对 IEEE 802.11 协议多跳传输时的时延性能改进*

熊珺洁,屈玉贵,赵保华,刘桂英
(中国科学技术大学 电子工程与信息科学系,安徽 合肥 230027)

摘　要: 提出一种新方法来降低 IEEE 802.11 协议中多跳传输的时延,并利用 NS2 仿真器分析比较了原来的 IEEE802.11 协议和改进后的协议性能, 证明了后者在任何链路负荷的情况下都更加适合多跳传输。

关键词: 802.11 协议　多跳传输时延　无线网络

1　现有的缩短时延的方法

时延是衡量网络性能的一大标准。为缩短时延,国内外提出了许多方法[1]。在无线传感网络中提出了消息传递机制来缩短时延。该机制一次将大量数据完整地从一个节点传递到下一个节点,会导致这 2 个节点间的信道长期不能被其他节点使用,造成其他需要使用这段信道的节点的长时间等待,并不能真正起到整体缩短时延的效果[2]。因此提出了自适应方法来缩短树型应用中多跳传输时延, 但该方法的适用范围很有限。基于非同步的用于 Ad Hoc 网络中实时传输的机制 MACA/PR 和 RTMAC 来缩短时延用到了时隙分配(类似 TDMA),这又会带来其他开销[3][4]。

本文利用无线网络的一个独特之处: 一个节点发送的数据只会传给它的传输范围内的所有节点,它的数据活动范围是一个空间的球体,而有线网络的则是一条线。利用这个由物理层特性决定的特点可节省部分控制帧 ACK 的发送(多跳传输时),缩短传输时延,使目的节点更早收到分组。但是发送节点等待确认帧 ACK 的时间变长了。当然,如果节省下来的 ACK 帧的发送能量能够抵消掉接收所需的额外能量,在以节能为目的的无线传感网络中也可以这样做。

2　对 IEEE 802.11 的 MAC 层协议的改进

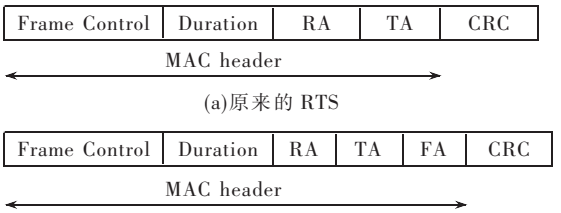
2.1 原 IEEE 802.11 的思想

IEEE 802.11 协议(以下都简称 802.11)利用 CSMA/CA 技术[5]避免冲突。工作过程:站 A 向站 B 发送数据前,先向 B 发送一个请求发送帧 RTS,RTS 中含有整个通信过程需要持续的时间(duration)。立即发送站地址、立即接收站地址、非立即接收站的节点收到该帧就依据该帧中 duration 域的值设置自己的 NAV(网络分配向量,用于判断信道是否被其他节点占用), 等待该通信过程结束后再去竞争信道。B 收到 RTS 后就立即给 A 发送一个允许发送帧 CTS。CTS 中含有剩下通信过程所需时间及 A 站地址。其他节点收到该 CTS 帧后同样也要设置自己的 NAV。这样的 2 次握手可以很大程度上保证整个通信过程不受其他节点干扰。A 收到 CTS 后就发送数据,B 收到数据后就发送 ACK,于是一次通信过程结束。

2.2 改进思想

多跳情况下,如果数据从 A 发送给 C,中间经过 B 来转发, 当 B 收到 A 的数据后, 除了要立刻回送给 A 一个 ACK, 还要竞争信道,给 C 发送 RTS。考虑到 ACK 和 RTS 一个在前,一个在后,时间上连续,因此希望将 2 个帧合为 1 个帧,让该帧既能起到 ACK 的确认作用,又能发挥 RTS 的请求发送功能。这样就可以克服 802.11 浪费控制帧的缺点,节约 1 个确认帧 ACK。

为实现新的功能,需构造 1 个新的控制帧(一种新的 RTS)。该控制帧就是通过在 RTS 中加 1 个地址(FA):转发节点的上一站的地址。原来的 RTS 与修改后的 RTS 的结构如图 1 所示。



(a)原来的 RTS

(b)修改后的 RTS

图 1　RTS 结构图

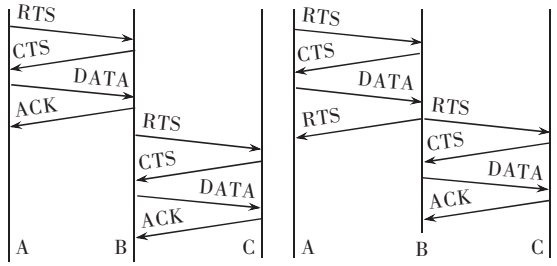
B 不发送 ACK,而发送出这样一个修改后的 RTS。当 A 收到后(C 也会收到这个 RTS,不过对于 C 而言,这个就是请求通信的 RTS),发现该帧不是发给自己的,但是该帧的前一站的地址是自己的地址,即可判断该帧是发送给自

* 基金项目:国家"973"项目(No.HH0616000001);国家自然科学基金项目(No.6024004)

己的ACK,于是结束自己和B之间的通信。通常RTS的竞争时间比较长,所以有时候A需要等待很长时间才能收到B发送的RTS,因此选择A等待RTS形式的ACK的time-out时间很关键。此外各个帧的duration域的值也变了,必须重新计算。NAV函数(虚拟载波检测函数)也需在原来代码基础上修改。

当不再需要转发的时候,也就不需要再发送RTS。所以没有必要再发送RTS与ACK的合成帧,只需要发送ACK。此时需要一个机制来区别对待,即通过上层(路由层)来做特殊处理,因为MAC层无法判断出最终的目的地址是不是本节点(IP地址对MAC层是透明的)。为了不破坏分层的思想,达到真正的模拟效果,需在路由层和链路层增加相应的判断和处理代码。

其实,该方法只不过是一种捎带思想,在发送RTS时捎带了ACK,不发送RTS时就不捎带。改进前后802.11的数据转发过程如图2所示。



(a)原来　　(b)改进后

图2　改进前后802.11的数据转发过程

2.3 NS2下802.11源代码及修改

主要修改之处是在程序Mac-802_11.h中为RTS的帧增加一个前一站的地址。

```
struct rts_frame {
    struct frame_control rf_fc;                    //控制域
    u_int16_t       rf_duration;          //RTS的持续时间
    u_char          rf_ra[ETHER_ADDR_LEN];
                                          //立即接收站的地址
    u_char          rf_ta[ETHER_ADDR_LEN];
                                          //立即发送站的地址
    u_char          rf_fa[ETHER_ADDR_LEN];
                                          //增加的,上一站的地址
    u_char          rf_fcs[ETHER_FCS_LEN];
                                          //帧检验序列
    };
```

为了能让转发节点将上一站的地址提取出来并保存到立刻要发送的RTS帧的FA域中,在类Mac802_11中增加了一个中间变量former用于事先存储该地址。

接收站收到DATA后,会判断该分组是否是重复报文,如果是则要做处理。原来的802.11直接丢弃该报文,没有给发送站发送ACK,结果发送站就不断地重发数据,接收站不断发现是重复报文,就不断丢弃,直到发送站的重传次数达到限度,放弃重传为止。这样做极为浪费资源,所以本文的方法是立刻发送ACK。此外,发送站可能会收到迟到的ACK,此时该站正在重发数据,可能处于任何一种状态(发送RTS、等待CTS、发送DATA、等待ACK或IDLE)。但是不管它处在何种状态,它都应该立刻撤销重传,回复到初始状态。因此本文在类Mac802_11中增设了一个标志变量flag,将它用于重传机制中。

原来的NAV函数的功能为:收到一个不属于自己的分组时,依据分组的duration域设置虚拟探测仪的值,以后再收到不属于自己的分组时就要比较,看是否新分组的duration值会使虚拟探测仪的值更大。若是,则用新分组的duration值来重设虚拟探测仪的值;否则仍用原来的,不做改变。这样的功能不好,因为情况可能变了,不需要再等那么长的时间。例如新的协议中源节点可能很早就收到了ACK,其他节点就没必要还等那么长的时间,这样就不能充分利用空闲出来的信道,并且还会使收到RTS的节点依据虚拟探测仪就误以为信道忙,而不敢发送CTS,从而导致RTS的重传。所以新的NAV函数的功能应该是:只要收到不属于自己的分组,就要依据该分组的duration域的值来更新虚拟探测仪的值。

当多跳转发到终点后无需再发送RTS时,上层(路由层)应该通过发送一个新类型的空分组通知MAC层发送ACK。但是NS仿真中任何时候链路层都不会将MAC层收到的分组首先交给路由层,而是先将分组交给哈希地址分类器。地址分类器发现该分组的最终目的不是自己,才将它转交给路由层,若是自己的就不交给路由层,而交给端口分类器。端口分类器再将该分组交给相应的进程。但是由于新协议要求路由层能发现最终目的站是自己,然后通知下层发送ACK。如果在这里分组就被地址分类器给过滤掉了,路由层就永远无法通知下层发送ACK了,所以在哈希分类器的源代码中要加入处理代码,要求无论如何,即使分组已经到了目的站,此时不需要寻找路由了,还是要交给路由层处理。

本文编写了一个具有基本接收、发送和转发功能的静态路由协议SimRoute。用该路由协议,而不用NS中编好的路由协议模块AODV、DSR、DSDV等,是因为用它容易看出新的802.11协议的效果,排除由于路由寻找而带来的干扰,并且修改上层也比较容易。

3　模拟仿真

3.1 延时理论计算

在NS的系统文件ns-default.tcl中设定了802.11的许多参数的默认值,如:

SIFS　10μs(短帧间间隔)

PreambleLength　144b(物理层的前同步码长度)

PLCPHeaderLength　48b(物理层头部长度)

PLCPDataRate　1Mbps(物理层的数据传输速率)

另外根据文件 Mac802_11.h 中的各个控制帧的结构可以算得各个帧的长度：

RTSLength=44B(原来的 802.11 中)

RTSLength=50B(新的 802.11 中)

CTSLength=38B

ACKLength=38B

因默认数据传输速率是 1Mbps，故各帧所需传输时间是：

tRTS=44 * 8/1 μs=352μs(原来的 802.11 中)

tRTS=50 * 8/1 μs=400μs(新的 802.11 中)

tCTS=38 * 8/1 μs=304μs

tACK=38 * 8/1 μs=304μs

MAC 层的最大处理时延 DSSS_MaxPropagationDelay 为 2μs；LL 层处理从 MAC 层或路由层来的分组的默认时延是 25μs；路由层是静态路由，默认处理时延为 0。

新的 RTS 帧比原来的 RTS 长 6 个字节，故多需要6×8b÷1Mbps=48μs 的传输时间。先考虑只转发一次所节约的时延。由图 2 知，2 个协议都发送了 2 次 RTS，但新协议少发送了一个 ACK，少需时间 10μs+304μs。新协议中要求即使最后分组已经到了目的站，还是要将分组交给路由层让它通知下层发送 ACK，故多出 LL 层的处理时延 50μs(路由层将分组交给 LL 层，LL 层处理用了 25μs，LL 层再用 25μs 将分组交给 MAC 层)。但综合起来数据仍然提前到达目的节点的应用层，所耗时间可减少：-48-48+10+304-50=168μs。此数据与链路负荷轻时的仿真结果一致。

若需转发 N 次，数据传输速率为 VMbps，ACK 帧的长度为 A 字节，新的 RTS 增加的地址长度为 L 字节，发送 ACK 之前等待的时间为 $SIFS$μs，链路层处理一次分组所需时间为 Tμs，则一共发送了(N+1)个 RTS 帧，那么所节省的时间为：

$$t=[-8*L/V+(-8*L/V+SIFS+8*A/V-2*T)*N]\mu s$$

当 V=1，L=6，$SIFS$=10，T=25 时，t=(216N-48)μs，故 N 越大，即跳数越多，就越省时。

当 N=5 时，t=1 032μs，与后面链路负荷轻时的仿真结果一致。

3.2 TCL 仿真代码编写

在 TCL 仿真代码中使用新编写的静态路由协议 SimRoute。利用该路由协议的功能实现程序 simroute.cc 中的接口函数 command( )的功能，依据所要仿真的拓扑结构用 TCL 代码填写出相应的静态路由表。在流量安排的问题上，为简单起见，先不绑定应用层，而使用 UDP 代理，直接在 udp 层发送数据。这样就可以自如地控制流量，而不受上层应用的影响，将问题集中在改进前后的 802.11 协议的性能比较上。

3.3 NS 下的仿真数据

3.3.1 拓扑配置与流量安排

为测试不同情况下的协议效果，需要不同的拓扑配置和流量安排。

情况 1：为 3 个节点的拓扑结构，如图 3 所示。3 个节点排成一条线，间距 250m。这样选择是考虑到无线网络的传输范围是 250m，干扰范围是 550m。

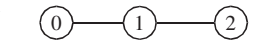


图 3　3 个节点的拓扑结构

节点 0 只发送一个数据给节点 2。节点 0 每隔 0.1s、0.01s、0.005s、0.003s、0.002s 给节点 2 发送数据。

没有必要考虑节点 0 和 2 同时给节点 1 发送数据的情况，因此时没有多跳，新协议显然没有任何优越性，反而增加了开销。

情况 2：是 7 个节点的拓扑结构，如图 4 所示。7 个节点，排成一条线，间距 250m。

图 4　7 个节点的拓扑结构

节点 0 给节点 6 发送一个数据。节点 0 每隔 0.1s、0.015s、0.012s、0.01s、0.001s 给节点 6 发送一个数据。节点 0 和 6 同时给节点 3 发送一个数据。节点 0 和 6 每隔 0.1s、0.015s、0.01s、0.008s 给节点 3 发送数据。

3.3.2 数据处理结果的表格统计

以上二种情况的数据统计分别如表 1 和表 2 所示。

表 1　情况 1 的数据统计

情况 1：无冲突	原来时延/μs	改进后时延/μs	(原来-改进)时延差/μs	丢包率/%	
				原来	改进
0 向 2 发送一 200 字节的包	6353.0	6185	168	0	0
0 每隔 0.1s，共 300 包	6390.0	6222.0	168	0	0
0 向 2 每隔 0.01s，300	6466.93	6298.93	168	0	0
0 向 2 每隔 0.005s，100	94544.5	98636.9	-4092.4	0	0
0 向 2 每隔 0.005s，300	224388	231949	-7561	9.33	11.33
0 向 2 每隔 0.003s，100	186170	183439	2668	5	7
0 向 2 每隔 0.003s，300	261499	263500	-2001	39.33	40.33
0 向 2 每隔 0.002s，100	195519	193165	2354	20	21
0 向 2 每隔 0.002s，300	258566	259407	-841	53.67	54.33

表 2　情况 2 的数据统计

情况 2：有冲突	原来时延/μs	改进时延/μs	时延差/μs	丢包率/%		冲突次数	
				原来	改进	原来	改进
0 向 6 发送一个包	19447	19887.8	1032	0	0	0	0
0 向 6 每隔 0.1s，共 300	19887.8	18855.8	1032	0	0	0	0
0 向 6 每隔 0.015s，300	19844	18740.3	1103.7	0	0	0	0
0 向 6 每隔 0.012s，300	668915	435551	233364	7	4.33	537	383
0 向 6 每隔 0.01s，300	959927	518223	441704	17.33	16.33	435	332
0 向 6 每隔 0.001s，300	535267	449933	85334	77	76	129	91
0，6 向 3 发送一个	12926.17	12451.83	474.33	0	0	1	1
0，6 向 3 每隔 0.1s，300	13269.7	12778.2	491.5	0	0	204	250
0，6 向 3 每隔 0.015s，300	144289	319536	-175247	0.67	0	315	329
0，6 向 3 每隔 0.01s，300	625129	662181	-37052	27	15	203	329
0，6 向 3 每隔 0.008s，300	673942	640000	33942	32	20.33	182	329

3.3.3 绘图及分析

图 5 为情况 1 下 2 个协议多跳传输的时延差。当每个时间间隔发送分组个数为 100 时，可以看到负荷重时改进的 802.11 延时性能不如原来的 802.11，且随着负荷加得更重，2 个协议延时性能的差距缩小。但是只要负荷不重，改后的 802.11 的延时性能要好一些。其原因是此时数据只转发了一次，跳数太少。同样，当每个时间间隔发送 300 个分组时，结果类似，只不过此时链路负荷更重，因此时延性能略差于只发送 100 个分组的情况。由表 1 数据知跳数不太多时，修改后的 802.11 的丢包率要高一点，但是随着负荷加重，丢包率的差距也减小。

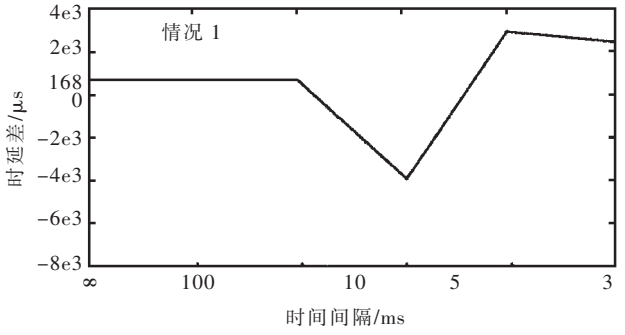


图 5 情况 1 的时延差(发送 100 个分组)

图 6 和图 7 分别为情况 2(即一点对一点)的时延差和丢包率。由图 6可知不论链路负荷有多重，修改后的 802.11 的时延始终比原来的 802.11 协议小。可见，跟原来的协议比，新协议更加适合多跳传输，跳数越多，它的优越性就越显著，且由图 7 可知新协议的丢包率也低一些。由表 2 可知，虽然新协议的冲突次数要少一些，但是 RTS 冲突导致 RTS 丢弃，即产生新协议的确认帧丢失，所以会出现重复分组。但是总的说来多跳传输性能还是比原来的 802.11 协议性能好得多。

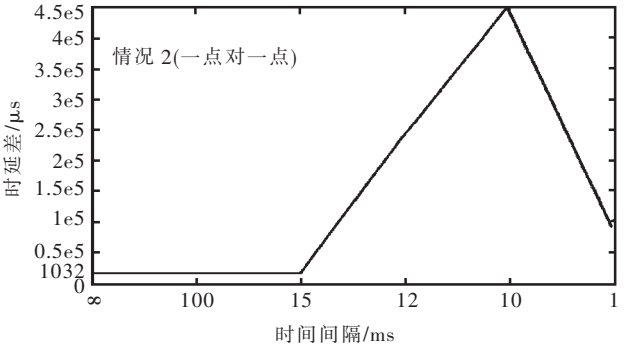


图 6 情况 2(一点对一点)的时延差

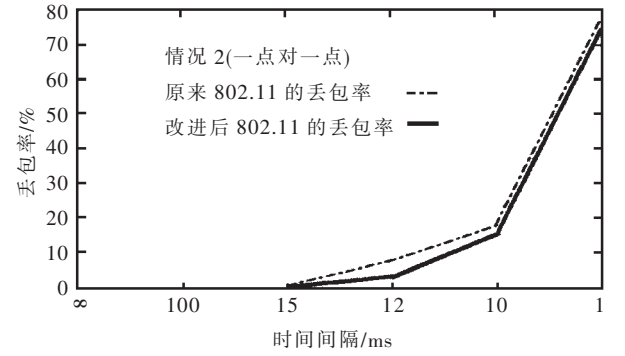


图 7 情况 2(一点对一点)的丢包率

情况 2 下多点对一点时的时延差和丢包率分别如图 8 和图 9 所示。由图 8 可知，负荷轻时改进的 802.11 始终比原来的 802.11 时延小。负荷比较重时则改进的协议时延要长得多，但是负荷更重的时候时延就短很多。原因是修改后的 802.11 转发时不发送 ACK，而是发送 RTS。这样除非 RTS 冲突，否则发送数据的节点就可以将信道的控制权成功地交给下一跳的节点，于是可以连续完成一次多跳传输；而原来的 802.11 转发时发送 ACK，这就导致信道的公平竞争，不能保证下一跳节点获得信道的控制权，也就不能保证连续完成一次多跳传输(尤其在负荷很重的时候，信道的控制权连续交给下一跳的概率将更低)，这就会在多跳且负荷很重时引入多余的时延。由图 9 知新协议的丢包率始终比原来的低，且链路负荷越重它们丢包率的差距就越大。

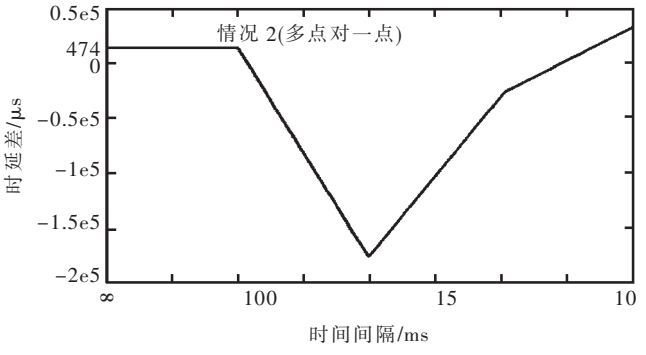


图 8 情况 2 下多点对一点时的时延差

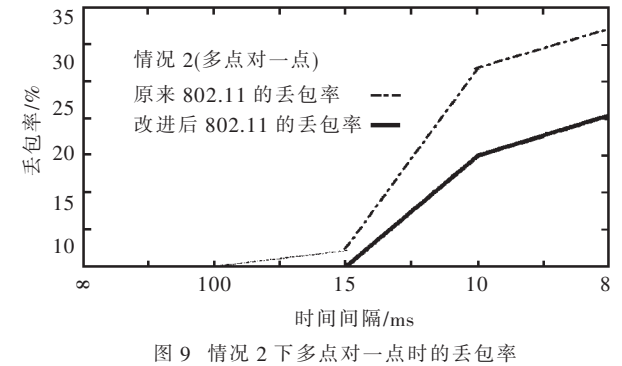


图 9 情况 2 下多点对一点时的丢包率

4 结 论

综上所述可知：没有太多冲突的时候，即使负荷很重，在多跳的情况下改进的 802.11 时延也小很多。但是如果跳数太少，如只转发一次，则改进的 802.11 协议的优越性就不明显；若负荷很重则还比原来的协议时延长不少，并且丢包率也较高。在多个节点同时给同一个节点发送数据时，2 个协议都有冲突，结果大多是丢弃 RTS。在改进的 802.11 协议中，此时由于 RTS 冲突被丢失而导致不必要的

数据分组重传。但此时原来的802.11协议的丢包率更高，成功传输的分组数目比改进的802.11协议少很多。在负荷很重时原来的802.11协议的时延反而更长。可见改进的802.11适用于多跳且冲突不多的情况。在一点对一点的情况下，无论链路负荷如何，改进的协议更适合多跳传输；在多点对一点的情况下，只要链路负荷不重，冲突不多，则修改后的协议在多跳传输方面更有价值。

参考文献

1 Ye W，Heidemann J，Estrin D.An Energy-efficient MAC Protocol for Wireless Sensor Networks.In：IEEE INFOCOM，2002

2 Lu G，Krishnamachari B，Raghavendra C S.An Adaptive Energy-efficient and Low-latency MAC for Data Gathering in Wireless Sensor Networks.In：IEEE Proceedings of the 18th International Parallel and Distributed Processing Symposium，2004

3 Lin C R，Gerla M.MACA/PR：Asynchronous Multimedia Multihop Wireless Network.In：Proc of IEEE INFOCOM，1997

4 Manoj B S，Siva R M.Real-time Traffic Support for Ad Hoc Wireless Networks.In：10th IEEE International Conference，2002

5 ANSI/IEEE Std 802.11.1999

（收稿日期：2005-06-14）