无线传感器网络
——MAC协议

重庆邮电大学

主要内容

- MAC协议需求与分类
- 竞争型MAC协议
- 分配型MAC协议
- 混合型MAC协议与跨层设计
- 总结

MAC协议需求与分类

MAC协议需求

- 在设计无线传感器网络**MAC**层协议时，需重点考虑如下问题：
  - 节省能量，即追求能量效率（**Energy efficiency**）
  - 可扩展性，适应节点数目、拓扑动态变化的网络
  - 网络效率，包括网络公平性、实时性、吞吐量以及带宽利用率等

MAC协议分类

- 从不同的角度入手，对**MAC**协议进行分类的方法有多种，可以根据**MAC**协议使用的信道数目分为基于单信道、基于双信道和基于多信道三类;可以根据**MAC**协议分配信道的方式分为竞争型、分配型以及混合型;可以根据网络类型是同步网络还是异步网络，将**MAC**协议分为同步、异步两类。
- 本书中采用根据**MAC**协议分配信道的方式来进行分类，从竞争型、分配型以及混合型三种类型入手，介绍目前比较有代表性的**MAC**协议。

经典的MAC协议列表

协议方案	出现时间	类型	是否需要精确同步	基本机制
SMAC	2002	竞争型	否	CSMA
TMAC	2003	竞争型	否	CSMA
PMAC	2005	竞争型	否	CSMA
WiseMAC	2004	竞争型	否	CSMA
Sift	2003	竞争型	否	CSMA
SMACS	2000	分配型	是	TDMA/FDMA
TRAMA	2003	分配型	是	TDMA/CSMA
DMAC	2004	分配型	是	TDMA/Sloted ALOHA
ZMAC	2005	混合型	是	TDMA/CSMA

竞争型MAC协议
SMAC，TMAC，BMAC

SMAC协议概述

- **SMAC (Sensor MAC)**协议是较早提出的一种基于竞争的无线传感器网络**MAC**协议，由**USC/ISI**的**Wei Ye**等人提出，并在**NS2**、 **TinyOS**等平台上进行了仿真和实现。该协议继承了**802. 11 MAC**协议和**PAMAS**协议的基本思想，在此基础上加以改进，以**WSN**的能量效率为主要设计目标，较好地解决了能量问题，同时兼顾了网络的可扩展性，为广大研究人员参考和比较。

MAC协议导致能量浪费的因素

- 冲突
  - 冲突后重传需要消耗能量
- 串扰（overhearing）（或串听）
  - 收到了发给别人的数据包，需丢弃
- 控制开销
  - 由于传输帧头等非实际负载所带来的能量消耗
- 空闲监听
  - 即便不接收数据，监听信道会消耗相当于接收的50%-100%的能量

S-MAC如何解决上述问题？

- 冲突
  - 解决方法：带**NAV**的**RTS/CTS**机制。
- 串扰
  - 解决方法：当邻居节点收到**RTS**或**CTS**帧后，立即休眠**NAV**时间才再次探测信道。
- 控制开销
  - 解决方法：消息传递机制。
- 空闲监听
  - 解决方法：周期性的监听和睡眠。

单选题 1分

设置

以太网采用的MAC接入方式是

A CSMA/CD

B CSMA/CA

C DCF

D PCF

提交

单选题 1分

设置

在以太网MAC中，发送方发完数据包后，是否需要等待接收方反馈的ACK（确认帧）？

A 需要等待ACK

B 不需要等待ACK

C 可以设置标志位来向接收方指示是否需要ACK

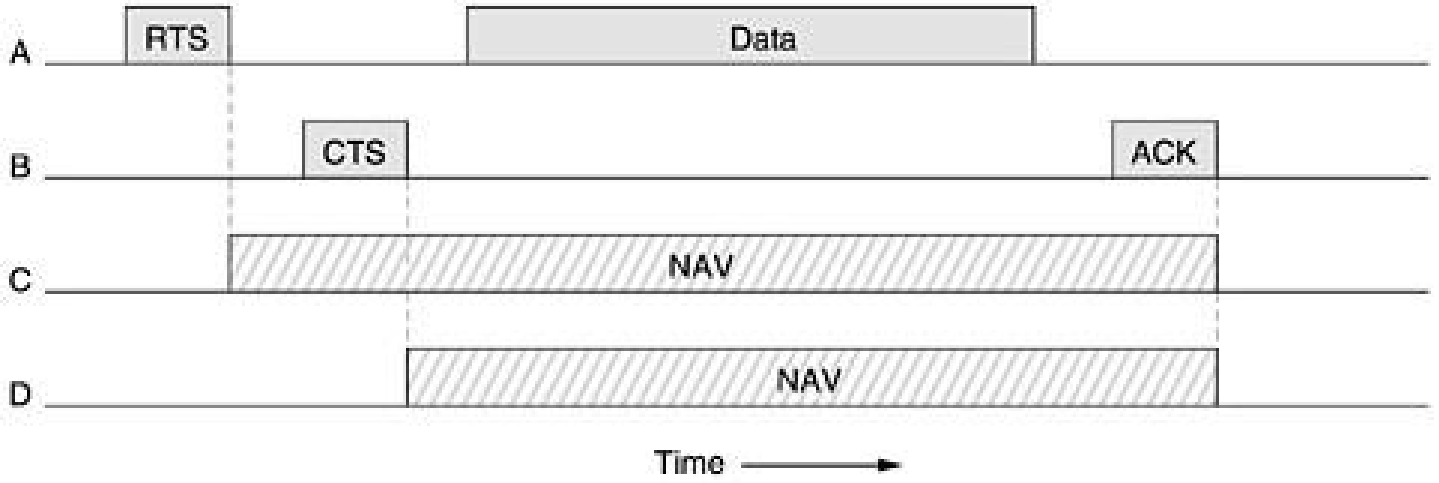
D 根据接收方意愿来反馈ACK

提交

RTS/CTS机制

- RTS：Request to Send 发送方在发送前先发送一个请求帧。
- CTS：Clear to Send 接收方收到请求帧后若同意接收，则回复此帧。
- NAV：Network Allocation Vector 网络分配向量 当某个节点监听到其他人之间的RTS或CTS帧时，当时不再进行信道接入，而是延迟一段时间再活动，这段时间称为NAV。
- **虚拟信道监听机制**：节点虽未在信道中检测到被占用，但收到RTS/CTS后虚拟的认为信道要被占用，不再尝试进行信道接入。需要等待的NAV时间携带在数据包的duration域中。

RTS/CTS实例



- ★ A想向B发数据，C位于A的无线范围内，D在B的无线范围内，不在A的无线范围内。
- ★ D实际上探测不到Data的传输，但因为收到了CTS，就会按照CTS中携带的NAV时间，在该时间段内不再探测信道，从而不会干扰A的发送过程，即虚拟监听信道。

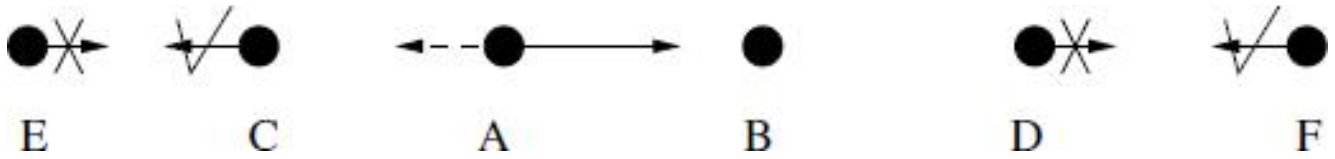
RTS/CTS分析

- 当多个节点都想向同一个节点发送数据时，通过RTS/CTS机制能够解决冲突问题。
- 带NAV的RTS/CTS机制基本能够解决隐蔽站问题。
- RTS/CTS机制来源于IEEE 802.11（WiFi）的MAC机制。

串扰的解决：NAV时去休眠！

- 在IEEE 802.11的RTS/CTS机制中，当节点处于虚拟监听（NAV）阶段时，仍会监听邻居的数据包。
- 在SMAC中，节点收到邻居节点的RTS或CTS后，就立即进入休眠状态，休眠NAV时间后才醒来再竞争信道，从而避免了接收不属于自己的数据包，浪费能量。

串扰示例



- 上图中，假如每个节点只能跟自己的邻居通信，当A发送数据给B时，为了防止冲突，按图中各个节点发送数据的方向，哪些节点应该去休眠？

为什么所有邻居节点都需要去休眠？

- D必须要去休眠，否则其信道会干扰B的接收。
- E和F因为在两跳之外，不会干扰A到B的传输，所以不需要休眠。
- C是否需要休眠？C发送给E的数据并不会干扰A-B之间的数据传输，但是，**C虽然能发送数据给E，但却不能正确接收到E返回的数据，例如CTS、ACK等帧，因为其接收会受到A的干扰！**
- **所以SMAC规定，A和B一跳之内的邻居都应该去睡觉！无论其是想发送还是想接收，统统禁止。**

控制传输开销的两难选择

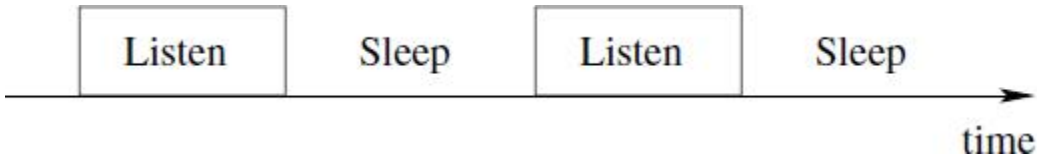
- 当要传输的消息较长时，有两种方法：
  - 一是一次性发送，但如果由于几个比特错误造成重传，则会造成较大的延时和能量损耗。
  - 二是将长包分成若干个小数据包单独传输（分段），则又会由于**RTS/CTS**的使用形成过多的控制开销。

SMAC的解决方案

- SMAC提出了“消息传递”机制。将长的信息包分成若干个DATA，并将它们一次传递，但是只使用一个RTS/CTS控制分组作为交互。节点为整个传输预留信道，当一个分段没有收到ACK响应时，节点便自动将信道预留向后延长一个分段传输时间，并重传该分段，整个传输过程中DATA和ACK都带有通信剩余时间信息（NAV），邻居节点可以根据此时间信息避免串扰。

周期性监听和睡眠

- 很多传感器网络中，节点长期处于监听状态，但并没有什么事件发生，就浪费了很多监听能量。
- SMAC中为了减少监听对能量的浪费，要求节点周期性的交替监听与睡眠，例如在一个周期内如果一半时间监听一半时间睡眠，就可以节省50%的监听能量。
- 节点周期性的醒来，看有没有节点想与它通信，没有的话就再次睡觉，再次醒来，再次睡觉… …



调度表与虚拟簇

- 为了使周期性监听和睡眠能够运转，节点需要知道所有邻居节点的睡眠和醒来的时间（以**调度表**形式存放在自己的内存中），同时也需要让邻居节点知道自己的睡眠和醒来时间，这样它们才能在对方醒来时及时进行通信。
- 最好的方式是某个节点与它周围的邻居节点能够按同一个步调睡觉和醒来，即调度表相同。如果某一片区域节点的调度表一致，则称它们形成了一个**虚拟簇**。

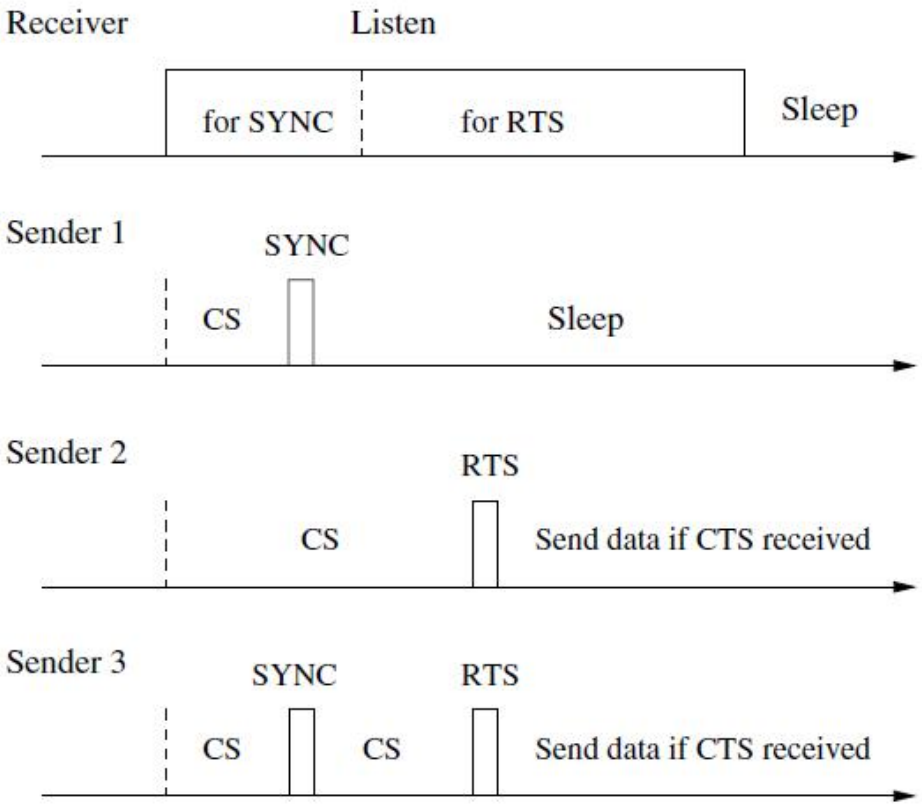
调度信息的传播过程

- 节点首先监听一段时间。如果一直没有收到邻居节点的调度信息，就把自己构造的调度信息向邻居广播（通过SYNC消息广播），成为同步发起者（synchronizer）。
- 如果节点在时间截止前收到了邻居的调度信息，就把自己的调度信息修改成跟邻居的调度信息一致，称为跟随者（follower）。等待一段时间后，再把该调度信息广播出去。
- 如果节点在广播了自己的调度信息后又收到了邻居发来的调度信息，则该节点同时遵循两个调度信息。（即每个调度信息需要醒来的时候都醒来）。

调度消息与数据帧的共存

- 为了使一个节点既能收到调度信息帧（SYNC)，又不影响正常的数据收发，将醒来后的监听周期分为两个阶段：第一阶段用于发送或接收SYNC帧；第二个阶段用于发送或接收RTS帧。

各种不同的共存情况



- 接收节点：分为两部分接收
- 发送节点1：只想发送调度信息
- 发送节点2：只想发送数据（RTS先请求）
- 发送节点3：既想发送调度信息，又想发送数据

SMAC小结

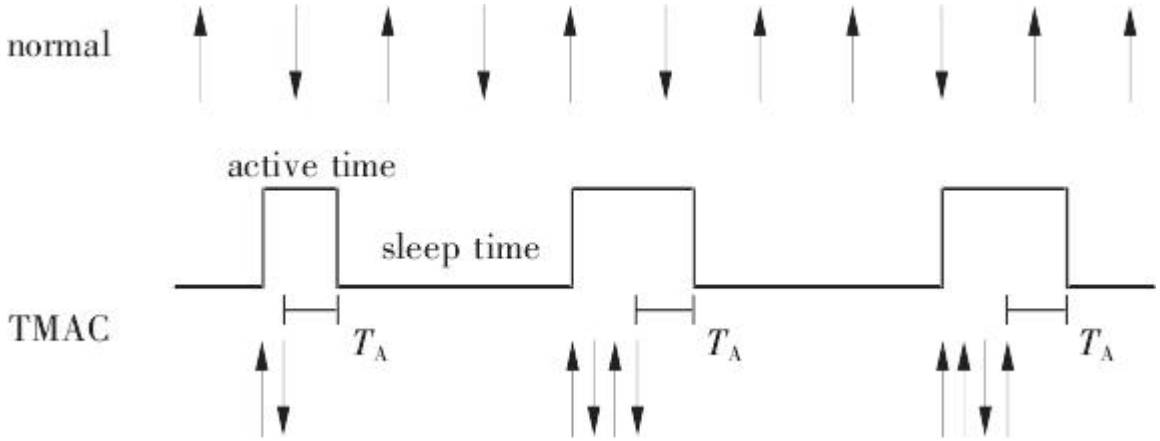
- 通过采用低占空比的周期性监听和睡眠的调度，SMAC协议减少了节点空闲监听的能量损耗;通过采用串扰避免和消息传递机制，SMAC协议减少了串扰和控制数据包带来的能量损耗;为低功耗无线传感器网络应用开发提供了基础。

TMAC协议的提出

- SMAC的不足：
  - 虽然较好的解决了能量消耗问题，但采用固定的调度周期，不能很好的适应网络流量的变化。
  - 例如当节点醒来时无事可做时，仍要等待固定的监听时间才能转入休眠状态，这仍然会浪费一定的不必要的能量。
- TMAC能够动态调整调度周期中活跃时间的长度，从而自适应的调整占空比。

TMAC协议运行示意图

问题：如何才能实现自适应调整活跃时间？



- 可以看出，TMAC能够根据网络流量大小自动调整活跃时间。

自适应调整监听时间

- TMAC协议中，每个节点都周期性地唤醒，进入活跃状态，和邻居进行通信，然后进入睡眠状态，直到下一个周期开始。
- **节点之间使用RTS-CTS-DATA-ACK交互的方法，以确保避免冲突和可靠传输。**
- 在醒来时，如果连续一个时间段T_A内没有任何激活事件发生，则自动结束活跃状态，进入休眠。
- T_A决定了在一个调度周期中进行空闲监听的最短时间。

★ 接下来的问题是：

★ 哪些事件可以作为激活事件？

★ TA应该如何取值才合适？

激活事件

- 在无线信道上收到数据
- 通过RSSI感知存在无线通信
- 通过侦听RTS/CTS分组，确认邻居的数据交换已经结束
- 周期时间定时器溢出

TMAC对RTS/CTS的改进

- 当节点发送RTS帧后，如果没有接收到相应的CTS帧，有以下三种可能:①接收节点处发生碰撞，没能正确接收RTS帧;②接收节点在此之前已经接收到串扰数据;③接收节点处于睡眠状态。如果发送节点在时间T_A之内没有接收到CTS帧，节点会进入睡眠状态。但是如果是前两种可能导致节点没有接收到CTS帧，那么当它进入睡眠时，它的接收节点还处于监听状态，发送节点此时进行睡眠会增加传输延迟。
- **因此在TMAC中，第一次发送RTS未能建立连接后，应再重复发送一次RTS，如仍未收到CTS才转入睡眠。**

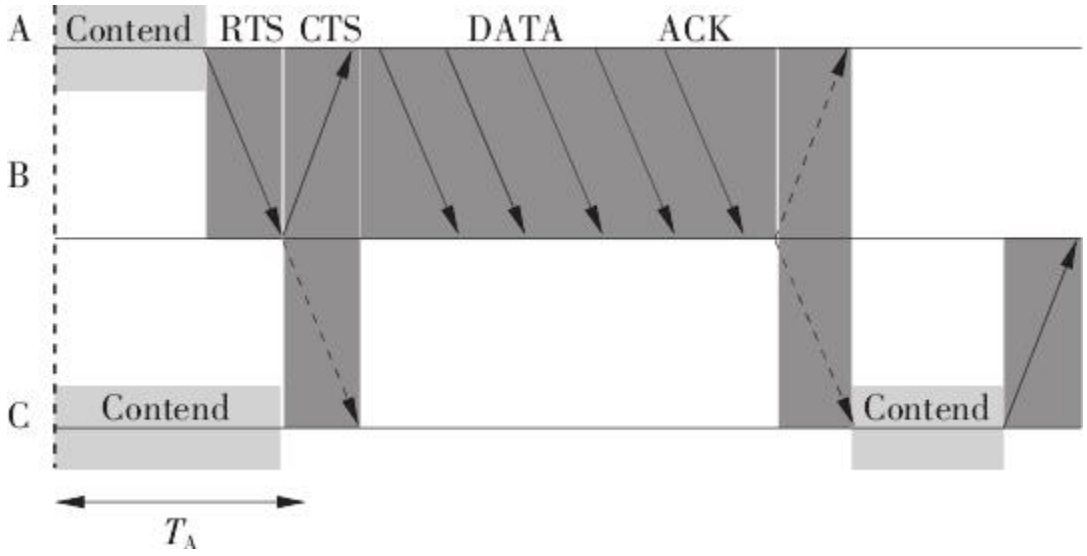
RTS/CTS机制下的T_A取值

- **TMAC**协议中，当邻居节点还处于通信状态时，节点不应该进行睡眠，因为节点可能是接下来信息的接收者。
- 节点发现串扰的**RTS**或**CTS**都能够触发一个新的监听间隔T_A。
- 为了确保节点能够发现邻居的串扰，T_A的取值必须保证当节点能够发现串扰的**CTS**，所以**TMAC**协议规定T_A的取值范围如下：

$$T_A > C+R+T$$

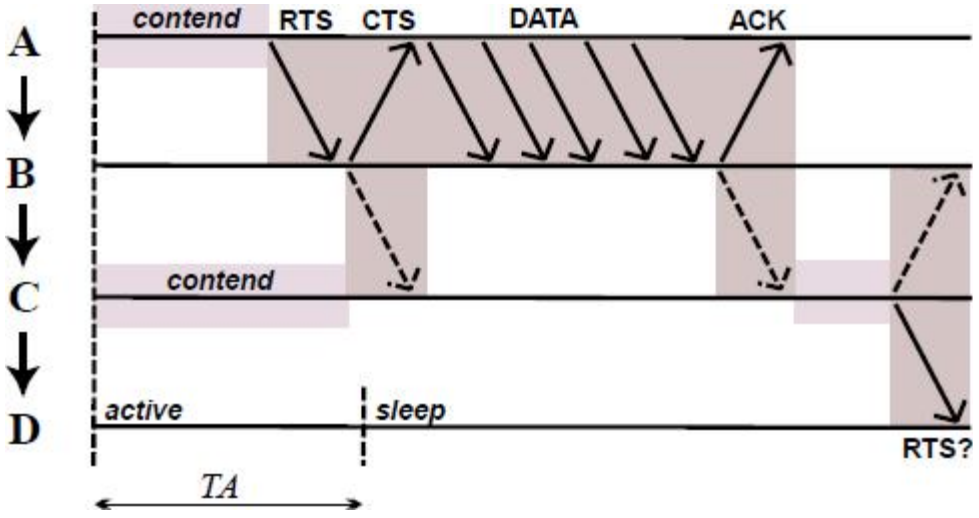
C为竞争信道的时间，**R**为发送**RTS**需要的时间，**T**为**RTS**发送结束到开始发送**CTS**的时间。

RTS/CTS交换示意图



早睡问题的产生

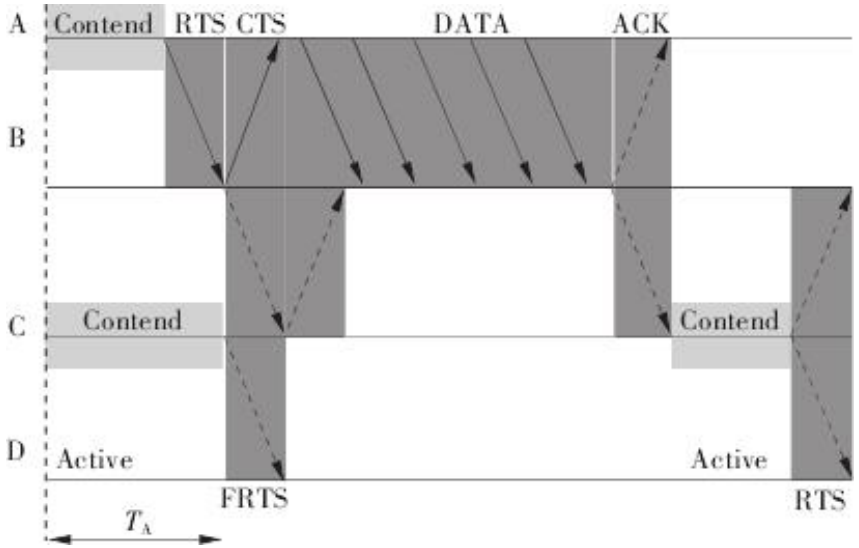
- 在采用周期性调度的MAC协议中，如果一个节点在邻居准备向其发送数据时进入了睡眠状态，这种现象称为早睡。



C想跟D通信，但C监听到了A和B之间的CTS，会继续监听等待，但D没能监听到RTS/CTS，所以超过T_A时间后就休眠了。等A-B传输结束后C占有信道想发数据给D时，D已经早睡了。

早睡问题的解决方法（1）

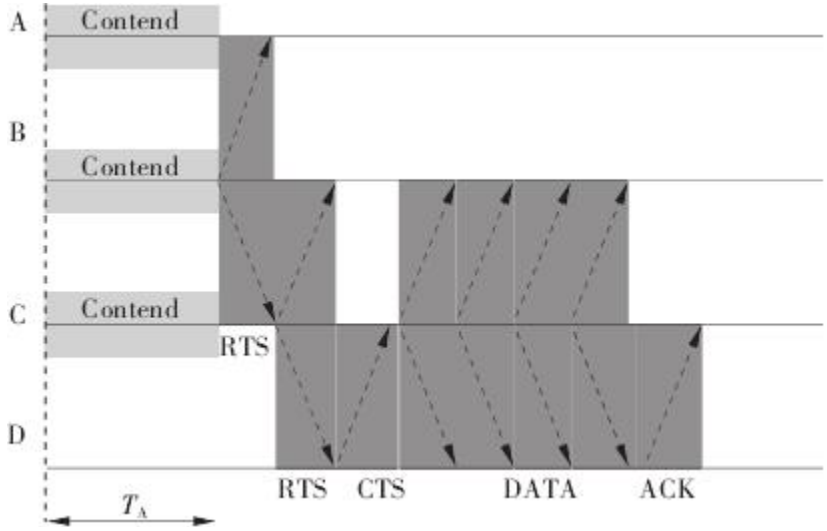
- **Future request-to-send**
- 第一种方法是未来请求发送。当节点C收到B发给A的CTS后，立即向D发送一个FRTS帧。FRTS帧包含节点D接收数据前需要等待的时间长度，D在此时间内须保持监听状态。



早睡问题的解决方法（2）

Full-buffer priority

- 第二种方法是满缓冲区优先。当节点的缓冲区接近占满时，对接收到的RTS帧不回复CTS，而是立即向缓冲区中数据包的目的节点发送RTS，以建立数据传输。B向C发送RTS，C因缓冲区快占满不发送CTS,而是发送RTS给D。这个方法的优点是减少了早睡问题发生的可能性。

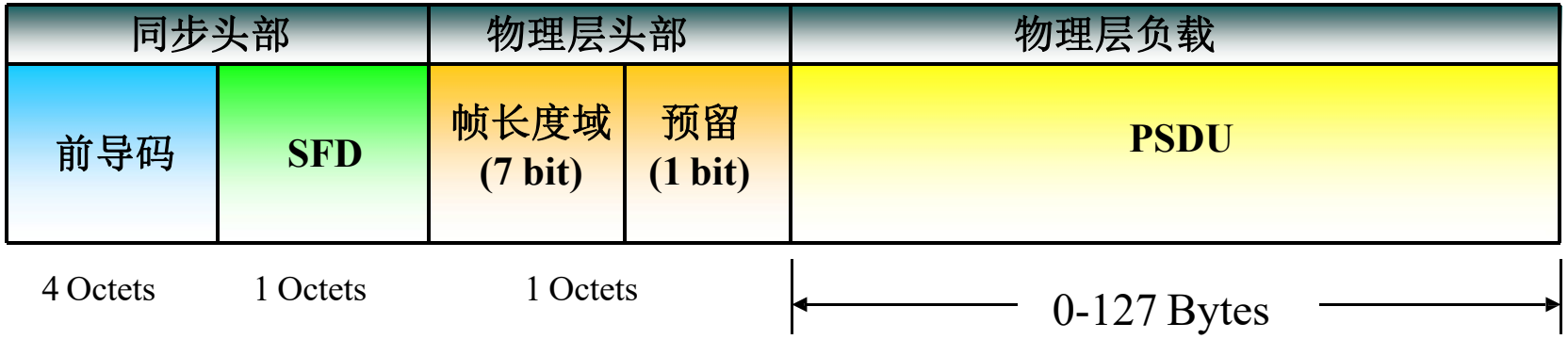


BMAC概述

- 在SMAC中，为了实现节点间同步的监听和睡眠，需要构造虚拟簇。而虚拟簇的建立和维护需要大量的控制开销，从而导致能量消耗。
- BMAC针对这个问题，设计了一种基于异步周期性监听/睡眠调度机制的MAC协议。
- BMAC协议让各个节点独立决定自己的睡眠和调度时间，不需要在所有节点间建立和保持同步，而是只在发送者有数据发送时直接和接收者建立同步。

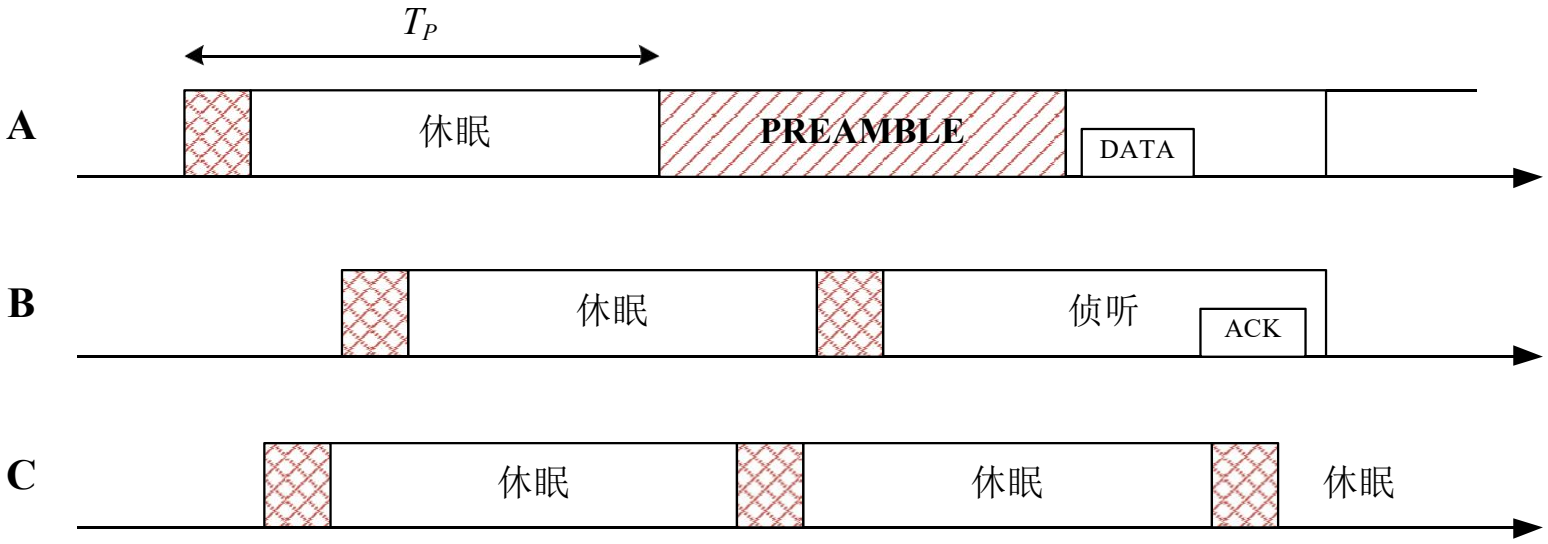
前导采样

- BMAC采用前导采样来实现发送/接收节点之间的同步，也称为低功耗侦听（LPL，Low Power Listening）。
- **前导码知识补充：**在无线通信中，发送方在传输数据前，一般会发送一些比特流，用于完成收发双方的同步（步调一致），即前导码。
- 以IEEE 802.15.4物理层帧为例：



低功耗侦听

- 当发送节点需要发送数据时，首先在信道上传输一个扩展前导码用于唤醒接收者。
- 每个节点都周期性的醒来，检查信道中是否有针对自己的扩展前导码。如果有，就保持接收状态；否则就返回到睡眠状态。



几个参数的设置

- 为避免分组空传，前导序列时间长度要大于接收方的睡眠时间。
- 侦听周期T_p是一个重要参数，直接影响节点的能量消耗。
  - 如果T_p设置的过小，接收节点就会因频繁的唤醒侦听而浪费能量。
  - 如果T_p设置的过大，发送节点就需要设置很长的前导分组，从而增加能量消耗。

BMAC信道评估机制

- BMAC协议还设计了一套改进的空闲信道评估机制（CCA），对信道状态进行准确感知。
  - 通过多次采样，确定噪声基准值，并动态更新；
  - 发送数据前，对信道进行再次采样，如果在多个接收信号采样中检测到有明显低于噪声基准的采样值（即孤立点），就认为信道空闲；反之则忙。
- 低功耗侦听机制消除了节点间通过发送SYNC消息建立虚拟簇的缺点，**付出的代价是在每个数据包发送之间发送扩展前导分组。**
- 当网络流量业务较低时，前导采样比固定占空比的休眠/唤醒模式更节省能量。

分配型MAC协议
DMAC

DMAC的原理

- 大部分的传感器网络是数据汇聚型网络。
- 普通节点采集数据后以树型结构向sink节点汇聚。
- 在SMAC和TMAC思想的基础上，可以采用对休眠和传输数据进行预分配的方式来避免延迟。
- 通过巧妙的预先安排，使一个节点向它的上级发送数据时该节点正好醒来。

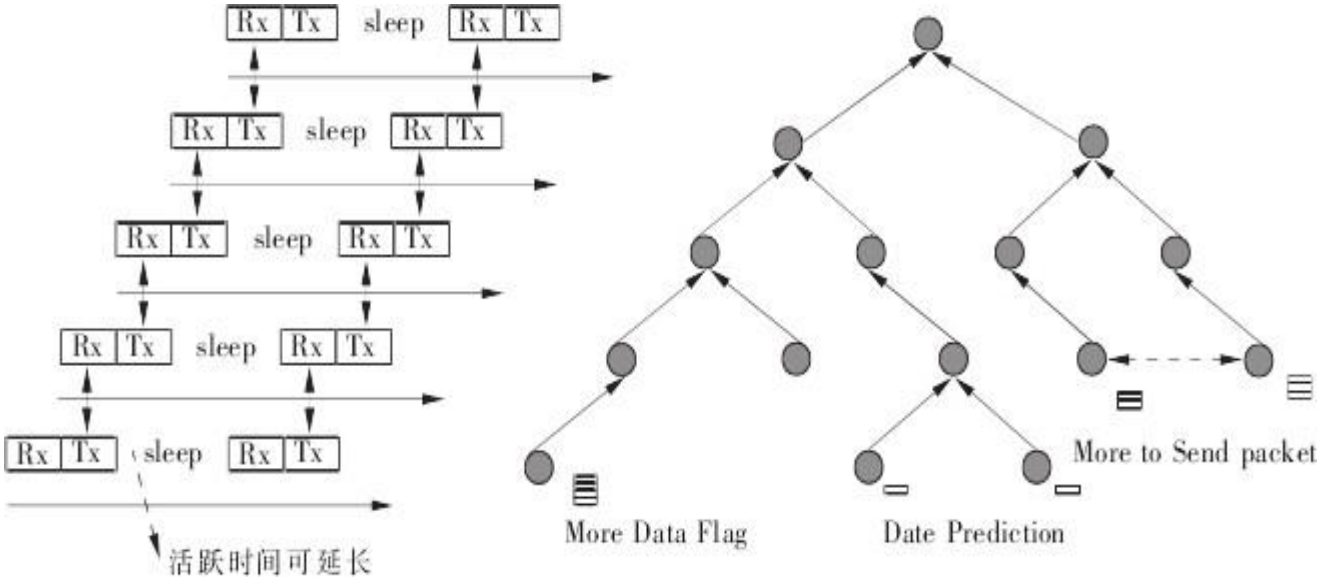
SMAC仍有不足！

- DMAC协议深入分析了SMAC协议中的监听睡眠调度机制的缺点，同步的睡眠会增加多跳传输的延迟，同步的监听和竞争信道会增加冲突的可能。
  - 通过同步形成的虚拟簇是双刃剑，既方便了节点之间相互通信，但会增大冲突的可能性，并加大传输延迟。
- SMAC协议虽然又引入了NAV睡眠机制，但只减少了两跳延迟，数据在多跳传输到Sink的过程中仍会因中间节点的睡眠而中止。
- 为了解决这些问题，DMAC协议引入了一种交错的监听睡眠调度机制，保证数据在多跳路径上的连续传输。

交错唤醒机制

- 在一些传感器网络应用中,数据从多个数据源汇聚到一个Sink节点，数据传输的路径都包含在一个树状拓扑结构中。DMAC协议将其定义为**数据采集树**。在数据采集树上节点之间采用交错唤醒方法。每个间隔分为接收、发送和睡眠三个周期。

DMAC传输示意图


Rx
Tx
sleep
活跃时间可延长
More Data Flag
Date Prediction
More to Send packet

ACK（确认帧）机制

- DMAC协议中数据的传输没有采用RTS-CTS机制，减少了控制开销。采用ACK机制来保障可靠传输。如果节点发送完成后没有收到ACK，必须缓存该数据，并等到下个发送周期再重传，通常重传次数超过三次就丢弃该数据包。

发送周期和接收周期的长度

- 为了减少在发送周期中处于树中同一深度的节点之间的碰撞，每个节点在发送数据之前先退避一个固定时间**BP (Backoff Period)**，然后在竞争时间窗口 **CW (Contend Window)**中再退避一个随机时间。
- 接收到数据的节点在等待一个短周期**SP (Short Period)**后回复一个**ACK**应答。
- 发送周期和接收周期的长度**u**可以由下式得出:

  u=BP+CW+DATA+SP+ACK

  式中**DATA**为数据包的传输时间，**ACK**为**ACK**帧的传输时间。

DMAC其它机制

- 自适应占空比机制
- 数据预测机制
- MTS（More To Send）帧机制
- 上述机制的功能：
  - 通过引入自适应占空比机制，DMAC协议能根据网络数据流量动态调整占空比；通过引入数据预测和MTS机制，DMAC协议能降低干扰造成的传输延迟。

DMAC小结

- DMAC协议是一种针对树状数据采集网络提出的能量高效、低延迟的MAC协议。DMAC协议根据节点在数据采集树上的深度为节点分配交错的活动/睡眠周期，在占空比方式下避免了数据多跳传输中的睡眠延迟。
- 由于DMAC协议提出了一系列的假设，这在一定程序上限制了它的应用，但是对于符合假设的应用，DMAC协议能较好地满足性能需要。

混合型MAC协议与跨层设计

混合型MAC协议与跨层设计

- 混合型MAC协议：ZMAC，既有CSMA竞争机制，又包含基于时隙和时间帧（超帧）概念的TDMA分配机制，对竞争方式和分配方式进行组合。
  - 当网络的业务流量低时，Z-MAC采用CSMA方式，可以有效提高信道利用率并降低时延。
  - 当网络的业务流量高时，Z-MAC使用TDMA方式，可以有效减少冲突。
- MAC层与跨层设计：MINA结构，将网络自组织、MAC协议和路由协议联合起来进行跨层设计与优化。

本章总结

- 本章首先结合无线传感器网络的特点，对MAC协议的研究热点进行了综述，并对无线传感器网络MAC协议进行了分类整理。然后根据本章的分类，从协议背景、基本思想、关键技术、算法等方面出发，对每个协议进行了详细介绍，并对协议的特点进行了归纳总结。无线传感器网络是面向应用的网络，不同的应用场景对网络性能会有不同的侧重，从而对MAC协议的要求也会不同。从对诸多的研究成果进行总结可以发现，仅仅通过MAC协议的设计来优化性能很难实现整体的性能优化，通过跨层设计实现分层的融合是一个值得深入研究的方向。