (or $4/34) = 0.115$(or 0.118).

**C. Theoretical bounds**

In the previous example, the minimum throughput ($T_{min}$) of our algorithm is much better than that of the previous known algorithms. Now we’ll show that our algorithm is better than the GRAND algorithm in almost all cases (Since Ju and Li’s algorithm uses the GRAND algorithm to construct TSV’s, we only consider the GRAND algorithm for comparison.). In order to do this, we first describe some properties of $A(n, d, w)$, and then construct a new class of efficient $(n, d, w)$ codes.

The upper and lower bounds on $A(n, d, w)$ have been extensively studied for almost 40 years by coding theory researchers[1][3][9][13]. Although none of them found the general answer for these bounds, various upper and lower bounds have been known so far. The following theorem is due to S. M. Johnson.
**Theorem 1** (Johnson’s bound):

\[ A(n, d, w) \leq \left\lfloor \frac{n}{n - w} \right\rfloor \cdot A(n - 1, d, w) \]

Proof: The proof is in [9].

From Johnson’s upper bound we can get a lower bound of \( A(n - 1, d, w) \).

\[ A(n - 1, d, w) \geq \left\lfloor \left( \frac{n - w}{n} \right) \cdot A(n, d, w) \right\rfloor \quad (1) \]

Since Ju and Li’s analysis shows that the best value of \( k \) is mostly 1 unless the network size is extremely large, the proper value of \( k \) is usually 1 except for some extremely sparse and large networks. Therefore, we consider only the case \( k = 1 \).

Let \( S \) be the set of TSV’s obtained by the GRAND algorithm. A TSV in \( S \) can be divided into \( q \) subframes, and has exactly one transmission slot in each subframe. Thus, we can add the following \( q \) additional TSV’s \( (TSV_1, TSV_2, TSV_3, \ldots, TSV_q) \) to \( S \) without changing the properties of \( S \).

\[
\begin{align*}
TSV_1 &= (111\ldots1 000\ldots0 000\ldots0 \ldots 000\ldots0) \\
TSV_2 &= (000\ldots0 111\ldots1 000\ldots0 \ldots 000\ldots0) \\
TSV_3 &= (000\ldots0 000\ldots0 111\ldots1 \ldots 000\ldots0) \\
&\quad \quad \vdots \\
TSV_q &= (000\ldots0 000\ldots0 000\ldots0 \ldots 111\ldots1)
\end{align*}
\]

This set \( S \) of the scheduling vectors forms a \((q^2, 2(q - 1), q)\) binary constant weight code, and the number of codewords is \( q^2 + q \), where \( q \) is a prime or a prime power.

Thus, \( A(q^2, 2(q - 1), q) \geq q^2 + q \).

By (1),

\[
A(q^2 - 1, 2(q - 1), q) \geq \left\lfloor \frac{(q^2 - q)}{q^2} \cdot A(q^2, 2(q-1), q) \right\rfloor \\
\geq \left\lfloor \frac{(q^2 - q)}{q^2} \cdot (q^2 + q) \right\rfloor \\
= \left\lfloor \frac{(q^2 - q) \cdot (q^2 + q)}{q^2} \right\rfloor \\
= \left\lfloor \frac{q^2((q - 1) \cdot (q + 1))}{q^2} \right\rfloor
\]

* This lower bound is shown in [3]
\[(q - 1) \cdot (q + 1) = q^2 - 1\]

Therefore, if the number of mobile hosts is less than \(q^2\), then we can reduce the frame size by at least 1. Since the frame size is reduced while preserving other properties, it is easy to see that the minimum system throughput \(T_{\text{min}}\) of the proposed algorithm is better than that of the GRAND algorithm. Regarding the delay bound of the scheduling algorithm B, the same argument can be applied.

### 4. Simulation study

In the previous section, we showed that the proposed algorithms outperform the previously known algorithms in the worst situations. However, in designing a realistic network system, the system designer should consider not only the guaranteed minimum system throughput but also the average system throughput. Thus, we discuss the average system throughput in this section.

In order to compare the average system throughput of the proposed algorithms to that of previously known algorithms, the GloMoSim library [15] is used for the simulation. Since the implementation of the GRAND algorithm is freely available in the library, this implementation is used without any modification.

#### A. Simulation model

Since it is extremely difficult to maintain the bounded degree of a mobile network using random mobility models, we assume that the nodes are static (However, in a realistic network system, a topology control technique such as [14] can be used to maintain the bounded degree of a mobile network.). We consider one type of network topology as shown in Fig. 1; there are 12 nodes in the network and the maximum degree is 3. A unique transmission schedule generated by the different scheduling algorithms as shown in Example 3, is randomly assigned to each host. The transmit and receive characteristics
of the radio model are similar to Lucent’s WaveLAN network interface. We assume radio propagation range for each mobile is 200 m, and the channel capacity is 2 Mbits/sec. The source sends four data packets per second, and the size of the data packet is 512 bytes. To evaluate the relation between the data traffic load and the system throughput, we increase the number of connections up to 5; the source and destination pairs used in the simulation are also shown in Fig. 1. Each simulation is run for 600 seconds of simulation time to measure the average system throughput.

![Simulated network topology](image)

**Fig. 1. Simulated network topology**

**B. Simulation results and discussion**

Since each node has two or three neighbors, there will be one or two collisions in each frame for the low or moderate data traffic load. Similarly, there will be two or three collisions for the heavy data traffic load. As the traffic load increases, the number of collisions will also increases. The maximum number of collisions can be up to two or three depending on the number of neighbors. We estimate the system throughputs before the simulation. These estimations are shown in Table 1. According to our estimation, we expect that the system throughput will be MinDelay > MaxThrou > GRAND > Ju and Li’s > TDMA for the moderate data traffic load, and MaxThrou > MinDelay > GRAND > Ju and Li’s > TDMA for the heavy data traffic load.
Table 1. The estimated system throughputs

<table>
<thead>
<tr>
<th>Scheduling algorithm</th>
<th>Moderate load (50%: 1 collision, 50%: 2 collisions)</th>
<th>Heavy load (50%: two collisions, 50%: three collisions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDMA</td>
<td>$(1/12 + 1/12)/2 = 0.083$</td>
<td>$(1/12 + 1/12)/2 = 0.083$</td>
</tr>
<tr>
<td>GRAND</td>
<td>$(3/16 + 2/16)/2 = 0.156$</td>
<td>$(2/16 + 1/16)/2 = 0.094$</td>
</tr>
<tr>
<td>Ju and Li’s</td>
<td>$(6/49 + 5/49)/2 = 0.112$</td>
<td>$(5/49 + 4/49)/2 = 0.092$</td>
</tr>
<tr>
<td>MaxThrou</td>
<td>$(5/26 + 4/26)/2 = 0.173$</td>
<td>$(4/26 + 3/26)/2 = 0.135$</td>
</tr>
<tr>
<td>MinDelay</td>
<td>$(3/13 + 2/13)/2 = 0.193$</td>
<td>$(2/13 + 1/13)/2 = 0.115$</td>
</tr>
</tbody>
</table>

We run each simulation 10 times and measure the mean system throughput of the mobile network. Table 2 and Fig. 2 display the mean system throughput as a function of the offered traffic load. The simulation results are very similar to our expectation for the moderate data load (up to 3 FTP connections). Although Ju and Li’s algorithm guarantees better than the GRAND algorithm in terms of minimum system throughput, the mean system throughput of their algorithm was poor. This is because their algorithm has a low channel utilization rate. For example, in the simulated case, Ju and Li’s algorithm uses only 7 slots among 49 slots (14%) while the GRAND algorithm uses 4 slots among 16 slots (25%). However, Ju and Li’s algorithm outperforms the GRAND algorithm when the system is heavily loaded (5 FTP applications). This is because, in the heavy load situation, the average system throughput is very close to the worst system throughput. Based on the simulation results, we conclude that MaxThrou and MinDelay algorithms outperform the previously known algorithms in terms of mean system throughput.

Table 2. Summary of simulation results (mean system throughputs: bps)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Data Traffic load (The number of FTP connections)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>TDMA</td>
<td>20504</td>
</tr>
<tr>
<td>GRAND</td>
<td>29391</td>
</tr>
<tr>
<td>Ju and Li’s</td>
<td>25899</td>
</tr>
<tr>
<td>MaxThrou</td>
<td>36123</td>
</tr>
<tr>
<td>MinDelay</td>
<td>44344</td>
</tr>
</tbody>
</table>
5. Conclusions

In this paper, we showed that the topology-transparent transmission scheduling problem is equivalent to the problem of constructing constant weight codes which satisfy certain properties. Constructing constants weight codes and finding the upper and lower bounds on constant weight codes are extensively studied for almost four decades. Thus, we can use the previously known results to find the proper scheduling for the given networks. Further, we have given some improved constant weight codes; when these codes are used for transmission scheduling, we get better minimum system throughput while at the same time preserving other properties such as topology independence, guaranteed minimum throughput, bounded maximum delay, and fair transmission policy. In the simulation study, we measure the average system throughput of transmission scheduling algorithms. The simulation results show the proposed algorithms outperform the previously known algorithms in terms of mean system throughput.
References


