MIMO 链路 ad hoc 网络传输调度算法

陈丹*, 李建东, 李长乐

西安电子科技大学综合业务网络理论及关键技术国家重点实验室, 西安 710071

* 通信作者. E-mail: dan_chen@tom.com

收稿日期: 2010-10-09; 接受日期: 2011-05-04

国家杰出青年科学基金 (批准号: 60725105)、国家重点基础研究发展计划 (批准号: 2009CB320404)、长江学者和创新团队发展计划 (批准号: IRT0852)、国家高技术研究发展研究计划 (批准号: 2007AA01Z217)、国家自然科学基金 (批准号: 60702057, 60972047, 60972048) 和高等学校学科创新引智计划 (批准号: B08038) 资助项目

摘要 针对 MIMO 链路 ad hoc 网络中已有传输调度算法未充分利用 MIMO 空间复用能力的问题, 提出了一种新的传输调度算法. 该算法通过考虑 MIMO 的多流接收能力而有效地减小了调度帧长, 通过合理安排节点分配顺序使得节点分配的数据流比较平均, 同时, 该算法通过允许节点发送多个数据流而高效利用了 MIMO 的空间复用能力. 推导分析了传输调度算法的通过量和时延性能, 结果表明, 与已有算法相比, 采用文中算法可以提高网络的通过量, 减小网络的平均时延和最大时延.

关键词 多输入多输出 空间复用 多流接收 ad hoc 网络 传输调度

1 引言

Ad hoc 网络是一种由若干移动节点构成的无线网络, 它不需要固定基础设施的支持并且布设简单, 主要可用于抢险救灾、军事行动、应急任务等场合中. 由于无线信道的开放特性和节点的传输功率有限, 仅位于某个发送节点传输范围内的节点才能够接收到该发送节点传输的数据, 而其他节点则不能接收到. 因此网络中各节点观察到的信道忙闲状态并不相同, 这会引发诸如隐藏终端等问题的发生, 从而导致通信因发生冲突而失败. 节点传输调度通过合理地给每个节点分配传输时隙来避免冲突的发生, 可以有效提高网络的通过量和减小分组时延. Ad hoc 网络的节点传输调度算法已经获得了广泛地研究, 如文献 [1~6]. 文献 [1,2] 中称其提出的调度策略为“广播调度”, 但它们并不是指将分组传播到网络中所有节点, 而仅指发送节点向其所有邻节点发送数据的情况, 因此应称为节点传输调度 [3]. 其中文献 [1] 证明了能够获得最优通过量的传输调度问题是一个 NP 完全问题, 并提出通过为每个节点分配不同的时隙来避免冲突, 然而这样得到的调度帧长较长, 新到达分组等待传输的时间就比较长, 因此网络时延比较大. 文献 [2,3] 考虑了地域空间复用 (spatial reuse), 仅为两跳内的节点分配不同的时隙, 因此可以获得比文献 [1] 更短的帧长, 减小了网络时延. 文献 [4~6] 提出的调度算法通过自适应地调节时隙的分配来适应网络拓扑的变化, 并且时隙的调整仅在有限的范围内进行, 因此其开销较小. 由文献 [2~6] 的分析可知, 单天线节点组成的网络的调度帧长需要大于任一网络节点的两跳以内节点数的最大值.

引用格式: 陈丹, 李建东, 李长乐. MIMO 链路 ad hoc 网络传输调度算法. 中国科学: 信息科学, 2012, 42: 161-170

近来,作为通信新技术之一的多输入多输出 (MIMO) 技术也被应用于 ad hoc 网络中用以改善网络的性能,如减小接收分组的时延抖动和提高通过量^[7,8]. 由于 MIMO 技术为网络节点带来了空间自由度,使得节点能够发送或接收多个数据流,因此只要相距两跳的节点发送的数据流数不超过接收节点的接收能力,它们就可以在同一时隙传输. 这样在为网络节点分配时隙时,可以根据各接收节点的接收能力来为相距两跳的节点分配相同的时隙,从而减小调度帧长,降低网络时延. 同时,由于发送节点可以传输多个数据流,可以在考虑周围节点接收能力的情况下尽量为其分配更多的数据流,以提高网络的通过量. 文献 [9~13] 提出了 MIMO 链路 ad hoc 网络传输调度算法. 其中文献 [9] 利用图论中的方法来设计调度算法,需要找到竞争图中的最大团,复杂度比较高,且该文提出的流控增益 (stream control gains) 目前并没有定量的研究,因此无法准确衡量其性能. 文献 [10] 中的拓扑未知调度算法能够保证网络节点在一帧中少数几个时隙可以无冲突传输,然而由于发生冲突的时隙比较多,造成了时隙的浪费,网络通过量也比较小. 文献 [11,12] 中假设节点发送控制分组 (RTS 和 CTS) 时使用掩码,认为控制分组的传输不会发生冲突,然而这依赖于掩码的数量和同时传输的节点数量,因此在实际网络中使用这两种调度算法仍会导致冲突的发生,造成时隙的浪费. 文献 [13] 中提出的传输调度算法是集中式算法,虽然其调度结果没有冲突,但由于该算法前几步中没有考虑 MIMO 的多流接收能力,没有获得最短帧长,同时,由于不允许节点在一个时隙内发送多个数据流,网络的通过量性能也受到了影响. 本文研究的传输调度问题为: 发送节点向其所有一跳邻节点发送数据,并保证所有一跳邻节点都能正确接收. 针对 MIMO 链路 ad hoc 网络中的此类传输调度问题,本文提出了一种新的集中式传输调度算法,该算法通过考虑 MIMO 的多流接收能力而有效地减小了调度帧长,通过合理安排节点分配顺序使得节点分配的数据流比较平均,同时,该算法通过允许节点发送多个数据流而高效利用了 MIMO 的空间复用 (spatial multiplexing) 能力. 本文推导分析了传输调度算法的通过量和时延性能,结果表明,与已有算法相比,采用本文算法可以提高网络的通过量,减小网络的平均时延和最大时延.

2 MIMO 模型和网络模型

2.1 MIMO 模型

MIMO 系统在发送端和接收端均配备了多个天线,通过发送天线阵列和接收天线阵列联合来改善传输性能. MIMO 链路可以提供 3 种增益: 阵列增益、分集增益和空间复用增益. 由于空间复用增益可以提供较高的数据传输速率,本文仅考虑利用 MIMO 的空间复用增益. 在多径较为丰富的情况下, MIMO 系统收发两端的多个天线能够提供多个独立的信道,通过在这些独立的信道上发送独立的数据流可以实现空间复用,并且不需要增加额外的功率. 接收端根据各发送数据流的不同空间特征可以成功将它们分离. 通常,如果多径足够丰富,具有 M 个天线的接收机可以成功地接收不多于 M 个数据流. 在 ad hoc 网络中,相互干扰的链路通过控制各自发送的数据流数还可以获得流控增益^[9],然而鉴于目前并没有定量的研究,本文将不考虑流控增益.

2.2 网络模型

考虑链路对称的 Ad hoc 网络,则它可以用无向图 $G(V, E)$ 来表示,其中 V 表示网络中节点的集合, E 表示网络链路的集合,当且仅当节点 $u, v (\forall u, v \in V)$ 处于彼此传输范围之内时,有 $(u, v) \in E$. 对于任一节点 $u \in V$, 令 N_u 和 N_u^2 分别表示其一跳邻节点和两跳邻节点的集合,有 $N_u = \{w | (w, u) \in E\}$.

$E, \forall w \in V\}$, $N_u^2 = \{n | n \in \bigcup_{v \in N_u} N_v, n \notin N_u \cup u\}$. 定义节点 u 的度 $\deg(u)$ 为 u 的一跳邻节点的个数, 即 $\deg(u) = |N_u|$.

我们假设冲突是造成传输失败的唯一原因, 这里的冲突包含两种类型. 第 1 类冲突是节点 $u (\forall u \in V)$ 和其邻节点同时发送数据所造成的冲突. 由于无线收发机不能同时收发, 当 u 发送数据时, 如果其邻节点 $v (v \in N_u)$ 也同时发送, 则 v 不能接收 u 发送的数据, 导致 v 处发生第 1 类冲突. 第 2 类冲突是 u 不发送而其所有一跳邻节点共发送了多于 M 个数据流时造成的冲突. 由于节点 u 至多可以接收 M 个数据流, 如果其一跳节点共发送了多于 M 个数据流, 则 u 不能正确接收任何一个数据流, 导致 u 处发生第 2 类冲突.

3 传输调度算法

传输调度算法的目标是: 在无冲突传输的限制下, 为每个节点分配尽可能多的时隙, 并使帧长最短 [3,13]. 基于这样的目标, 本文提出了一种新的支持 MIMO 的 ad hoc 网络节点传输调度算法, 通过充分利用 MIMO 的空间复用能力来改善网络性能. 算法分为 3 个阶段:

(1) 在考虑 MIMO 空间复用的条件下为每个节点分配一个时隙, 从而获得最小帧长, 此帧长在后两个阶段将不再改变. 与单天线的节点不同, MIMO 技术为网络节点带来了空间自由度, 使得节点能够接收多个数据流, 因此在不发生第 2 类冲突的前提下, 相距两跳的节点就可以分配相同的时隙, 这样就可以减小调度帧长.

(2) 尽量为每个节点分配更多的时隙, 但此阶段中对于已分配给节点 u 的时隙不会再重复分配给 u , 即此阶段不允许节点在单个时隙内分配多个数据流. 文献 [3] 中指出, 减小调度帧长和使节点传输时隙比较均匀的分布在一帧内有利于减小分组调度时延, 而此阶段不为节点重复分配已分配时隙正是为了使节点的传输时隙在一帧内的分布比较均匀.

(3) 在不引起第 2 类冲突的情况下增加节点在已分配时隙内传输的数据流数, 从而提高网络的通过量.

在调度算法执行的过程中, 当某个节点 u 有多个时隙可以分配时, 需要按照一定的规则来选择一个时隙分配给 u . 文献 [3,13] 中提出了剩余时隙自由度函数来衡量节点 u 分配时隙 x 后对其一跳和两跳节点剩余时隙自由度的影响, 这里剩余时隙自由度指节点还可以分配的时隙数. 受剩余时隙自由度函数的启发, 同时考虑到 MIMO 技术带来的空间自由度, 本文定义剩余空-时自由度函数 $G(u, x)$ 来衡量为节点分配某时隙后对其一跳和两跳节点的空间和时间自由度的影响, 这里空间自由度指节点在某时隙内还可以增加的数据流数, 时间自由度指空间自由度大于 0 的时隙数, $G(u, x)$ 的表达式为

$$G(u, x) = \sum_{v \in N_u \cup N_u^2} G^v(u, x), \quad (1)$$

其中

$$G^v(u, x) = D_T^v \times D_S^v, \quad (2)$$

$$D_T^v = [|N_u| + |N_u^2|]^{-i^v(u, x)}, \quad (3)$$

$$D_S^v = \sum_{j=0}^M k_j^v(u, x) l^{-(j+1)}. \quad (4)$$

上面的式子中, l 表示算法执行到当前时刻的帧长, $k_j^v(u, x)$ 表示 u 选择时隙 x 以后对 v 而言 l 个时隙中空间自由度为 j 的时隙数, $|N_u|$ 和 $|N_u^2|$ 分别表示节点 u 的一跳邻节点和两跳邻节点的个数, $i^v(u, x)$ 表示 u 选择时隙 x 以后节点 v 的可选时隙数 (即空间自由度不为 0 的时隙, 包含已为 v 分配但当前空间自由度不为 0 的时隙), D_T^v 和 D_S^v 分别为节点 u 选择时隙 x 后节点 v 的剩余时间自由度函数和剩余空间自由度函数. 由 (1) 式可以看到, $G(u, x)$ 越小, 节点 u 的一跳和两跳节点可分配的时隙和数据流越多, 因此当节点 u 有多个时隙可分配时, 应当选择 $x^* = \arg \min_x G(u, x)$.

由于一跳和两跳邻节点越多的节点在分配时隙时受到的限制越大, 故需要优先为其分配时隙; 然而, 某节点 u 的两跳邻节点和一跳邻节点分配时隙 i 时对节点 u 的空间自由度的影响却不相同. 若为 u 的一跳邻节点分配时隙 i , 则为了避免第一类冲突的发生, 节点 u 不能分配时隙 i , 因此可以认为 u 的一跳邻节点分配时隙时对 u 空间自由度的影响为 M (M 为天线数); 若为 u 的两跳邻节点分配时隙 i , 则 u 仍可以分配时隙 i , 只是能够发送的数据流减少了 1 个, 所以 u 的两跳邻节点分配时隙时对 u 空间自由度的影响为 1. 考虑上述两点, 我们将网络节点按照如下规则分组排序以确定节点分配顺序: 首先计算 $\deg_2(u) = |N_u| + |N_u^2|/M, \forall u \in V$, 将 $\deg_2$ 值相同的节点分在一组, 各组之间按照 $\deg_2$ 由大到小的顺序排列; 每个组内节点再按照 $\deg$ 的大小重新分组并排序, $\deg$ 相同的节点分在一组, 这里 $\deg$ 为第 2 节中定义的节点的度. 最后得到的分组排序 $\{S_k\} (k = 1, 2, \dots, K)$ 满足

- (1) $\sum_{k=1}^K |S_k| = |V|$;
- (2) $\forall u, v \in S_k, \deg_2(u) = \deg_2(v)$ and $\deg(u) = \deg(v)$;
- (3) $\forall u \in S_k, w \in S_{k+1}, \deg_2(u) > \deg_2(w)$ or $(\deg_2(u) = \deg_2(w) \text{ and } \deg(u) > \deg(w))$,

其中 $|V|$ 表示网络节点数, and 和 or 分别表示“与”和“或”. 分配时隙时按组进行分配, 对于同一组内的节点需要考虑其时间和空间自由度的不同, 因此类似 (2) 式定义节点的空-时自由度函数为

$$G^v = \sum_{j=0}^M k_j^v l^{-(j+1)} \times [|N_v| + |N_v^2|]^{-i^v}, \quad (5)$$

其中 l 表示算法执行到当前时刻的帧长, i^v 表示算法执行到当前时刻节点 v 还可以选择时隙数, k_j^v 表示对 v 而言 l 个时隙中空间自由度为 j 的时隙数, $|N_v|$ 和 $|N_v^2|$ 分别表示节点 v 的一跳邻节点和两跳邻节点的个数, G^v 越大的节点空-时自由度越小, 因此应当优先分配. 下面详细描述算法:

第 1 阶段: 首先为 S_1 中的任一节点 n 和 n 的一跳邻节点分配时隙, 接着为网络中其他节点分配时隙, 此阶段每个节点分配一个时隙, 步骤如下:

- (1) 将节点按照前述规则分组排序, 得到 $\{S_k\} (k = 1, 2, \dots, K)$, 令 $V_{\text{temp}} = V$;
- (2) 令 $l = 1$, 任意选择 $n \in S_1$, 将 l 分配给 n , 将 n 从 V_{temp} 中删去, 即 $V_{\text{temp}} = V_{\text{temp}} \setminus n$;
- (3) 令 $S = N_n$;
- (4) $k := 1$;
- (5) $S_{\text{temp}} = S \cap S_k$, 如果 $S_{\text{temp}} = \emptyset$, 转到 (8);
- (6) $w = \arg \max_{v \in S_{\text{temp}}} G^v$, 为 w 分配时隙. 如果没有时隙可分配, $l = l + 1$, 将 l 分配给 w ; 如果有 1 个时隙可分配, 则将该时隙分配给 w ; 如果有多个时隙可分配, 选择时隙 $x^* = \arg \min_x G(w, x)$ 分配给 w ;
- (7) $V_{\text{temp}} = V_{\text{temp}} \setminus w, S_{\text{temp}} = S_{\text{temp}} \setminus w$, 如果 $S_{\text{temp}} \neq \emptyset$, 转到 (6);
- (8) 如果 $k < K, k := k + 1$, 转到 (5);
- (9) $S = V_{\text{temp}}$, 如果 $S \neq \emptyset$, 转到 (4).

第 2 阶段: 再次为节点分配时隙以增加时隙复用和 MIMO 的空间复用, 此阶段不允许节点在同一时隙分配多个数据流, 步骤为:

(10) $k := 1$;

(11) $S_{\text{temp}} = S_k$;

(12) $w = \arg \max_{v \in S_{\text{temp}}} G^v$, 为 w 分配时隙. 如果没有时隙可分配, 这个阶段不再为 w 分配时隙; 如果一个时隙可分配, 则将该时隙分配给 w ; 如果有多个时隙可分配, 选择时隙 $x^* = \arg \min_x G(w, x)$ 分配给 w ;

(13) $S_{\text{temp}} = S_{\text{temp}} \setminus w$, 如果 $S_{\text{temp}} \neq \emptyset$, 转到 (12);

(14) 如果 $k < K$, $k := k + 1$, 转到 (11);

(15) 如果再没有任何节点可以分配时隙, 则此阶段结束, 否则转到 (10).

第 3 阶段: 为节点重复分配已分配的时隙, 以增加节点在分配时隙内传输的数据流, 从而充分利用 MIMO 的空间复用能力. 由于此阶段是增加已分配时隙内的数据流, 因此仅影响两跳节点空间自由度, 故节点按照 $|N_u^2|$ 由小到大分组排序以确定节点分配顺序, 此阶段时隙分配步骤为:

(16) 将节点按 $|N_u^2|$ 由小到大分组顺序, 得到的 $\{P_k\}(k = 1, 2, \dots, K')$ 满足 $\forall u, v \in P_k, |N_u^2| = |N_v^2|$ 和 $\forall u \in P_k, w \in P_{k+1}, |N_u^2| < |N_w^2|$;

(17) $k := 1$;

(18) $S_{\text{temp}} = P_k$;

(19) $w = \arg \max_{v \in S_{\text{temp}}} G^v$, 为 w 分配时隙. 如果没有时隙可分配, 不再为 w 分配时隙; 如果有一个时隙可分配, 则将该时隙分配给 w ; 如果有多个时隙可分配, 选择时隙 $x^* = \arg \min_x G(w, x)$ 分配给 w ;

(20) $S_{\text{temp}} = S_{\text{temp}} \setminus w$, 如果 $S_{\text{temp}} \neq \emptyset$, 转到 (19);

(21) 如果 $k < K'$, $k := k + 1$, 转到 (18);

(22) 如果再没有任何节点可以分配时隙, 则算法结束, 否则转到 (17).

下面用图 1 的例子来说明本文算法, 假设节点天线数 $M = 3$. 首先将节点分组排序得到 $S_1 = \{B, D\}$, $S_2 = \{A, E, C\}$. 为 S_1 中的节点 B 分配时隙 1, 接着为 B 的一跳邻节点 A, C, D 分配时隙. 由于 $D \in S_1$, 优先分配. 而当前 $l = 1$ 个时隙中没有时隙可以给 D 分配, 因此 $l = l + 1$, 将时隙 2 分配给 D . 接下来考虑节点 A 和 C , 由于 $G^C = 1 > G^A = 5/32$, 先为 C 分配时隙 3, 再为 A 分配时隙. 由于 A 有时隙 2 和 3 可以选择, 需要比较 $G(A, 2)$ 和 $G(A, 3)$ 的大小. 由于 $G(A, 2) = 247/432 < G(A, 3) = 312/432$, 为 A 分配时隙 2. 至此 B 的一跳邻节点分配完毕, 开始考虑剩下的节点 E . E 可以选择时隙 1 和 3, $G(E, 1) = 257/432 < G(E, 3) = 328/432$, 为 E 分配时隙 1, 第 1 阶段分配结束. 第 2 阶段 A 和 E 有新的时隙 3 可以分配, 比较 G^A 和 G^E 的大小有 $G^A = G^E = 13/432$, 可任选 A 和 E 中的一个, 这里将时隙 3 分配给节点 A . 第 3 阶段节点分组排序为 $P_1 = \{B, D\}$, $P_2 = \{A, E, C\}$, 优先考虑 P_1 中节点 B 和 D , 由于 $G^B = G^D = 7/36$, 分配顺序可任选, 为节点 B 分配时隙 1, 为节点 D 分配时隙 2. 接着考虑 P_2 中的节点, 由于仅节点 A 和 C 可以分配时隙 3, 可以任选其一, 这里为 C 分配时隙 3. 至此其他节点再没有时隙可分配, 算法结束. 最终分配结果如表 1, 其中 T 和 T^2 分别表示节点在时隙内发送 1 个和 2 个数据流.

图 1 网络 1

Figure 1 Network 1

表 1 网络 1 时隙分配结果

Table 1 Slot allocation result for network 1

slot	node				
	A	B	C	D	E
1		T^2			T
2	T			T^2	
3	T		T^2		

4 性能分析

由于文献 [3,13] 中的性能评估方法仅适用于一个时隙内发送一个数据流的情况, 而本文算法允许节点在一个时隙内发送多个数据流, 故文献 [3,13] 中的性能评估方法将不再适用. 为此我们提出了新的方法来衡量调度算法的性能, 而衡量性能的指标仍采用文献 [3,13] 中使用的网络通过量和时延.

设调度算法得到的帧长为 L , 调度的结果可以用 $|V| \times L$ 的矩阵 $\mathbf{T}$ 来表示, 其元素 t_{ij} 表示节点 i 在时隙 j 发送的数据流数, 因此节点 i 的通过量 $C(i)$ 为

$$C(i) = \sum_{j=1}^L t_{ij} / L. \quad (6)$$

整个网络的通过量 C 为

$$C = \sum_{i=1}^{|V|} C(i). \quad (7)$$

下面仅分析重负载情况下网络的时延, 这里的时延指分组的排队时延, 单位为时隙. 由于我们的算法允许节点在一个时隙内发送多个数据流, 而每个节点在每个时隙内最多可以发送 M 个数据流, 因此我们假设网络节点的缓冲队列长度 $Q \geq M$. 在重负载条件下, 节点 i 在时隙 j 发送以后, $j \bmod L + 1$ 时隙开始时刻 i 上层到达的分组会将缓冲队列填满, 这里 j 对 L 求模是因为调度算法执行完成后调度结果是每帧重复的. $j \bmod L + 1$ 时隙开始时刻到达的分组在缓冲队列中的位置为 $(Q - t_{ij} + 1, Q - t_{ij} + 2, \dots, Q)$, 对于位于 $p (p \in (Q - t_{ij} + 1, Q - t_{ij} + 2, \dots, Q))$ 位置的分组, 其时延 $d_{i,j \bmod L+1,p}$ 为

$$d_{i,j \bmod L+1,p} = t^* - j, \quad (8)$$

其中

$$t^* = \min_{t \in \mathbb{N}, t \geq j} t, \quad (9)$$

$$\text{s. t. } p - \sum_{m=j}^t t_{i,m \bmod L+1} \leq 0. \quad (10)$$

(8) 和 (9) 式中的 t^* 表示以时隙 j 所在帧的起点为起点, p 位置的分组被发送的时刻. 节点 i 的平均分组时延为

$$D(i) = \sum_{j \in X_i} \sum_{p=M-t_{ij}+1}^M d_{i,j \bmod L+1,p} / \sum_{j=1}^L t_{ij}, \quad (11)$$

图 2 网络 2
Figure 2 Network 2

图 3 网络 3
Figure 3 Network 3

图 4 两种调度算法下网络通过量的比较

Figure 4 Network throughput under the two algorithms. (a) Network 1; (b) network 2; (c) network 3

其中 $X_i = \{q | t_{iq} \geq 0, \forall q \in (1, 2, \dots, L)\}$. 与文献 [3] 类似, 我们将用 $D(i) (i = 1, 2, \dots, |V|)$ 的平均值和最大值, 即

$$D_{av} = \frac{1}{|V|} \sum_{i=1}^{|V|} D(i), \quad (12)$$

$$D_{max} = \max_i D(i) \quad (13)$$

来衡量不同调度算法下网络的时延性能, 并分别称 D_{av} 和 D_{max} 为网络平均时延和最大时延.

下面比较本文和文献 [13] 中传输调度算法的性能, 这里先采用图 1~3 所示的网络来测试两种算法的通过量和时延性能, 然后再考虑更一般的场景. 为了表示方便, 下文中简称文献 [13] 中的算法为 Zhang 算法. 图 4 中比较了两种调度算法下各网络的通过量, 由图可见, 当天线数 $M = 2$ 时, 采用本文算法可以获得与 Zhang 算法近似的通过量, 而当 $M > 2$ 时, 在本文算法下网络获得的通过量大于 Zhang 算法下的通过量. 同时, 采用 Zhang 算法时网络的通过量随节点天线数的增加而变化较小, 而采用本文算法时网络通过量随天线数的增加而明显增加, 因此两种算法下网络通过量的差异也随天线数的增加而增加. 这是因为 Zhang 算法不允许节点在一个时隙内发送多个数据流, 不能充分利用天线数增加带来的空间自由度, 而本文算法通过在步骤 (17)~(22) 中增加节点发送的数据流数而弥补了这一缺陷, 提高了网络通过量.

接着考察时延性能, 为了简化计算, 在分析时延时假设网络节点的缓冲队列长度 $Q = M$. 图 5 比较了两种算法下网络的平均时延. 这里 Zhang 算法的时延采用了本文中的方法来计算, 因为文献 [3,13] 的方法相当于分析节点缓冲队列长度为 1 的情况, 而本文算法由于允许节点发送多个数据流需要设节点缓冲队列为 Q , 因此如果用文献 [3,13] 的方法来获得 Zhang 算法的时延性能并与本文算法进行比较显然有失公平, 所以我们也采用本文的方法来计算缓冲队列长度为 $Q = M$ 时 Zhang 算法的

图 5 两种调度算法下网络平均时延的比较

Figure 5 Average delay under the two algorithms. (a) Network 1; (b) network 2; (c) network 3

图 6 两种调度算法下网络最大时延的比较

Figure 6 Maximum delay under the two algorithms. (a) Network 1; (b) network 2; (c) network 3

图 7 随机拓扑网络中两种调度算法性能比较

Figure 7 Performance of random topology networks under the two algorithms. (a) Throughput; (b) average delay; (c) maximum delay

时延. 由图 5 可以看出, 当 $M = 2$ 时, 采用两种算法网络平均时延近似. 而当 $M > 2$ 时, 本文算法平均时延优于 Zhang 算法, 并且随天线数的增加, 采用本文算法时网络平均时延的变化量较小, 较 Zhang 算法的优势也越大. 这是由于本文算法在步骤 (2)~(8) 中为节点 $n \in S_1$ 和 n 的一跳邻节点分配时隙时考虑到了 MIMO 的空间复用, 从而减小了帧长, 同时在步骤 (17)~(22) 中允许节点发送多个数据流也减小了某些节点的分组时延.

图 6 中比较了两种算法下网络的最大时延, 可以看到, 在不同天线数下, 采用本文算法时网络的最大时延均优于 Zhang 算法下的最大时延. 因为网络的最大时延由调度帧长及分配到时域和空域资源最少的节点决定的. 本文算法在步骤 (2)~(8) 中充分利用 MIMO 的空间复用来减小调度帧长, 在步骤 (17)~(22) 中增加了节点发送的数据流, 同时本文算法在步骤 (1) 和 (16) 中采用的节点排序方法以及步骤 (6)、(12)、(19) 中采用的空 - 时自由度函数也使得节点分配到的资源相对比较平均, 因此显著减小了网络最大时延.

为了考察本文算法在更一般场景下的性能, 我们随机产生 100 个网络拓扑来测试本文算法和 Zhang 算法. 具体的场景为, 50 个节点随机分布在 $1000\text{ m} \times 1000\text{ m}$ 的范围内, 节点的传输半径为 250 m, 任一节点至少存在一个一跳邻节点. 最终的通过量和时延参数通过对 100 个网络拓扑下的参数进行平均来获得. 图 7 展示了随机拓扑下两种算法的通过量和时延性能. 可以看到, 本文的算法可以获得比 Zhang 算法更大的通过量和更低的时延, 并且两种算法性能的差异随天线数的增加也越来越大. 由此可见, 本文的算法在一般的场景下依然十分有效.

5 结论

本文提出了一种新的支持 MIMO 的 ad hoc 网络传输调度算法. 该算法通过在时隙分配的各阶段充分利用 MIMO 的空间复用能力, 有效地减少了调度帧长, 增加了节点传输的数据流数. 与已有的算法相比, 采用本文算法可以提高网络的通过量, 减小网络的平均时延和最大时延.

参考文献

- 1 Ephremides A, Truong T V. Scheduling broadcasts in multihop radio networks. *IEEE Trans Commun*, 1990, 38: 456–460
- 2 Ramaswami R, Parhi K K. Distributed scheduling of broadcasts in a radio network. In: *Proceedings of IEEE INFOCOM '89*. Ottawa: IEEE, 1989. 497–504
- 3 Hung K W, Yum T S. Fair and efficient transmission scheduling in multihop packet radio networks. In: *Proceedings of IEEE GLOBECOM '92*. Orlando: IEEE, 1992. 6–10
- 4 Madhavi S. A new adaptive transmission scheduling in packet radio networks. In: *Proceedings of 4th International Conference on Networking and Services*. Gosier: IEEE, 2008. 7–12
- 5 Kirt T. Fast and adaptive scheduling in ad hoc networks. In: *Proceedings of 11th International Biennial Baltic Electronics Conference*. Tallinn: IEEE, 2008. 251–254
- 6 Rhee I, Warrier A, Min J, et al. DRAND: distributed randomized TDMA scheduling for wireless ad-hoc networks. *IEEE Trans Mobile Comput*, 2009, 8: 1384–1396
- 7 Demirkol M F, Ingram M A. Power-controlled capacity for interfering MIMO links. In: *Proceedings of IEEE VTC'01 Fall*. Atlantic City: IEEE, 2001. 187–191
- 8 Demirkol M F, Ingram M A. Stream control in networks with interfering MIMO links. In: *Proceedings of IEEE WCNC'03*. New Orleans: IEEE, 2003. 343–348
- 9 Sundaresan K, Sivakumar R, Ingram M A, et al. Medium access control in ad hoc networks with MIMO links: optimization considerations and algorithms. *IEEE Trans Mobile Comput*, 2004, 3: 350–365
- 10 Zhang G H, Li J D, Sheng M, et al. Topology-transparent reservation time division multiple access protocol with MIMO links in multihop ad hoc networks. *IEEE Commun Lett*, 2006, 10: 411–413
- 11 Chu S, Wang X. Opportunistic and cooperative spatial multiplexing in MIMO ad hoc networks. In: *Proceedings of ACM Mobihoc'08*. Hong Kong: ACM, 2008. 63–72
- 12 Chu S, Wang X. Adaptive and distributed scheduling in heterogeneous MIMO-based ad hoc networks. In: *Proceedings of IEEE MASS'09*. Macau: IEEE, 2009. 217–226
- 13 Zhang G H, Li J D, Zhao M, et al. Broadcast scheduling with MIMO links in multi-hop ad hoc networks. *J Electron (China)*, 2007, 24: 477–483

Transmission scheduling algorithm for MIMO link ad hoc networks

CHEN Dan*, LI JianDong & LI ChangLe

State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China

*E-mail: dan_chen@tom.com

Abstract Most reported works on transmission scheduling algorithm for multiple input multiple output (MIMO) link ad hoc networks do not fully exploit the spatial multiplexing potential inherent to multiple antennas system. In this paper, a novel transmission scheduling algorithm is proposed to further enhance the performance of MIMO link ad hoc networks. By utilizing the MIMO multi-stream reception capabilities and ordering the nodes' allocation sequence, the proposed algorithm can dramatically shorten the scheduling frame length and more evenly distribute the data streams allocated to the network nodes. Meanwhile, allowing the nodes to transmit multiple streams in a slot, the algorithm can fully exploit the MIMO spatial multiplexing potential. Additionally, we give the theoretical analysis to quantify the throughput and delay performance with respect to this scheme. Numerical results demonstrate that the proposed algorithm can efficiently enhance the network throughput and reduce the average and maximum delay.

Keywords multiple input multiple output(MIMO), spatial multiplexing, multi-stream reception, ad hoc networks, transmission scheduling

CHEN Dan was born in 1984. He received the B.E. and M.S. degrees from Xidian University, Xi'an in 2006 and 2009 respectively, all in electrical engineering. Currently he is a Ph.D. student at State Key Laboratory of Integrated Service Networks in Xidian University. His current research interests include communication protocols and transmission scheduling for ad hoc networks.

LI JianDong was born in 1962. He received the B.E., M.S. and Ph.D. degrees from Xidian University, Xi'an in 1982, 1985 and 1991 respectively, all in electrical engineering. He has been a faculty member of Telecommunications Engineering at Xidian University since 1985, where he is currently a Professor and Director of State Key Laboratory of Integrated Service Networks. Professor Li is a senior member of IEEE and fellow of CIC. His major research

interests are wireless communication theory, cognitive radio and signal processing.

LI ChangLe was born in 1976. He received the B.E. degree in electromagnetic and microwave technology in 1998, M.E. and Ph.D. degrees in communication and information system in 2001 and 2005, respectively, all from Xidian University, Xi'an. Since April 2005, he has been with State Key Laboratory of Integrated Service Networks at Xidian University, where he is currently an Associate Professor. His research

interests also include cognitive radio, cross-layer design, smart antenna technology and communication protocols for cellular, mobile ad hoc and ultra-wideband networks.