LỜI CẢM ƠN

Trước hết em xin gửi lời cảm ơn chân thành tới các các thầy cô giáo Khoa Công nghệ thông tin cùng toàn thể các thầy cô giáo trường Đại học Dân lập Hải Phòng, những người đã dạy dỗ, trang bị cho em những kiến thức cơ bản, cần thiết trong những năm học vừa qua, nhất là ThS.Nguyễn Trọng Thể, giáo viên Khoa Công nghệ thông tin, trường Đại học dân lập Hải Phòng, người đã nhiệt tình giúp đỡ, chỉ bảo để em có thể hoàn thành đề tài tốt nghiệp của mình.

Đặc biệt em xin bày tỏ lòng biết ơn sâu sắc tới PGS.TS Vương Đạo Vy, Khoa Điện tử viễn thông, trường Đại học Công nghệ - Đại học Quốc gia Hà Nội, người đã hướng dẫn chỉ bảo tận tình cho em trong suốt thời gian làm đề tài tốt nghiệp.

Cuối cùng em xin gửi lời cảm ơn tới gia đình, bạn bè - những người đã ủng hộ, quan tâm, giúp đỡ, động viên em trong suốt thời gian qua và là chỗ dựa vững chắc giúp cho em có thể hoàn thành đề tài tốt nghiệp.

Em xin chân thành cảm ơn!

Hải Phòng, tháng 07 năm 2010

Sinh viên

Nguyễn Công Tiến

MỤC LỤC

LỜI NÓI ĐẦU

Trong những năm gần đây, cùng với sự phát triển mạnh mẽ của khoa học kỹ thuật nói chung và công nghệ thông tin nói riêng, mạng cảm biến không dây (Wireless Sensor Networks – WSN) là một trong những lĩnh vực hiện đang được rất nhiều nơi trên thế giới nghiên cứu và phát triển. Nghiên cứu về WSN mở ra những ứng dụng đem lại lợi ích thiết thực cho đời sống con, giúp con người không mất quá nhiều công sức, nhân lực nhưng vẫn mang lại hiệu quả cao cho công việc. Những ứng dụng của WSN trong tương lai không xa sẽ trở thành một phần không thể thiếu trong đời sống con người nếu chúng ta có thể phát huy hết được các ưu điểm của nó. Sức mạnh của WSN nằm ở chỗ khả năng triển khai một số lượng lớn các thiết bị nhỏ có thể tự thiết lập cấu hình hệ thống. Sử dụng những thiết bị này để theo dõi theo thời gian thực, để giám sát điều kiện môi trường, để theo dõi cấu trúc hoặc tình trạng thiết bị….

Mặc dù có rất nhiều ưu điểm nhưng WSN vẫn còn nhiều nhược điểm, hạn chế cần được giải quyết. Một trong những thách thức lớn nhất của WSN chính là nguồn năng lượng bị giới hạn làm ảnh hưởng tới thời gian sống của các nút mạng. Các nút mạng cảm biến lại được phân bố và hoạt động với số lượng lớn ở những vùng địa lý và môi trường khác nhau nên việc nạp lại hay thay thế năng lượng cho các nút mạng là vô cùng khó khăn. Vì vậy mà rất nhiều nghiên cứu hiện nay đang tập trung vào vấn đề làm thế để tăng hiệu quả năng lượng của WSN trong từng lĩnh vực khác nhau.

Nhận thấy được sự quan trọng của việc tăng hiệu quả năng lượng cho WSN nên trong khóa luận này em sẽ tập trung vào nghiên cứu tìm hiểu về vấn đề: ***"Thâm nhập môi trường (MAC), hiệu quả năng lượng cho các nút mạng cảm biến không dây"***. Nội dung khóa luận này bao gồm 3 chương, phần mở đầu, phần kết luận và tài liệu tham khảo:

Chương 1: Tổng quan về mạng cảm biến không dây: giới thiệu một cách tổng quan về WSN, các ưu - nhược điểm và các ứng dụng đang được triển khai trong đời sống con người.

Chương 2: Giao thức điều khiển thâm nhập môi trường trong mạng cảm biến không dây: tìm hiểu về các thông số cần thiết khi thiết kế các giao thức thâm nhập môi trường (MAC), các nguyên nhân gây ra sự hao phí năng lượng và giới thiệu một số giao thức MAC phổ biến.

Chương 3: Thực nghiệm mô phỏng giải thuật thâm nhập môi trường cho mạng cảm biến không dây: tập trung tìm hiểu về phương pháp lập lịch tập trung và tiến hành làm thực nghiệm đo kiểm tính toán hiệu quả năng lượng của giao thức này.

Phần kết luận: tổng kết đánh giá lại những vấn đề đã thực hiện và kết quả đạt được.

CHƯƠNG 1: TỔNG QUAN VỀ MẠNG CẢM BIẾN KHÔNG DÂY (WIRELESS SENSOR NETWORK -WSN)

1. 1. Định nghĩa:

Mạng cảm biến không dây (Wireless Sensor Network - WSN) là một mạng không dây phân tán rộng bao gồm nhiều nút cảm biến được liên kết với nhau bằng sóng vô tuyến. Các nút cảm biến thường là các thiết bị đơn giản sử dụng các vi điều khiển, cảm biến, bộ truyền tín hiệu sóng radio,… với kích thước rất nhỏ, tiêu thụ năng lượng ít, tự tổ chức, giá thành thấp dùng để đo các dữ liệu và truyền thông không dây giữa các nút trong mạng. Mỗi nút của mạng có thể hoạt động độc lập để tiến hành đo các thông số khác nhau của môi trường như: nhiệt độ, độ ẩm, áp suất, ánh sáng, độ ồn, … Thay vì gửi số liệu thô tới nút đích thì các nút cảm biến với bộ vi xử lý bên trong có thể tiến hành xử lý đơn giản và gửi dữ liệu đã được xử lý theo yêu cầu. Mạng cảm biến không dây ra đời nhằm đáp ứng cho nhu cầu thu thập thông tin về môi trường tại một tập hợp các điểm xác định trong một khoảng thời gian nhất định nhằm phát hiện xu hướng hoặc quy luật vận động của môi trường.

1.2. Các thành phần của mạng cảm biến không dây:

1.2.1. Nút cảm biến:

Cấu tạo cơ bản của một nút cảm biến gồm 4 thành phần chính:

- Bộ phận cảm biến (Sensing unit): bao gồm cảm biến và bộ phận chuyển đổi tín hiệu tương thành tín hiệu số (Analog to Digital Converter – ADC). Bộ cảm biến dựa trên những thông số thu được từ môi trường sản sinh ra tín hiệu tương tự, những tín hiệu này được chuyển sang tín hiệu số bằng bộ ADC rồi sau đó được đưa vào đơn vị xử lý.

- Đơn vị xử lý (Processing unit): thường được kết hợp với một bộ lưu trữ nhỏ (Storage unit) quản lý các thủ tục làm cho các nút kết hợp với nhau để thực hiện các nhiệm vụ định sẵn.

- Bộ phận truyền nhận (Transceiver unit): kết nối nút với mạng.

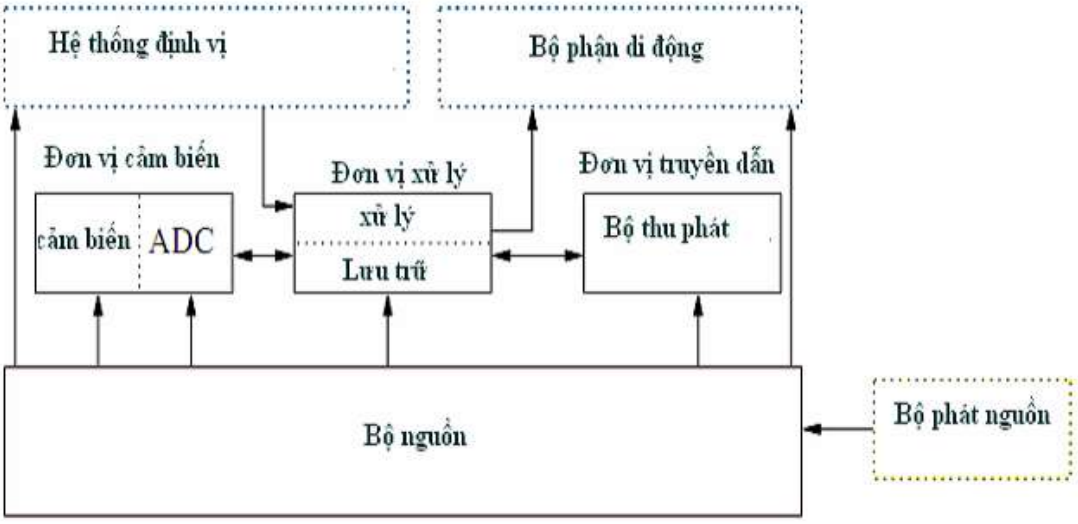
- Bộ nguồn (Power Unit): là một trong các thành phần quan trọng nhất của một nút mạng vì nó cung cấp năng lượng hoạt động cho mọi hoạt động của nút.

Ngoài ra, tùy theo mục đích mà các nút cảm biến có thể được bổ sung thêm những thành phần khác như:

- Hệ thống định vị (Location finding System): hầu hết kỹ thuật định tuyến và những nhiệm vụ cảm biến của mạng đòi hỏi có độ chính xác cao về vị trí thì nút cảm biến phải được gắn thêm bộ phận này.

- Bộ phận quản lý di động (Mobilizer): tùy thuộc vào ứng dụng mà nút cảm biến có thể được trang bị thêm bộ phận này để quản lý chuyển động khi nó được yêu cầu thực hiện nhiệm vụ định trước.

- Bộ thu phát nguồn (Power Generator): bộ nguồn thường được hỗ trợ bởi các bộ phận tiếp năng lượng như pin mặt trời.



Hình 1.1: Cấu tạo nút cảm biến.

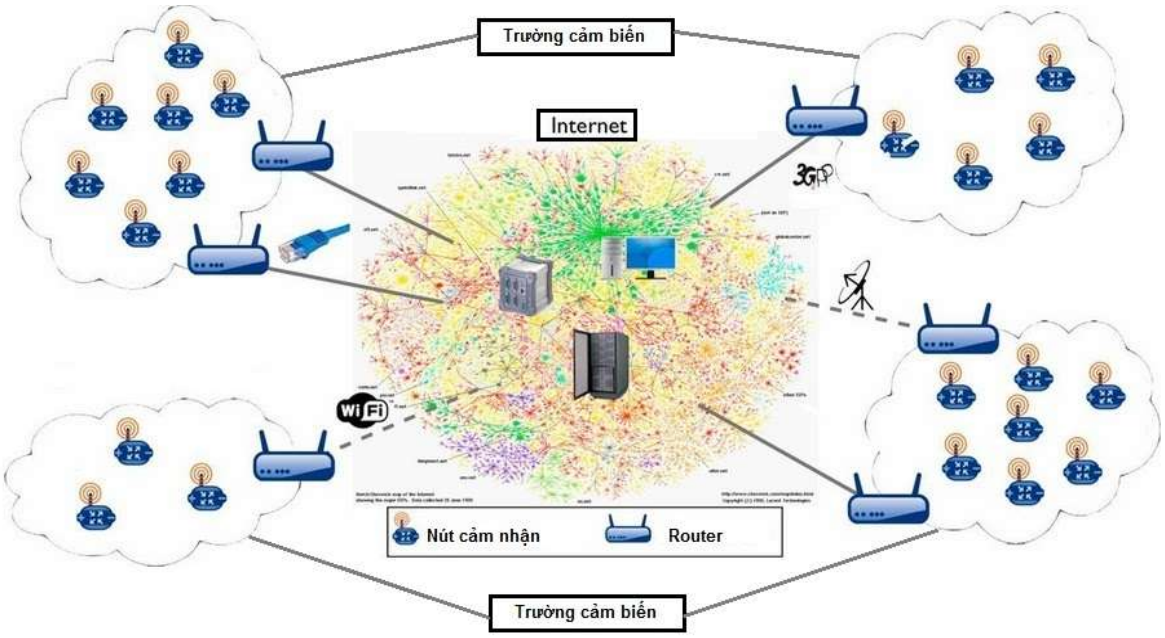
Tất cả các bộ phận này được tích hợp trong một mô đun với kích thước nhỏ chỉ bằng hộp diêm hoặc có khi nhỏ hơn $1cm^3$ và cần phải thỏa mãn các yêu cầu như: tiêu thụ rất ít năng lượng, hoạt động ở mật độ cao, có giá thành thấp, có thể tự hoạt động và thích ứng với sự biến đổi của môi trường.

Những nút cảm biến thường là không tác động được, tuổi thọ của một mạng cảm biến phụ thuộc vào tuổi thọ của những nguồn cung cấp năng lượng.

Vì kích thước giới hạn, năng lượng của nút cảm biến cũng trở thành một tài nguyên khan hiếm.

1.2.2. Mạng cảm biến:

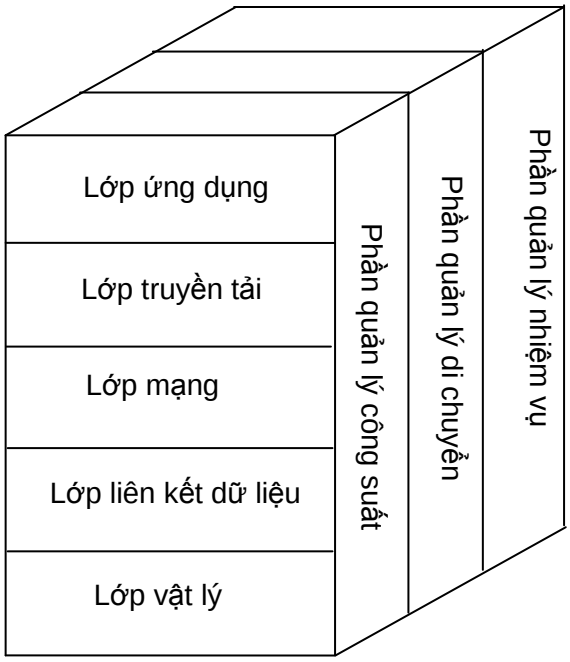
a. Mạng cảm biến: Bao gồm một số lượng lớn các nút cảm biến, các nút này thường được phân bố trong trường cảm biến. Mỗi nút có khả năng thu thập số liệu và chọn đường để chuyển số liệu tới nút gốc bằng việc chọn đường theo đa bước nhảy. Nút gốc có thể liên lạc với nơi quản lý nhiệm vụ thông qua mạng Internet hoặc vệ tinh. Việc thiết kế mạng cảm biến không dây khác hẳn các mạng truyền thống khác do các nút cảm biến có giới hạn về tài nguyên đặc biệt là năng lượng rất khắt khe và còn phụ thuộc vào nhiều yếu tố khác như: khả năng chịu lỗi, khả năng mở rộng, giá thành sản xuất, rằng buộc về phần cứng, cấu hình mạng, môi trường hoạt động, phương tiện truyền dẫn, sự tiêu thụ năng lượng.



Hình 1.2: Cấu trúc mạng cảm biến

b. Kiến trúc giao thức mạng cảm biến: Trong hình 1.3 là kiến trúc giao thức được sử dụng cho mạng cảm biến. Kiến trúc này bao gồm các lớp và các mặt phẳng quản lý. Các mặt phẳng này quản lý để các nút có thể làm việc cùng

nhau theo cách có hiệu quả nhất, định tuyến dữ liệu trong mạng cảm biến không dây và chia sẻ tài nguyên giữa các nút cảm biến.



Hình 1.3: Kiến trúc giao thức của mạng cảm biến

- Lớp vật lý: có nhiệm vụ lựa chọn tần số, tạo ra tần số sóng mang, phát hiện tín hiệu, điều chế và mã hóa tín hiệu

- Lớp liên kết dữ liệu: có nhiệm vụ ghép các luồng dữ liệu, phát hiện các khung (frame) dữ liệu, cách truy cập đường truyền và điều khiển lỗi.

- Lớp mạng: quan tâm đến việc chọn đường dữ liệu được cung cấp bởi lớp truyền tải

- Lớp truyền tải: giúp duy trì luồng số liệu nếu ứng dụng mạng cảm biến yêu cầu. Nó chỉ cần thiết khi hệ thống có kế hoạch được truy cập thông qua mạng Internet hoặc các mạng bên ngoài khác.

- Lớp ứng dụng: tuỳ theo nhiệm vụ cảm biến, các loại phần mềm ứng dụng khác nhau có thể được xây dựng và sử dụng ở lớp này.

- Mặt phẳng quản lý công suất: điều khiển việc sử dụng công suất của nút cảm biến. Ví dụ như khi mức công suất của nút cảm biến thấp, nút cảm biến phát quảng bá tới các nút lân cận để thông báo nó có mức công suất thấp và không thể tham gia vào các bản tin chọn đường. Công suất còn lại sẽ được dành riêng cho nhiệm vụ cảm biến.

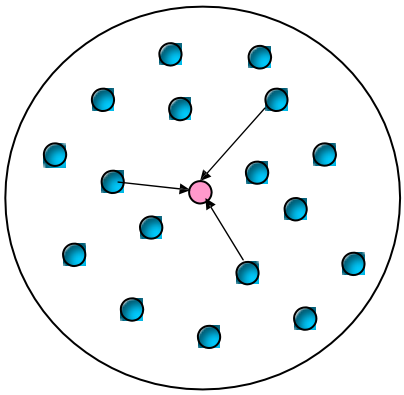
- Mặt phẳng quản lý di chuyển: có nhiệm vụ phát hiện và đăng ký sự chuyển động của các nút. Từ đó các nút có thể theo dõi xem ai là nút hàng xóm của chúng. Nhờ đó các nút cảm biến có thể cân bằng giữa công suất của nó và nhiệm vụ thực hiện.

- Mặt phẳng quản lý nhiệm vụ: cân bằng và sắp xếp nhiệm vụ cảm biến giữa các nút trong một vùng xác định. Không phải tất cả các nút cảm biến trong vùng đó điều phải thực hiện nhiệm vụ cảm biến tại cùng một thời điểm. Kết quả là một số nút cảm biến thực hiện nhiệm vụ nhiều hơn các nút khác tuỳ theo mức công suất của nó.

c. Hai cấu trúc đặc trưng của mạng cảm biến:

- Cấu trúc phẳng:

Trong cấu trúc phẳng (flat architecture) (hình 1.4), tất cả các nút đều ngang hàng và đồng nhất trong hình dạng và chức năng. Các nút giao tiếp với *sink* qua multihop sử dụng các nút ngang hàng làm bộ tiếp sóng. Với phạm vi truyền cố định, các nút gần sink hơn sẽ đảm bảo vai trò của bộ tiếp sóng đối với một số lượng lớn nguồn. Giả thiết rằng tất cả các nguồn đều dùng cùng một tần số để truyền dữ liệu, vì vậy có thể chia sẻ thời gian. Tuy nhiên cách này chỉ có hiệu quả với điều kiện là có nguồn chia sẻ đơn lẻ, ví dụ như thời gian, tần số…

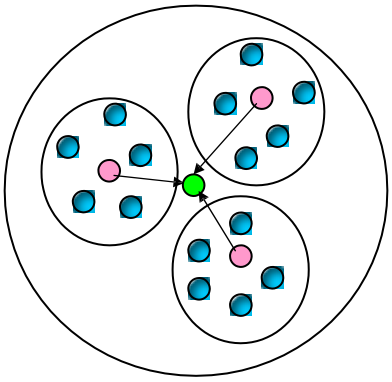


Hình 1.4: Cấu trúc phẳng của mạng cảm biến

- Cấu trúc tầng:

Trong cấu trúc tầng (tiered architecture) (hình 1.5), các cụm được tạo ra giúp

các tài nguyên trong cùng một cụm gửi dữ liệu single hop hay multihop (tùy thuộc vào kích cỡ của cụm) đến một nút định sẵn, thường gọi là nút chủ (cluster head). Trong cấu trúc này các nút tạo thành một hệ thống cấp bậc mà ở đó mỗi nút ở một mức xác định thực hiện các nhiệm vụ đã định sẵn.



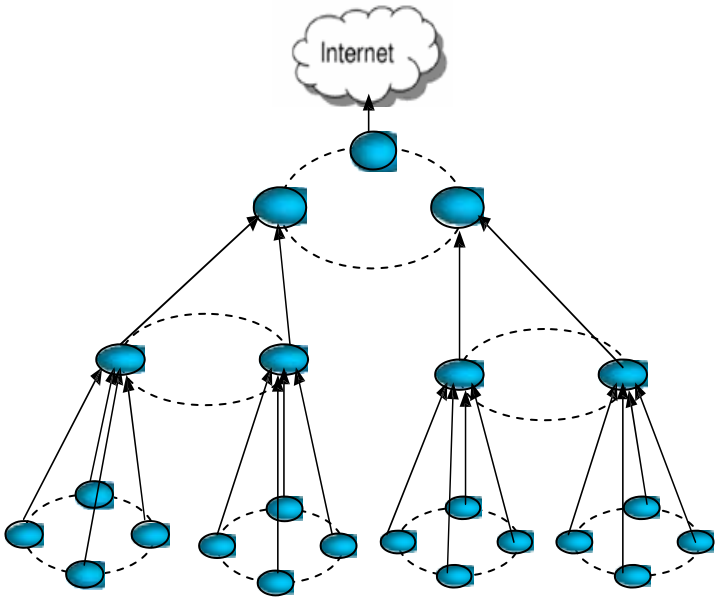
Hình 1.5: Cấu trúc tầng của mạng cảm biến

Trong cấu trúc tầng thì chức năng cảm nhận, tính toán và phân phối dữ liệu không đồng đều giữa các nút. Những chức năng này có thể phân theo cấp, cấp thấp nhất thực hiện tất cả nhiệm vụ cảm biến, cấp giữa thực hiện tính toán, và cấp trên cùng thực hiện phân phối dữ liệu (hình 1.6).

Cấp 2: Phân phối

Cấp 1 : Tính toán

Cấp 0: Cảm nhận



Hình 1.6: Cấu trúc mạng phân cấp chức năng theo lớp

Mạng cảm biến xây dựng theo cấu trúc tầng hoạt động hiệu quả hơn

cấu trúc phẳng, do các lý do sau:

-Cấu trúc tầng có thể giảm chi phí chi mạng cảm biến bằng việc định vị các tài nguyên ở vị trí mà chúng hoạt động hiệu quả nhất. Rõ ràng là nếu triển khai các phần cứng thống nhất, mỗi nút chỉ cần một lượng tài nguyên tối thiểu để thực hiện tất cả các nhiệm vụ. Vì số lượng các nút cần thiết phụ thuộc vào vùng phủ sóng xác định, chi phí của toàn mạng vì thế sẽ không cao. Thay vào đó, nếu một số lượng lớn các nút có chi phí thấp được chỉ định làm nhiệm vụ cảm biến, một số lượng nhỏ hơn các nút có chi phí cao hơn được chỉ định để phân tích dữ liệu, định vị và đồng bộ thời gian, chi phí cho toàn mạng sẽ giảm đi.

-Mạng cấu trúc tầng sẽ có tuổi thọ cao hơn cấu trúc mạng phẳng. Khi cần phải tính toán nhiều thì một bộ xử lý nhanh sẽ hiệu quả hơn, phụ thuộc vào thời gian yêu cầu thực hiện tính toán. Tuy nhiên, với các nhiệm vụ cảm biến cần hoạt động trong khoảng thời gian dài, các nút tiêu thụ ít năng lượng phù hợp với yêu cầu xử lý tối thiểu sẽ hoạt động hiệu quả hơn. Do vậy với cấu trúc tầng mà các chức năng mạng phân chia giữa các phần cứng đã được thiết kế riêng cho từng chức năng sẽ làm tăng tuổi thọ của mạng.

-Về độ tin cậy: mỗi mạng cảm biến phải phù hợp với với số lượng các nút yêu cầu thỏa mãn điều kiện về băng thông và thời gian sống. Với mạng cấu trúc phẳng, qua phân tích người ta đã xác định thông lượng tối ưu của mỗi nút trong mạng có n nút là $\frac{W}{\sqrt{n}}$, trong đó W là độ rộng băng tần của kênh chia sẻ. Do đó khi kích cỡ mạng tăng lên thì thông lượng của mỗi nút sẽ giảm về 0.

-Việc nghiên cứu các mạng cấu trúc tầng đem lại nhiều triển vọng để khắc phục vấn đề này. Một cách tiếp cận là dùng một kênh đơn lẻ trong cấu trúc phân cấp, trong đó các nút ở cấp thấp hơn tạo thành một cụm xung quanh trạm gốc. Mỗi một trạm gốc đóng vai trò là cầu nối với cấp cao hơn, cấp này đảm bảo việc giao tiếp trong cụm thông qua các bộ phận hữu tuyến. Trong trường hợp này, dung lượng của mạng tăng tuyến tính với số lượng các cụm, với điều

kiện là số lượng các cụm tăng ít nhất phải nhanh bằng *n*. Các nghiên cứu khác đã thử cách dùng các kênh khác nhau ở các mức khác nhau của cấu trúc phân cấp. Trong trường hợp này, dung lượng của mỗi lớp trong cấu trúc tầng và dung lượng của mỗi cụm trong mỗi lớp xác định là độc lập với nhau.

Tóm lại, việc tương thích giữa các chức năng trong mạng có thể đạt được khi dùng cấu trúc tầng. Đặc biệt người ta đang tập trung nghiên cứu về các tiện ích về tìm địa chỉ. Những chức năng như vậy có thể phân phối đến mọi nút, một phần phân bố đến tập con của các nút. Giả thiết rằng các nút đều không cố định và phải thay đổi địa chỉ một cách định kì, sự cân bằng giữa những lựa chọn này phụ thuộc vào tân số thích hợp của chức năng cập nhật và tìm kiếm. Hiện nay cũng đang có rất nhiều mô hình tìm kiếm địa chỉ trong mạng cấu trúc tầng.

1.3. Ứng dụng của mạng cảm biến không dây:

Tùy theo các loại cảm biến khác nhau được trang bị trong các nút mà mạng cảm biến mà nó thể được ứng dụng cho những mục đích khác nhau trong nhiều lĩnh vực như an ninh - quốc phòng, môi trường, y tế, gia đình, công nghiệp, thương mại….

*** Ứng dụng trong an ninh - quốc phòng:**

- Giám sát, phát hiện và thu thập thông tin về sự di chuyển, vũ khí, chất nổ…. của đối phương.
- Phát hiện phóng xạ hạt nhân.
- Bảo vệ an ninh cho các công trình trọng yếu
- Điều khiển tự động, kích hoạt các thiết bị quân sự, vũ khí, robot…
- Giám sát an ninh trong các khu dân cư, thương mại…
- Theo dõi biên giới kết hợp với vệ tinh.

*** Ứng dụng trong bảo vệ môi trường:**

- Theo dõi, nghiên cứu giám sát động vật hoang dã cần được bảo vệ.

- Theo dõi các yếu tố thời tiết như mưa bão, lụt lội, động đất, nước biển dâng…. để kịp thời cảnh báo phòng tránh giảm nhẹ thiệt hại.

- Phát hiện các yếu tố gây ảnh hưởng tới môi trường như ô nhiễm, chất thải, hóa chất độc hại….

*** Ứng dụng trong y tế:**

- Theo dõi tình trạng sức khỏe bệnh nhân, hỗ trợ chuẩn đoán điều trị và kịp thời cảnh báo cho các bác sĩ và nhân viên y tế khi có sự cố xảy ra.

- Giám sát dịch bệnh trong vùng dịch.

*** Ứng dụng trong gia đình:**

- Điều khiển từ xa các thiết bị trong gia đình.

- Giám sát tự động, cảnh báo an ninh cho gia đình từ xa khi có các vấn đề như cháy nổ, xâm nhập trái phép…

*** Ứng dụng trong lĩnh vực sản xuất công nghiệp, thương mại:**

- Điều khiển, quản lý tự động các khâu trong quá trình sản xuất sản phẩm.

- Giám sát qúa trình sản xuất, chất lượng nguyên vật liệu hay sản phẩm, kiểm soát môi trường làm việc, quản lý nhân viên.

- Cảnh báo, tự động xử lý các yếu tố nguy hiểm gây mất an toàn lao động.

- Xây dựng văn phòng thông minh.

- Quản lý cầu đường, các công trình xây dựng, các hệ thống viễn thông…

- Kiểm soát mức tiêu thụ điện năng, nước, khí gas, nguyên vật liệu…

1.4. Ưu điểm, nhược điểm của mạng cảm biến không dây:

1.4.1. Ưu điểm:

- Mạng cảm biến không dây có tính linh hoạt cao, không bị rằng buộc cố định về phân bố địa lý,

- Dễ triển khai, sử dụng trên những vùng có diện tích lớn và địa hình phức tạp.

- Có thể dễ dàng bổ sung hoặc thay thế các thiết bị tham gia vào mạng mà không phải cấu hình lại toàn bộ cấu trúc liên kết của mạng.

- Chi phí để triển khai lắp đạt mạng cảm biến không dây ngày càng giảm nhờ các tiến bộ về công nghệ đã làm giảm giá thành và kích thước các nút cảm biến.

- Theo dõi, giám sát các thay đổi của môi trường trong thời gian thực. Giúp cho hệ thống ngay lập tức có thể cảnh báo, xử lý được các tình huống bất lợi có thể xảy.

- Thay thế cho việc con người phải tự theo dõi giám sát trong những môi trường nguy hiểm, độc hại. Giảm bớt sức lực, nhân lực mà vẫn mang lại hiệu quả cao cho công việc.

1.4.2. Nhược điểm:

Mạng cảm biến không dây vẫn còn tồn tại những hạn chế cơ bản cần phải khắc phục sau:

- Tốc độ đường truyền chưa cao.
- Khả năng bị nhiễu và mất thông tin trên các vùng có địa hình xấu là lớn.
- Khả năng tính toán, bộ nhớ lưu trữ của nút cảm biến còn rất giới hạn.
- Giao thức quản lý mạng phức tạp.
- Đặc biệt là sự hạn chế về năng lượng sử dụng, công suất phát trong mạng.

1.5. Kết luận:

Chương này giới thiệu tổng quan về mạng cảm biến không dây và các ứng dụng của nó trong các lĩnh vực an ninh – quốc phòng, môi trường, y tế, gia đình, sản xuất – kinh doanh… Qua đó, chúng ta thấy được tầm quan trọng của mạng cảm biến trong đời sống của chúng ta ngày nay. Nắm được các ưu điểm để có thể triển khai một cách thích hợp cho các ứng dụng mới và nhược điểm cơ bản cần phải khắc phục để nâng cao thêm khả năng của mạng cảm biến không dây.

Đặc biệt trong đề tài này thì nhược điểm về vấn đề năng lượng của mạng cảm biến không dây sẽ được tìm hiểu và nghiên cứu cách khắc phục trong các giao thức điều khiển thâm nhập môi trường (MAC) cho mạng.

CHƯƠNG 2: GIAO THỨC ĐIỀU KHIỂN THÂM NHẬP MÔI TRƯỜNG TRONG MẠNG CẢM BIẾN KHÔNG DÂY

Một đặc tính cơ bản của mạng không dây là hoạt động trong môi trường không khí vốn đã được chia sẻ. Tất cả các giao thức điều khiển thâm nhập môi trường (Medium Access Control - MAC) cho mạng không dây quản lý việc sử dụng liên kết sóng vô tuyến để đảm bảo hiệu quả sử dụng của các băng thông chia sẻ. Những giao thức MAC được thiết kế cho mạng cảm biến không dây (WSN) ngoài mục đích giống các mạng không dây khác là quản lý hoạt động của sóng vô tuyến còn có một mục tiêu bổ sung là bảo toàn năng lượng cho các nút mạng. Nguyên nhân là do mạng cảm biến khác với các mạng không dây truyền thống trên một vài khía cạnh sau:

- Năng lượng cung cấp cho mọi hoạt động của các nút trong mạng cảm biến không dây đều dựa vào nguồn điện từ pin và rất khó để có thể nạp điện hoặc thay thế cho nguồn pin cho tất cả các nút.

- Các nút thường được triển khai theo kiểu phi cấu trúc tức là chúng phải tự tổ chức thành một mạng truyền thông.

- Nhiều ứng dụng của mạng cần phải sử dụng một số lượng lớn các nút có thể lên tới hàng nghìn, hàng triệu và mật độ của chúng sẽ thay đổi tùy theo những địa điểm và thời gian khác nhau.

- Hầu hết các lưu thông trong mạng được thúc đẩy bởi những sự kiện các nút cảm biến được từ môi trường.

Vì vậy, trong khi giao thức MAC truyền thống phải cân bằng tốc độ truyền tải, đỗ trệ và quan hệ công bằng giữa những nút, những người dùng, những ứng dụng khác nhau thì các giao thức MAC trong WSN đặt trọng tâm vào hiệu quả năng lượng là chủ yếu.

Trong chương này chúng ta sẽ tìm hiểu về một số giao thức MAC đã được đề xuất cho WSN.

2.1. Các thông số cần quan tâm khi thiết kế giao thức MAC cho WSN:

Có rất nhiều thông số cần quan tâm khi thiết kế giao thức MAC cho mạng không dây truyền thống nhưng trong WSN thì chúng ta chỉ quan tâm đến các vấn đề sau:

- Độ trễ (Delay): là lượng thời gian cần thiết để gói dữ liệu được xử lý trước khi nó được phát thành công. Sự quan trọng của đỗ trễ phụ thuộc vào ứng dụng của mạng cảm biến. Các ứng dụng như giám sát hoặc theo dõi thường yêu cầu rất khắt khe về độ trễ. Những ứng dụng này khi chưa cảm biến được sự kiện thì việc tiết kiệm năng lượng được coi trọng hơn độ trễ bởi vì khi đó thì mạng đang ở chế độ nghỉ và có rất ít dữ liệu được trao đổi trong mạng. Ngược lại, sau khi cảm biến xác định được sự kiện thì mục tiêu quan trọng của mạng cảm biến lúc này là phải hoạt động với độ trễ thấp.

- Thông lượng (Throughput): đề cập tới số lượng dữ liệu được chuyển thành công từ nơi gửi đến nơi nhận trong một khoảng thời gian cho trước. Nó thường được đo bằng thông điệp trên giây hoặc bit trên giây. Trong giao thức MAC của mạng không dây truyền thống thì vấn đề quan trọng là làm tối đa thông lượng kênh truyền với độ trễ nhỏ nhất. Nhưng cũng giống như với độ trễ, sự quan trọng của thông lượng trong WSN cũng phụ thuộc vào loại ứng dụng. Đối với những ứng dụng cảm biến mà yêu cầu thời gian sống của mạng dài thì giao thức MAC cần phải tăng độ trễ nhiều hơn và giảm thông lượng thấp hơn

- Độ chắc chắn (Robustness): là sự kết hợp của sự tin cậy, linh động và các yêu cầu phụ thuộc khác. Nó phản ánh mức độ của giao thức trong việc đối phó với lỗi và thông tin sai. Trong WSN thì để đạt được sự chắc chắn là rất khó khăn vì nó phụ thuộc vào tính chất của các yếu tố gây hư hỏng cho đường truyền và các nút.

- Khả năng mở rộng (Scalability): là khả năng của hệ thống đáp ứng được các thay đổi như kích thước mạng, mật độ và cấu trúc liên kết mạng. Do WSN được triển khai phi cấu trúc và các nút có khả năng tự tổ chức thành một mạng nên bất kỳ thời điểm nào người ta cũng có thể bổ sung thêm các nút mạng

để bổ sung cho mạng hoặc bổ sung thay thế các nút mạng không còn hoạt động. Một giao thức MAC tốt cần phải điều tiết một cách hợp lý những sự thay đổi này. Đây là một yêu cầu quan trọng khi thiết kế các giao thức MAC bởi vì WSN được triển khai phi cấu trúc và hoạt động trong những môi trường không chắc chắn với số lượng nút rất lớn, có thể lên tới hàng nghìn hàng triệu nút. Việc nhóm các nút cảm biến vào các cluster cho phép thiết kế các giao thức MAC với khả năng mở rộng cao.

- Tính ổn định (Stability): là khả năng của hệ thống sử dụng băng thông hoặc dung lượng kênh truyền như thế nào trong một khoảng thời gian dài hoạt động. Một giao thức MAC ổn định cần phải điều khiển được tải tức thời để không đạt tới mức tối đa dung lượng kênh truyền. Nó được coi là ổn định nếu lưu thông không bị tắc nghẽn khi tải tăng lên.

- Sự công bằng (Fairness): thể hiện khả năng của những người dùng, những nút hoặc những ứng dụng khác nhau cùng chia sẻ kênh truyền một cách công bằng. Đây là một thuộc tính quan trọng trong mạng không dây cũng như các mạng truyền thông khác khi mà mỗi người dùng đều muốn có cơ hội như nhau để gửi hoặc nhận dữ liệu cho những ứng dụng của mình. Tuy nhiên, trong WSN thì tất cả các nút hợp tác cho một nhiệm vụ chung đơn lẻ, nên có thể tại những thời điểm đặc biệt, một nút sẽ có thể có nhiều dữ liệu hơn để gửi so với các nút khác, điều này hiệu quả hơn là đối xử với mỗi nút công bằng tùy thuộc vào từng ứng dụng. Vì vậy nên sự công bằng với từng nút từng người dùng trong WSN trở nên ít quan trong hơn.

- Hiệu quả năng lượng (Energy efficiency): là một trong những yêu cầu quan trọng nhất trong việc thiết kế giao thức MAC cho WSN. Do các nút cảm biến hoạt động bằng pin nên việc thay thế hoặc nạp điện lại cho các nút này với số lượng lớn trên những vùng địa lý rộng là rất khó khăn và không khả thi. Trong thực tế thì hầu hết mục đích thiết kế các mạng cảm biến là xây dựng mạng bằng những nút đủ rẻ để vứt bỏ còn hơn là nạp lại để sử dụng nên việc kéo dài tuổi thọ của mỗi nút được xem vấn đề then chốt. Các nút cảm biến tiêu tốn

năng lượng chủ yếu cho việc thu phát sóng vô tuyến. Vì thế việc thiết kế các giao thức MAC cho WSN để trực tiếp điều khiển hoạt động thu phát sóng vô tuyến cần phải quan tâm nhất đến hiệu quả năng lượng bởi việc này sẽ ảnh hưởng đáng kể đến thời gian sống của các nút cũng như của toàn mạng.

Như vậy, khi thiết kế một giao thức MAC cho WSN thì các yếu tố quan trọng nhất là hiệu quả năng lượng và khả năng mở rộng, những thông số còn lại là thứ yếu và có thể không cần quan tâm.

2.2. Các nguyên nhân gây ra sự lãng phí năng lượng:

- Xung đột – tắc nghẽn (Collision – Obstruction): là nguyên nhân đầu tiên gây ra sự lãng phí năng lượng. Khi hai gói tin của 2 nút trong mạng được truyền cùng thời điểm thì sẽ xảy ra xung đột, chúng bị hỏng và bị loại bỏ. Yêu cầu truyền lại gói tin lần nữa sẽ làm hao phí thêm năng lượng của nút. Tuy nhiên có thể khi đó cả 2 nút cùng được yêu cầu truyền lại cùng một thời điểm nên rất có thể xảy ra xung đột một lần nữa. Điều này sẽ gây ra sự tắc nghẽn trong môi trường truyền khi mà 2 nút cứ cố gắng truyền lại mãi. Nếu cứ tiếp tục như vậy thì hao phí năng lượng sẽ là rất lớn vì việc thực hiện phát tín hiệu là công việc gây tiêu tốn năng lượng nhất của một nút. Do đó, tất cả các giao thức MAC đều phải cố gắng tránh gây ra xung đột bằng mọi cách để làm tăng hiệu quả năng lượng cho nút mạng.

-Nghe lỏm (Overhearing): do môi trường truyền dẫn của WSN là môi trường không khí nên vấn đề nghe lỏm sẽ xảy ra khi một nút nhận được những gói tin dành cho nút khác. Mà khi một nút muốn nhận được một gói tin thì nó sẽ phải tiêu tốn năng lượng cho việc này, tuy mức tiêu tốn không bằng việc phát sóng vô tuyến nhưng cũng vẫn gây tiêu hao năng lượng đáng kể của nút. Việc nghe lỏm những gói tin không cần thiết này cũng là một nhân tố chính gây ra việc tiêu hao năng lượng.

- Nghe nhàn rỗi (Idle listening): xảy ra khi nút nhận nghe kênh trong tất cả xem có dữ liệu không để nhận và cố gắng để không mất gói tin nào. Tuy

nhiên thì trong các mạng cảm biến không dây thì chỉ khi cảm biến được sự kiện xảy ra thì mới có dữ liệu được phát đi, còn lại thì không có dữ liệu trong một khoảng thời gian dài.Việc nghe nhàn rỗi trong một thời gian dài như vậy gây ra sự tiêu hao năng lượng cũng rất đáng kể.

- Xử lý các gói tin điều khiển (overhead): bao gồm việc gửi, nhận nghe các gói tin này của nút. Dù các gói tin điều khiển không không chuyên chở dữ liệu nhưng nó cũng làm tiêu thụ năng lượng đáng kể của nút mạng.

Một giao thức MAC thiết kế cho WSN cần giải quyết được các nguyên nhân trên bằng việc điều khiển các thành phần sóng vô tuyến để tránh hoặc giảm bớt tiêu hao năng lượng.

2.3. Các giao thức MAC trong WSN:

2.3.1. CSMA (Đa truy cập cảm biến sóng mang):

CSMA là các giao thức mà trong đó các trạm làm việc lắng nghe đường truyền trước khi đưa ra quyết định làm gì để tương thích với trạng thái đường truyền đó được gọi là giao thức có cảm nhận đường truyền. Nó hoạt động bằng cách: lắng nghe kênh truyền, nếu thấy kênh truyền rỗi thì bắt đầu truyền khung, nếu thấy bận thì trì hoãn lại việc gửi khung. Việc trì hoàn gửi khung này không thể kéo dài mãi nhưng vấn đề là không biết khi nào đường truyền rỗi nên người ta đã đưa ra ba giải pháp để giải quyết vấn đề này:

- Theo dõi không kiên trì (Non-persistent CSMA): Nếu đường truyền bận, đợi trong một khoảng thời gian ngẫu nhiên rồi tiếp tục nghe lại đường truyền.

- Theo dõi kiên trì (persistent CSMA): Nếu đường truyền bận, tiếp tục nghe đến khi đường truyền rỗi rồi thì truyền gói tin với xác suất bằng 1.

- Theo dõi kiên trì với xác xuất p (P-persistent CSMA): Nếu đường truyền bận, tiếp tục nghe đến khi đường truyền rỗi rồi thì truyền gói tin với xác suất bằng p.

Tuy nhiên, trong CSMA có thể phát sinh một vấn đề như sau: khi một nút vừa phát xong thì một nút khác cũng phát sinh yêu cầu phát khung và bắt đầu

nghe đường truyền. Nếu tín hiệu của nút thứ nhất chưa dến nút thứ hai, nút thứ hai sẽ cho rằng đường truyền đang rỗi và bắt đầu phát khung. Khi đó thì xung đột sẽ xảy ra làm cho khung bị mất và toàn bộ thời gian từ lúc xung đột xảy ra cho đến khi phát xong khung là lãng phí. Một vấn đề mới phát sinh nữa là các nút cần phải quan tâm theo dõi xung đột và sau khi phát hiện thì các nút sẽ phải làm gì.

Từ đó người ta xây dựng thêm giao thức mở rộng CSMA/CD (Đa truy cập cảm nhận sóng mang tránh xung đột). Giao thức này cơ bản là giống CSMA với cơ chế lắng nghe trước khi truyền nhưng được cải tiến thêm việc phát hiện xung đột và làm lại sau xung đột. Xung đột có thể được phát hiện bằng cách theo dõi năng lượng hay độ rộng của xung tín hiệu nhận được và đem so sánh với độ rộng của xung vừa truyền đi. Sau khi bị xung đột, nút sẽ chạy một thuật toán gọi là back-off dùng để tính toán lại lượng thời gian nó phải chờ trước khi gởi lại khung. Lượng thời gian này phải là ngẫu nhiên để các nút sau khi quay lại không bị xung đột với nhau nữa.

*** Vấn đề nút ẩn, nút hiện:**

CSMA truyền thống không cảnh báo được miền đụng độ và không hiệu quả trong các mạng không dây do có 2 vấn đề chính : vấn đề các nút ẩn và các vấn đề các nút hiện.

Vấn đề các nút ẩn minh hoạ ở hình 2.1(a), ở đây nút A truyền tới nút B. Nút C, nút mà nằm ngoài sóng của A, sẽ cảm nhận thấy kệnh truyền tới nút A đang rảnh và cũng bắt đầu truyền tới nút B. Trong trường hợp này CSMA không phát hiện cảnh báo xung đột được do A và C ẩn với nhau.

Vấn đề nút hiện được minh hoạ bởi hình 2.1 (b). Ở đây, trong khi nút B truyền tới nút A, nút C có một gói dành cho nút D. Tại vì nút C nằm trong vùng phủ sóng của nút B, nó sẽ cảm thấy là đường truyền đang bận và nó sẽ không truyền Tuy nhiên trên lý thuyết tại vì nút D nằm ngoài vùng phủ sóng của nút B, và A nằm ngoài vùng phủ sóng của C, có 2 phiên truyền mà không đụng độ với

nhau. Việc trì hoãn việc truyền bởi C sẽ làm lãng phí băng thông.

(a) (b)

Hình 2.1. Các vấn đề với CSMA căn bản trong môi trường không dây : (a) nút ẩn , (b) nút hiện

Có hai vấn đề cần quan tâm trong mỗi một cảm biến : ở nút ẩn vấn đề các gói bị đụng độ vì nút gửi không biết có nút khác đang sử dụng đường truyền, trong khi vấn đề của nút hiện có sự lãng phí cơ hội truyền một gói vì nhận biết sai lệch của truyền chống nhiễu. Vấn đề chính của lỗi này là không phải bộ truyền cảm nhận sóng mang mà là bộ thu. Một số giao tiếp giữa bộ truyền và bộ nhận cần phải có để giải quyết các vấn đề này.

2.3.2. S-MAC (Sensor-MAC):

S-MAC được giới thiệu bởi các tác giả: Wei Ye, Jonh Heidermann và Deborah Estrin tại Hội nghị INFOCOM lần thứ 21, năm 2002. SMAC là một trường hợp của đa phân chia thời gian thuần túy được xây dựng trên nền tảng của các giao thức cạnh tranh như 802.11, S-MAC cố gắng kế thừa sự linh hoạt, tính khả biến của giao thức trên nền cạnh tranh trong khi cải tiến tính hiệu quả sử dụng năng lượng trong mạng đa bước nhảy. Giao thức này cố gắng giảm bớt tiêu thụ năng lượng từ tất cả các nguồn được xác định là nguyên nhân gây tiêu hao năng lượng, đó là: nghe nhàn rỗi (idle listening), xung đột (collision), nghe lỏm (overhearing) và xử lý thông tin điều khiển (overhead). Để đạt được mục đích như thiết kế, S-MAC được thiết kế gồm có ba vấn đề chính là thực hiện chu kỳ thức - ngủ, tránh xung đột và nghe lỏm, xử lý thông điệp.

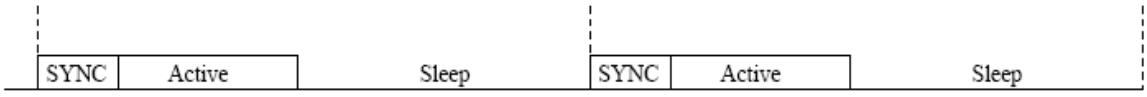
*** Thực hiện chu kỳ thức - ngủ:**

Trong những ứng dụng của WSN, nếu không xuất hiện sự kiện cảm biến thì các nút cảm biến thường ở trạng thái nhàn rỗi trong phần lớn thời gian. S-

MAC được thiết kế để giảm bớt thời gian thức bằng cách để cho nút cảm biến định kỳ chuyển sang trạng thái ngủ. Khi đó năng lượng của nút mạng sẽ được tiết kiệm một cách đáng kể do ở trong trạng thái ngủ nó sẽ tiêu tốn năng lượng ít hơn rất nhiều so với khi hoạt động.

a. Lược đồ cơ bản:

Mỗi nút cảm biến chuyển vào trạng thái ngủ trong một khoảng thời gian, sau đó tỉnh dậy và nghe xem liệu có nút nào muốn truyền tín hiệu tới nó. Trong thời gian ngủ, nút cảm biến tắt bộ phận thu phát vô tuyến và đặt thời gian để quay về trạng thái thức. Việc này tạo thành một chu kỳ là chu kỳ thức - ngủ và khoảng thời gian cho việc thức và ngủ có thể được lựa chọn theo những ứng dụng khác nhau.



Hình 2.2: Lược đồ S-MAC

Lược đồ trên yêu cầu có định kỳ sự đồng bộ giữa các nút cảm biến trong vùng tránh sai lệch thời gian. Ở đây có thể sử dụng hai kỹ thuật để đồng bộ thời gian: Thứ nhất, chúng trao đổi các thông số thời gian qua các gói tin *timestamps*, thời gian được đồng bộ là tương đối. Thứ hai, tăng khoảng thời gian nghe lên đáng kể so với thời gian bị sai lệch do lỗi. Ví dụ, khoảng thời gian nghe là 0.5s gấp 10^5 lần thời gian lệch. Như vậy yêu cầu điều kiện đồng bộ giữa các nút lân cận trong S-MAC không quá khắt khe. Tất cả các nút cảm biến đều tự do lập lịch cho mình chu kỳ thức-ngủ. Tuy nhiên, để giảm bớt phải xử lý những gói tin điều khiển, tốt hơn là để cho các nút trong vùng đồng bộ cùng nhau. Có nghĩa là chúng thức cùng lúc và chuyển sang trạng thái ngủ cùng lúc. Nhưng cũng cần chú ý trong một mạng đa bước nhảy không phải tất cả các nút lân cận có thể đồng bộ hóa cùng nhau. Hai nút lân cận A và B có thể có lịch khác nhau vì chúng tiến hành đồng bộ với những nút khác nhau, C và D (Hình 2.3).

Hình 2.3: Đồng bộ giữa các nút. Hai nút lân cận A, B có lịch khác nhau vì A đồng bộ với C, B đồng bộ với D

Các nút cảm biến trao đổi với nhau thông tin lịch làm việc của chúng bằng cách phát quảng bá cho tất cả các nút lân cận hiện thời. Điều này bảo đảm rằng tất cả các nút trong vùng vẫn có thể nói chuyện được với nhau dù chúng có lịch làm việc khác nhau. Ví dụ trong Hình 2.3, nếu nút A muốn nói chuyện với nút B, nó chỉ cần đợi cho đến khi B ở trạng thái thức. Nếu có nhiều nút trong vùng lân cận muốn nói chuyện với một nút, thì chúng cần tiến hành cạnh tranh chiếm đường truyền khi nút nhận ở trạng thái thức. Cơ chế cạnh tranh dành quyền truy nhập cũng giống chuẩn IEEE 802.11, sử dụng gói tin RTS *(Request to Send)* và CTS *(Clear to Send)*. Nút nào gửi gói tin RTS ra trước sẽ giành quyền truy nhập và nút nhận sẽ trả lời với một gói CTS. Sau đó chúng bắt đầu sự truyền dữ liệu, lúc này chúng không tuân theo lịch làm việc trước đó của chúng cho đến khi chúng kết thúc truyền dữ liệu.

Đặc trưng khác của lược đồ trên là nó hình thành những nút vào trong cấu trúc liên kết phẳng. Các nút cảm biến trong vùng lân cận tự do nói chuyện với nhau bất kể lịch làm việc nào mà chúng có. Các nút được đồng bộ tự hình thành một nhóm ảo. Lược đồ này khá dễ để làm thích nghi đối với mạng có thay đổi cấu trúc liên kết.

Mặt trái của lược đồ là sự gia tăng độ trễ do duy trì chu kỳ ngủ *(sleep)* của mỗi nút. Hơn nữa, độ trễ có thể tích lũy qua mỗi chặng (*hop)*, nên yêu cầu giới hạn độ trễ của ứng dụng tạo ra giới hạn thời gian ngủ trong chu kỳ làm việc của các nút cảm biến.

b. Tiến trình lựa chọn và duy trì lịch làm việc:

Trước khi bắt đầu chu kỳ thức-ngủ, mỗi nút cần phải chọn một lịch biểu làm việc (khi nào thức, khi nào ngủ) và trao đổi lịch này với các nút lân cận.

Mỗi nút duy trì một bảng lưu giữ tất cả các thời gian biểu của các nút lân cận mà nó biết. Quy trình chọn thời gian biểu của mỗi nút như sau:

Bước 1: đầu tiên nút cảm biến nghe trong một khoảng thời gian nhất định. Nếu nó không nghe thấy thời gian biểu từ nút khác, nó chọn ngẫu nhiên một khoảng thời gian để bắt đầu ngủ và tức thời quảng bá thời gian biểu của nó trong thông điệp SYNC, thông điệp này thông báo rằng nó sẽ ngủ sau khoảng thời gian t giây. Chúng ta gọi một nút trên là *nút đồng bộ (synchronizer)*, nó đã tự chọn cho mình một thời gian biểu độc lập và những nút khác sẽ phải đồng bộ theo nó.

Bước 2: nếu một nút nhận được một thời gian biểu của một nút lân cận trước khi tự chọn cho mình thì nó sẽ lấy thời gian biểu đó thiết lập thời gian biểu cho mình. Ta gọi nút như vậy là *nút đồng bộ theo (follower)*. Sau đó nó đợi một khoảng thời gian ngẫu nhiên t_d và phát quảng bá lại thời gian biểu này, và thông báo rằng nó sẽ ngủ sau $t - t_d$ giây nữa. Sở dĩ phải đợi ngẫu nhiên khoảng thời gian t_d để tránh xung đột, vì nhiều khả năng có nhiều nút đồng bộ theo một nút, khi quảng bá lại thời gian biểu của mình cùng một thời điểm sẽ xảy ra xung đột.

Bước 3: nếu một nút nhận được một thời gian biểu khác sau khi nó lựa chọn và quảng bá thời gian biểu của mình, nó sẽ chấp nhận cả hai (Ví dụ, nó sẽ lập lịch cho nó tỉnh dậy tại những thời điểm trong cả hai thời gian biểu, của chính nó và của nút lân cận). Nó quảng bá thời gian biểu của nó trước khi chuyển sang trạng thái ngủ.

Rất hiếm khi xảy ra các nút phải duy trì nhiều thời gian biểu. Các nút sẽ cố gắng chọn một thời gian biểu đã tồn tại trước khi tự chọn cho mình một thời gian biểu độc lập. Mặt khác, xảy ra trường hợp các nút lân cận thất bại trong việc khám phá, phát hiện ra nhau tại thời điểm ban đầu do xung đột khi quảng bá thời gian biểu, thì chúng vẫn có thể tìm thấy nhau trong chu kỳ kế tiếp.

Để minh họa giải thuật này, hãy xem xét một mạng gồm các nút có thể "nghe" thấy lẫn nhau. Đồng hồ hệ thống của một nút sẽ khởi động tính giờ trước và sự quảng bá sẽ đồng bộ tất cả trong thời gian biểu của nó. Nếu thay vì hai nút

độc lập đăng ký thời gian biểu (vì chúng không thể nghe thấy lẫn nhau, hoặc vì chúng tình cờ truyền ở tại thời điểm gần nhau), những nút trên nằm ở ranh giới giữa thời gian biểu sẽ chấp nhận cả hai. Với cách này, một nút chỉ cần gửi một lần cho một gói tin quảng bá. Điều bất lợi là những nút trên có ít thời gian hơn để ngủ và do đó sẽ tiêu thụ nhiều năng lượng hơn những nút khác.

Một tùy chọn khác để cho những nút trên vùng biên chấp nhận duy nhất một thời gian biểu là chấp nhận cái đến trước tiên. Khi nó biết thời gian biểu khác mà một số nút lân cận của nó theo, nó có thể vẫn còn nói chuyện với chúng. Tuy nhiên, với những gói quảng bá, nó cần gửi hai lần với hai thời gian biểu khác nhau. Ưu điểm của phương pháp này là các nút nằm trong vùng biên sẽ có cùng chu kỳ nghe ngủ với những nút khác.

c. Thực hiện đồng bộ:

Lược đồ thức-ngủ yêu cầu sự đồng bộ giữa những nút trong vùng lân cận. Việc các nút trong vùng lân cận định kỳ cập nhật lẫn nhau thời gian biểu của chúng là cần thiết để ngăn ngừa sự sai lệch thời điểm của chu kỳ nghe-ngủ.

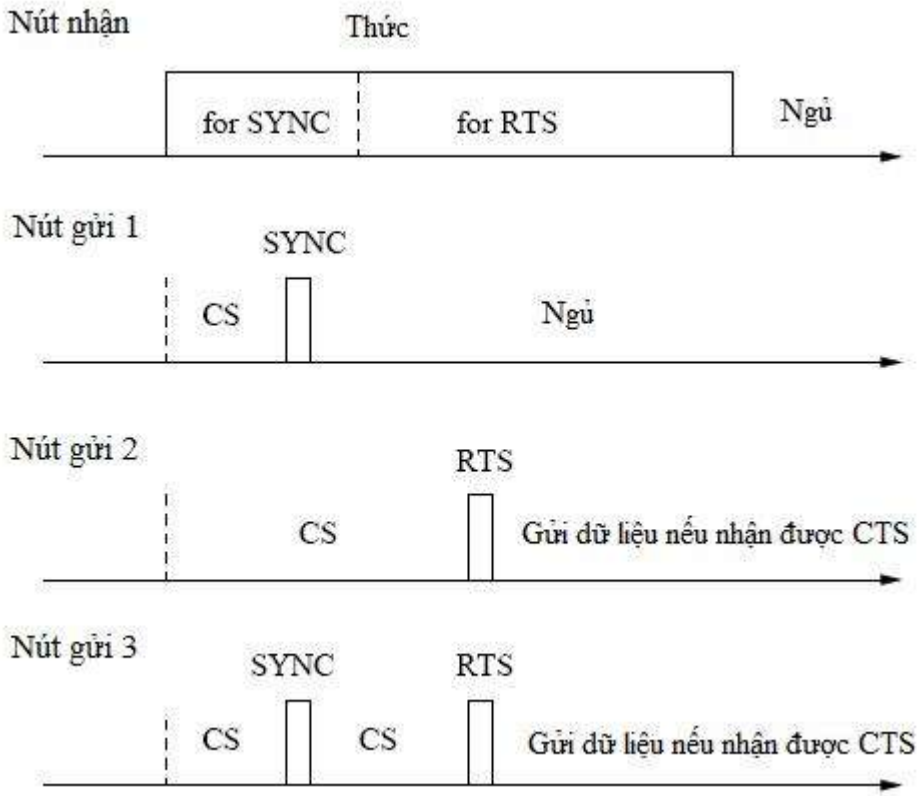
Việc cập nhật thời gian biểu được thực hiện bằng trao đổi gói tin đồng bộ SYNC. Gói tin SYNC rất ngắn, và bao gồm địa chỉ của nút gửi và thời điểm chuyển sang trạng thái ngủ tiếp theo của nó. Thời điểm ngủ tiếp theo liên quan đến thời điểm mà nơi gửi kết thúc truyền gói tin đồng bộ SYNC, cũng xấp xỉ khi nút nhận nhận được gói tin (khi độ trễ truyền ngắn). Những nút nhận sẽ điều chỉnh đồng hồ của chúng ngay sau khi nhận được gói tin đồng bộ. Nút cảm biến sẽ chuyển sang trạng thái ngủ khi đồng hồ của nó kết thúc tính giờ, báo đến thời điểm ngủ.

Để một nút nhận được cả những gói đồng bộ lẫn những gói dữ liệu, chúng ta chia khoảng thức *(active time)* của nó thành hai phần. Phần đầu tiên để nhận những gói tin đồng bộ, phần hai để nhận những gói RTS (Hình 2.4). Mỗi phần được chia tiếp thành nhiều khe thời gian cho những nút gửi để thực hiện cảm nhận sóng mang. Ví dụ, nếu một nút gửi muốn gửi một gói tin đồng bộ thì nó khởi động cảm nhận sóng mang khi nút nhận bắt đầu nghe. Nó ngẫu nhiên lựa

chọn một khe thời gian để kết thúc cảm nhận sóng mang. Nếu nó không phát hiện ra bất kỳ sự truyền nào vào khoảng cuối khe, thì nó chiếm được đường truyền và bắt đầu gửi gói tin đồng bộ của nó ở tại thời điểm ấy. Việc thực hiện truyền gói dữ liệu cũng được thực hiện tương tự.

Hình 2.4 cũng thể hiện mối quan hệ định thời của ba trường hợp có thể khi một nút gửi thực hiện truyền tới một nút nhận. CS là cảm ứng sóng mang. Trong lược đồ, Nút gửi 1 chỉ gửi một gói tin đồng bộ SYNC. Nút gửi 2 chỉ muốn gửi dữ liệu. Nút gửi 3 gửi một gói tin đồng bộ và một gói tin RTS.

Mỗi nút định kỳ quảng bá những gói tin đồng bộ tới các lân cận của nó kể cả khi nó không có nút đồng bộ theo. Điều này cho phép một nút mới gia nhập nhóm lân cận đã hình thành trước đó. Nút mới thực hiện thủ tục để chọn một thời gian biểu có sẵn làm thời gian biểu của nó. Quãng thời gian nghe đủ dài để nó có khả năng học và theo một thời gian biểu có sẵn trước khi nó tự chọn cho mình một thời gian biểu độc lập.



Hình 2.4: Quan hệ định thời giữa nút nhận và các nút gửi.

Mỗi nút định kỳ quảng bá những gói tin đồng bộ tới các lân cận của nó kể cả khi nó không có nút đồng bộ theo. Điều này cho phép một nút mới gia nhập nhóm lân cận đã hình thành trước đó. Nút mới thực hiện thủ tục để chọn một thời gian biểu có sẵn làm thời gian biểu của nó. Quãng thời gian nghe đủ dài để nó có khả năng học và theo một thời gian biểu có sẵn trước khi nó tự chọn cho mình một thời gian biểu độc lập.

*** Tránh xung đột và nghe lỏm:**

Tránh xung đột là một nhiệm vụ cơ bản của giao thức MAC. S-MAC sử dụng một lược đồ tránh xung đột trên nền cạnh tranh. Khi một nút phát đi một gói tin, gói tin đó được thu bởi tất cả các nút lân cận của nó mặc dù chỉ một trong số chúng là nút nhận, đó chính là nghe lỏm. Phải nghe lỏm làm cho giao thức MAC trên nền cạnh tranh kém hiệu quả về tiết kiệm năng lượng nên nó cần phải tránh.

a. Tránh xung đột:

Khi nhiều nút có nhu cầu gửi số liệu vào cùng một thời điểm, chúng cần cạnh tranh để quyết định một nút được quyền gửi (chiếm đường truyền). Trong số những giao thức cạnh tranh, 802.11 thực hiện rất tốt việc tránh xung đột. S-MAC sử dụng các kỹ thuật như chuẩn 802.11, bao gồm cảm nhận sóng mang vật lý, cảm nhận sóng mang ảo lẫn thực hiện trao đổi RTS/CTS.

Có một trường độ dài phát (*duration field)* trong mỗi gói tin được truyền đi để chỉ rằng việc truyền này sẽ duy trì trong thời gian bao lâu. Như vậy nếu một nút nhận được một gói tin dành cho nút khác, thì nó biết việc nó phải giữ yên lặng bao lâu. Nút ghi giá trị này trong một biến gọi là vectơ thời gian chiếm giữ mạng (Network allocation Vector - NAV) và đặt một đồng hồ tính giờ cho nó. Vào mọi thời điểm khi đồng hồ NAV hoạt động, nút cảm biến tuần tự giảm giá trị của NAV cho đến khi nó về giá trị 0. Khi một nút có dữ liệu để gửi, đầu tiên nó kiểm tra đồng hồ NAV. Nếu giá trị của NAV khác 0, thì nút xác định

rằng đường truyền bận và sẽ không thực hiện phát dữ liệu. Kỹ thuật này được gọi là cảm nhận sóng mang ảo *(Vitual Carrier Sense)*.

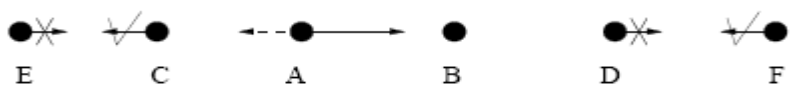
Cảm nhận sóng mang vật lý được thực hiện ở tại lớp vật lý bằng cách thực hiện nghe kênh để truyền. Thời gian ngẫu nhiên cho việc cảm nhận sóng mang rất quan trọng cho viêc tránh xung đột. Đường truyền chỉ được xác định là rỗi nếu cả cảm nhận sóng mang vật lý lẫn cảm nhận sóng mang ảo đều xác định đường truyền rỗi.

Tất cả các nút gửi thực hiện cảm nhận sóng mang trước khi bắt đầu phát dữ liệu. Nếu một nút thất bại trong việc thăm dò đường truyền, thì nó chuyển sang trạng thái ngủ và thức giấc tại thời điểm nút nhận ở trạng thái nghe và đường truyền rỗi trở lại. Những gói tin quảng bá được gửi mà không sử dụng kỹ thuật RTS/CTS. Những gói tin Unicast sẽ theo tuần tự RTS/CTS/Data/ACK giữa nút gửi và nút nhận.

b, Tránh nghe lỏm:

Ở chuẩn 802.11, mỗi nút duy trì trạng thái nghe cho việc truyền tới tất cả các nút lân cận của nó để thực hiện có hiệu quả việc cảm nhận sóng mang ảo. Kết quả là mỗi nút phải nghe thừa nhiều gói không gửi cho nó. Đây là một trong những nguyên nhân chính cho việc tiêu phí năng lượng không cần thiết, đặc biệt khi mật độ nút lớn và lưu lượng mạng tăng.

S-MAC được thiết kế với mục tiêu cố gắng tránh nghe thừa bằng cách để cho những nút có khả năng gây nhiễu không tham gia vào quá trình truyền phát dữ liệu, chuyển sang trạng thái ngủ sau khi chúng nhận được một gói RTS hoặc CTS. Khi những gói dữ liệu luôn dài hơn gói tin điều khiển, cách tiếp cận là ngăn cản các nút lân cận nghe thừa những gói dữ liệu dài và sử dụng gói tin ACK theo sau. Phần tiếp theo sẽ mô tả cách truyền có hiệu quả một gói tin dài kết hợp tránh nghe thừa.



Hình 2.5: Thực hiện tránh nghe thừa. Nút nào nên chuyển tới trạng thái ngủ.

Trong Hình 2.4, nút A, B, C, D, E, Và F hình thành một mạng đa bước nhảy mà từng nút chỉ có thể nghe thông tin truyền từ lân cận hiện thời của nó. Giả thiết nút A đang truyền một gói dữ liệu tới nút B. Câu hỏi đặt ra những nút nào phải chuyển sang trạng thái ngủ.

Xung đột dễ xảy ra ở nút nhận, nút D cần phải ngủ vì sự truyền của nó ảnh hưởng tới sự tiếp nhận tín hiệu của B. Cũng dễ để nhận ra nút E và nút F không phát sinh nhiễu, vì vậy chúng không cần phải ngủ. Nút C có nên đi ngủ hay không? C là cách hai bước tới B, và sự truyền của nó không gây nhiễu tới sự tiếp nhận của B, như vậy nó tự do được phép truyền tới lân cận của nó, ví dụ như E. Tuy nhiên, C không thể nhận bất kỳ sự trả lời nào từ E, vì sự truyền của E xung đột với sự truyền của A tại nút C. Như vậy sự truyền của C đơn giản là một sự tiêu phí năng lượng. Tóm lại, tất cả lân cận tức thời của cả nút gửi và nút nhận cần phải chuyển trạng thái ngủ khi chúng nghe thấy gói RTS hoặc CTS cho đến khi sự truyền hiện thời kết thúc.

Mỗi nút duy trì NAV để chỉ báo hoạt động trong khu lân cận của nó. Khi một nút nhận một gói dành cho tới những nút khác, nó cập nhật NAV của nó tại trường *duration* trong định dạng gói tin. Một giá trị NAV lớn hơn 0 chỉ báo rằng có một nút đang gửi số liệu trong khu vực lân cận của nó. Giá trị NAV giảm dần theo thời gian. Như vậy một nút cần phải ở trạng thái ngủ để tránh nghe thừa khi giá trị NAV của nó khác 0.

*** Xử lý thông điệp:**

Truyền dữ liệu dài trong một gói tin thì chi phí cho việc truyền lại khi chỉ có một vài bít lỗi trong lần truyền đầu tiên là rất cao. Tuy nhiên, nếu chúng ta chia nhỏ thông điệp vào trong nhiều gói nhỏ độc lập, chúng ta phải xử lý quá nhiều gói tin điều khiển do vậy độ trễ truyền sẽ tăng.

S-MAC xử lý vấn đề trên bằng cách chia nhỏ thông điệp dài thành nhiều phân đoạn nhỏ và truyền chúng trong một cụm (burst) nhưng chỉ sử dụng một gói tin RTS và một gói tin CTS. Chúng chiếm dụng đường truyền truyền tất cả các đoạn. Mỗi lần một đoạn dữ liệu được truyền, nơi gửi đợi một xác nhận ACK

từ nơi nhận. Nếu nó không nhận được ACK, nó sẽ mở rộng thời gian chiếm dụng đường truyền cho đủ một phân đoạn nữa, và truyền lại ngay phân đoạn dữ liệu hiện thời.

Như đã biết, tất cả các gói tin đều có trường *duration*, bây giờ nó là thời gian cần cho sự phát tất cả các phân đoạn dữ liệu còn lại và những gói ACK. Nếu một nút trong vùng lân cận nhận được một gói RTS hoặc CTS, thì nó sẽ chuyển sang trạng thái ngủ trong khoảng thời gian truyền tất cả các phân đoạn.

Mục đích của việc sử dụng ACK sau mỗi phân đoạn dữ liệu nhằm ngăn ngừa vấn đề nút ẩn (hidden station). Có thể một nút lân cận thức dậy hoặc một nút mới gia nhập vùng lân cận trong quá trình truyền. Nếu nút chỉ là lân cận của nút nhận nhưng không phải nút gửi, thì nó sẽ không nghe thấy các phân đoạn dữ liệu đang được phát từ nút gửi. Nếu nút nhận không gửi ACK thường xuyên, thì nút mới có thể gây nhiễu vì cảm nhận sóng mang trong việc thăm dò đường truyền sẽ thông báo đường truyền rỗi. Nếu nó khởi động tiến trình phát, thì quá trình truyền hiện thời sẽ bị hỏng ở tại nút nhận.

Mỗi phân đoạn dữ liệu và gói tin ACK cũng có trường *duration*. Bằng cách này, nếu một nút tỉnh dậy hoặc một nút mới gia nhập trong quá trình truyền, thì nó chuyển sang trạng thái ngủ bất kể nó là lân cận của nút gửi hay nút nhận. Ví dụ, giả sử một nút lân cận nhận được một RTS của nút gửi hoặc một CTS từ nút nhận, nó sẽ chuyển trạng thái ngủ trong toàn bộ thời gian được cung cấp trong thông điệp. Nếu bên gửi mở rộng thời gian truyền do mất phân đoạn dữ liệu hoặc do lỗi, vì ngủ nên các nút lân cận không ý thức được sự mở rộng này ngay lập tức. Tuy nhiên, các nút sẽ biết được điều này từ những phân đoạn mở rộng hoặc những gói tin ACK khi nó tỉnh dậy.

2.3.3. T-MAC (Time out - MAC):

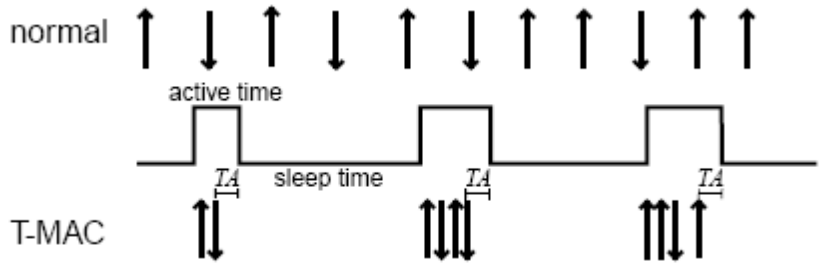
Giao thức điều khiển truy nhập T-MAC (Timeout-MAC) do hai tác giả Tijs van Dam và Koen Langendoen, khoa Công nghệ thông tin và các hệ thống, Trường đại học công nghệ Delft, Hà Lan, giới thiệu tại Hội nghị quốc tế về các

hệ thống mạng cảm biến nhúng lần thứ nhất tại Los Angeles, Mỹ, năm 2003 (Sensys'03). Giao thức này là sự cải tiến nhằm khắc phục nhược điểm của S-MAC.

S-MAC có hai tham số quan trọng: độ lớn của khung thời gian (frame time) và độ dài thời gian thức (active time). Độ lớn khung thời gian bị giới hạn bởi yêu cầu về độ trễ cho phép và độ lớn bộ đệm. Độ lớn thời gian thức phụ thuộc chủ yếu trên tốc độ phát sinh thông điệp: nó phải đủ lớn để nút cảm biến có thể phát đi tất cả các thông điệp của nó trong khoảng thời gian thức. Trong khi yêu cầu độ trễ và không gian bộ đệm nói chung là cố định thì tốc độ phát sinh thông điệp thường thay đổi. Để đảm bảo tất cả các thông điệp được phát như mong muốn, nút cảm biến phải được cài đặt một thời gian thức sao cho có thể xử lý ở mức thông lượng cao nhất. Nhưng khi thông lượng xuống thấp thì thời gian thức sẽ không được sử dụng tối ưu và do đó năng lượng sẽ bị lãng phí do vấn đề nghe nhàn rỗi (idle listening).

Ý tưởng mới của giao thức T-MAC là giảm bớt thời gian nghe khi rỗi bằng việc truyền tất cả các thông điệp trong những cụm (burst) có độ dài thay đổi tùy theo, và thực hiện ngủ giữa các cụm, xác định một cách mềm dẻo độ dài tối ưu thời gian thức theo sự thay đổi của lưu lượng đường truyền.

*** Những vấn đề cơ bản:**



Hình 2.6: Lược đồ cơ bản T-MAC với thời gian thức thay đổi

Hình 2.6 cho thấy lược đồ cơ bản của giao thức T-MAC. Mỗi nút định kỳ tỉnh dậy liên lạc các nút lân cận, sau đó ngủ tiếp cho đến khi khung tiếp theo. Trong lúc đó, những thông điệp mới được đưa vào hàng đợi. T-MAC cũng sử dụng kỹ thuật RTS, CTS, Data, ACK để tránh xung đột và truyền số liệu tin cậy.

Một nút sẽ được đặt ở chế độ nghe và sẵn sàng thực hiện truyền số liệu khi nó đang ở trong trạng thái thức. Trạng thái thức sẽ kết thúc khi không có một sự kiện kích hoạt (activation event) nào xuất hiện một khoảng thời gian TA. Một sự kiện kích hoạt là:

+ Sự kết thúc một khung thời gian theo định kỳ.

+ Sự tiếp nhận bất kỳ dữ liệu nào trên sóng vô tuyến.

+ Sự xuất hiện sự kiện cảm biến được phát hiện qua thành phần vô tuyến.

+ Sự kết thúc truyền dữ liệu của một nút có sở hữu gói dữ liệu hoặc sự biên nhận ACK;

+ Thông tin về sự kết thúc trao đổi dữ liệu của các nút lân cận qua nhận được các gói RTS, CTS.

Thông số TA xác định thời gian tối thiểu cho việc thức chờ nghe trên một khung thời gian.

Lược đồ timeout chuyển tất cả các giao dịch vào một cụm tại điểm bắt đầu của khung. Khi đó những thông điệp giữa các thời gian hoạt động phải được đưa vào bộ đệm. Độ lớn của bộ đệm xác định cận trên của độ lớn khung thời gian cực đại.

*** Phân nhóm và đồng bộ:**

Đồng bộ khung thời gian được thực hiện qua sự hình thành phân nhóm ảo như được mô tả trong giao thức S-MAC. Khi một nút cảm biến bắt đầu quá trình hoạt động của mình, nó bắt đầu bằng việc đợi và nghe. Nếu nó không nghe thấy gì trong một khoảng thời gian nhất định, thì nó tự chọn cho mình một lịch làm việc và truyền một gói tin đồng bộ SYNC chứa đựng thời gian khởi tạo của khung tiếp theo trong lịch làm việc. Nếu nút cảm biến trong thời gian khởi động nghe thấy một gói tin đồng bộ từ nút khác, thì nó sẽ theo lịch làm việc trong gói tin đồng bộ đó và quảng bá gói tin đồng bộ tương ứng của chính mình.

Các nút cảm biến thực hiện phát lại ngay gói tin đồng bộ của chúng. Chúng thực hiện nghe đầy đủ một khung một cách không thường xuyên, vì vậy chúng có thể phát hiện ra sự tồn tại của những thời gian biểu khác nhau. Điều

này này cho phép các nút mới hoặc các nút di động có thể được chấp nhận gia nhập nhóm đã tồn tại trước đó.

Nếu một nút đã có một thời gian biểu nhưng lại nghe được từ gói tin đồng bộ một thời gian biểu khác (từ nút khác), thì nó chấp nhận cả hai và thực hiện phát một gói tin đồng bộ chứa thời gian biểu của mình để cho các nút khác biết có sự tồn tại thời gian biểu đó. Việc chấp nhận cả hai thời gian biểu làm việc có nghĩa rằng nút sẽ có những sự kiện kích hoạt ở tại thời điểm bắt đầu của cả hai khung.

Muốn truyền dữ liệu, các nút cảm biến phải khởi động tại điểm bắt đầu khoảng thời gian thức quy định trong lịch biểu của chúng. Tại thời điểm đó, cả các nút lân cận có cùng thời gian biểu, và các lân cận mà đã chấp nhận thời gian biểu như sự bổ sung đều ở trạng thái thức. Nếu nó thực hiện ở điểm bắt đầu của một khung của nút lân cận, thì có thể nó phát trong khi lân cận của nó vẫn đang trong trạng thái ngủ. Với lược đồ này làm có thể thực hiện quảng bá mà chỉ cần phát một lần duy nhất.

Lược đồ đồng bộ được mô tả ở trên được gọi là quá trình phân nhóm ảo, thúc đẩy các nút hình thành nhóm với cùng thời gian biểu mà không bắt buộc thời gian biểu này áp dụng tới tất cả các nút trong mạng. Nó cho phép thực hiện quảng bá có hiệu quả, và tránh sự duy trì thông tin các nút lân cận.

*** Thực hiện gửi RTS và chọn TA trong T-MAC:**

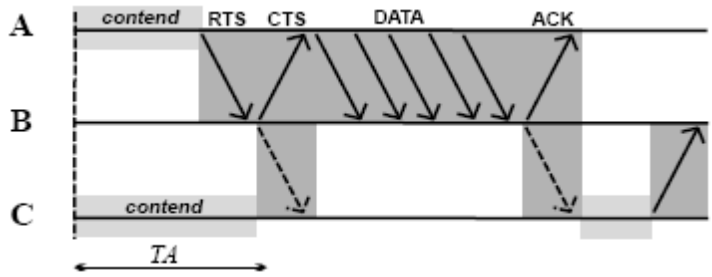
T-MAC cần bổ sung một số đặc tính so với S-MAC để thực hiện sự điều chỉnh tối ưu thời gian thức.

a. Khoảng cạnh tranh cố định (Fixed contention interval):

Trong những giao thức trên nền cạnh tranh, như IEEE 802.11, các nút đợi ngẫu nhiên một khoảng thời gian nhất định, gọi là khoảng thời gian cạnh tranh, sau khi phát hiện có xung đột. Chỉ khi đường truyền rỗi trong thời gian ấy chúng mới khởi động lại sự truyền. Thông thường, một lược đồ back-off được sử dụng: khoảng thời gian cạnh tranh tăng thêm khi lưu lượng đường truyền tăng. Lược

đồ back-off giảm bớt xác suất xảy ra xung đột khi tải tăng cao, trong khi tối thiểu độ trễ khi tải thấp.

Trong giao thức T-MAC, mỗi nút truyền các thông điệp trong hàng đợi của nó vào một cụm tại điểm bắt đầu của khung. Trong thời gian truyền cụm này, đường truyền là bão hòa: những thông điệp được truyền ở tốc độ cực đại. Mọi nút đều muốn giành quyền truy nhập đường truyền mỗi khi nó gửi một gói tin RTS. Khoảng cạnh tranh ngày càng tăng thì lại không có ích khi tải phần lớn đã cao và không thay đổi. Bởi vậy, sự truyền RTS trong T-MAC bắt đầu bởi việc đợi và nghe một khoảng thời gian ngẫu nhiên trong phạm vi một khoảng cạnh tranh cố định. Khoảng này được điều chỉnh phù hợp với tải cực đại. Khoảng thời gian cạnh tranh luôn luôn được sử dụng dù không có xung đột.



Hình 2.7: Lược đồ trao đổi dữ liệu cơ bản. Nút C nghe được CTS từ nút B và sẽ không làm phiền giao tiếp giữa A và B. TA phải đủ dài để C có thể nghe được phần đầu CTS

b. Thử lại phát lại RTS:

Khi một nút phát một gói tin RTS, nhưng không nhận được trở lại một CTS, có thể một trong ba trường hợp xảy ra:

+ Nút nhận không nghe được RTS vì xung đột, hoặc

+ Nút nhận bị ngăn cản trả lời vì nghe được RTS hoặc CTS, hoặc

+ Nút nhận đang ngủ.

Khi nút gửi không nhận câu trả lời trong khoảng TA, nó có thể chuyển sang trạng thái ngủ. Tuy nhiên, điều đó có thể không hợp lý trong những trường hợp 1 và 2: sẽ xảy ra hiện tượng nút muốn gửi chuyển sang trạng thái ngủ trong khi nút nhận vẫn thức. Khi trường hợp này xảy ra thậm chí ở ngay tại thông báo

đầu tiên của khung, thông lượng giảm đáng kể. Bởi vậy, một nút cần phải cố gắng gửi lại RTS nếu nó không nhận được câu trả lời. Nếu không có còn sự trả lời sau khi thử lại, nó cần phải từ bỏ ý định truyền và sang trạng thái ngủ.

c. Xác định khoảng TA:

Một nút không nên chuyển sang trạng thái ngủ trong khi các nút lân cận của nó vẫn còn trao đổi số liệu, một khi nút lân cận đó có thể là nút nhận của một thông báo kế tiếp. Khi bắt đầu nhận được gói tin RTS hoặc CTS của một nút lân cận cũng đủ thực hiện một tác vụ kích hoạt khởi tạo khoảng TA.

Không nằm trong vùng lân cận, nên một nút sẽ không nhận được thông điệp RTS từ một nút mà khởi tạo truyền thông với lân cận của nó. Khoảng TA phải đủ dài để nhận ít nhất bắt đầu của gói CTS (Hình 2.7). Sự quan sát này cho chúng ta một cận dưới của độ dài khoảng TA:

$$TA > C + R + T$$

Ở đây C là chiều dài khoảng cạnh tranh, R là độ dài một gói RTS, và T là thời gian turn-around (khoảng thời gian ngắn giữa kết thúc của gói RTS và sự bắt đầu của gói CTS). Chọn thời gian TA lớn sẽ làm tăng sự tiêu phí năng lượng.

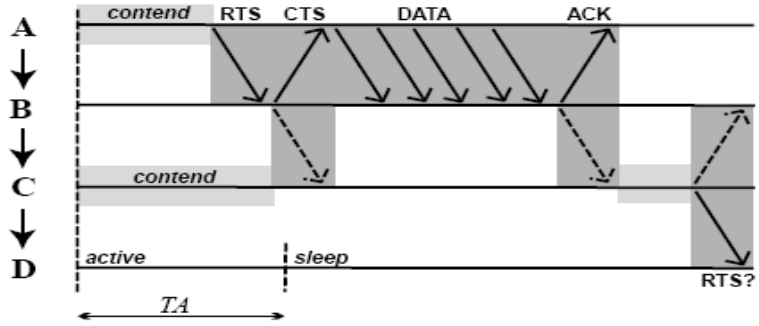
*** Tránh nghe lỏm:**

Giao thức S-MAC đưa ra ý tưởng nút sẽ sang trạng thái ngủ sau khi nghe được một gói tin RTS hoặc CTS dành cho cho nút khác. Khi đó nút bị ngăn cản việc gửi dữ liệu trong thời gian đó, nó không thể tham gia bất kỳ truyền thông nào và tốt nhất là tắt bộ phận thu phát vô tuyến của nó để tiết kiệm năng lượng.

Tránh nghe lỏm là một tùy chọn trong giao thức T-MAC để giảm năng lượng tiêu thụ. Tuy nhiên, chúng sẽ làm xung đột do thông tin điều khiển (overhead collision) cao hơn: một nút có thể không nhận được gói tin RTS và CTS trong khi ngủ và làm phiền giao tiếp nào đó khi nó tỉnh dậy trở lại. Do vậy, lưu lượng cực đại giảm bớt. Mặc dầu việc tránh nghe thừa sẽ tiết kiệm điện năng nhưng nó không được sử dụng khi muốn đạt băng thông cực đại.

*** Truyền thông bất đối xứng:**

Do lưu lượng trên mạng cảm biến phần lớn là đẳng hướng, như dạng truyền thông từ nhiều nút tới nút gốc (Nodes-to-Sink), nên T-MAC xuất hiện hiện tượng làm giảm thông lượng cực đại của mạng. Hiện tượng này được mô tả như sau (Hình 2.8): Các nút từ A đến D hình thành một tế bào với các lân cận của nó. Các thông điệp di chuyển từ trên xuống dưới, như vậy nút A chỉ phát tới B, B chỉ phát tới C, và C chỉ phát tới D.



Hình 2.8: Hiện tượng ngủ sớm. D đi ngủ trước khi C gửi một RTS cho nó

Khi nút C muốn phát dữ liệu tới nút D, nó phải tiến hành cạnh tranh, thăm dò đường truyền để giành quyền phát. Việc thăm dò có thể không tiến hành được vì trước đó nó nhận một thông điệp RTS từ nút B, hoặc nghe được thông điệp CTS từ nút B trả lời tới nút A.

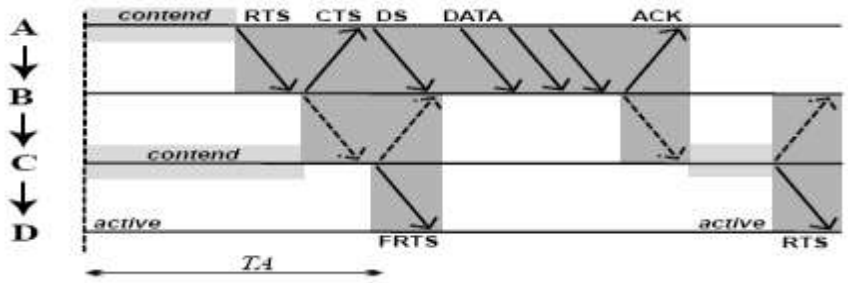
Khi C không tiến hành được việc thăm dò đường truyền do nhận được thông điệp RTS từ nút B, nó sẽ trả lời B một thông điệp CTS, D sẽ nghe được thông điệp này và đặt lịch chuyển sang trạng thái thức khi truyền thông giữa C và B kết thúc. Tuy nhiên, nếu C không tiến hành được là do nghe được thông điệp CTS từ B trả lời A (Hình 2.11), nó phải giữ im lặng. Ở trường hợp này, do D không biết truyền thông giữa A và B, không nhận được thông điệp muốn truyền dữ liệu từ C, nó sẽ chuyển sang trạng thái ngủ khi thời gian thức theo lịch kết. Chỉ ở tại điểm bắt đầu của khung tiếp theo, nút C mới có cơ hội để thực hiện thăm dò và tiến hành trao đổi dữ liệu với nút D.

Những vấn đề quan sát được ở trên được gọi là hiện tượng ngủ sớm (early sleeping problem), tức là một nút chuyển sang trạng thái ngủ khi một nút lân cận vẫn thức và muốn trao đổi dữ liệu với nó. Trong dạng truyền thông từ nút đến

nút gốc, ngủ sớm làm giảm thông lượng có thể của T-MAC tới ít hơn một nửa thông lượng cực đại của những giao thức truyền thống, hoặc so với S-MAC. Có hai giải pháp để khắc phục hiện tượng trên:

- *Gửi sớm RTS (Future request to send):*

Ý tưởng của giải pháp gửi sớm RTS là sẽ để cho nút nhận tiềm năng (nút D) biết được có một nút muốn gửi dữ liệu cho nó, nhưng đang trong tình trạng phải giữ im lặng để không làm ảnh hưởng đến giao tiếp khác. Khi một nút nghe được một thông điệp CTS dành cho cho nút khác, nó ngay lập tức gửi một gói FRTS (nút C trong Hình 2.9). Gói FRTS chứa thông tin về độ dài của khối dữ liệu truyền thông lấy được trong thông điệp CTS.



Hình 2.9. Thực hiện gửi sớm RTS. Gói tin FRTS giữ D thức

Một nút nhận được gói tin FRTS thì nó biết rằng trong khoảng thời gian t tiếp theo nó sẽ nhận được một thông điệp RTS, do vậy phải lập lịch thức trước thời gian ấy. Thông tin thời gian t được lấy trong thông điệp FRTS.

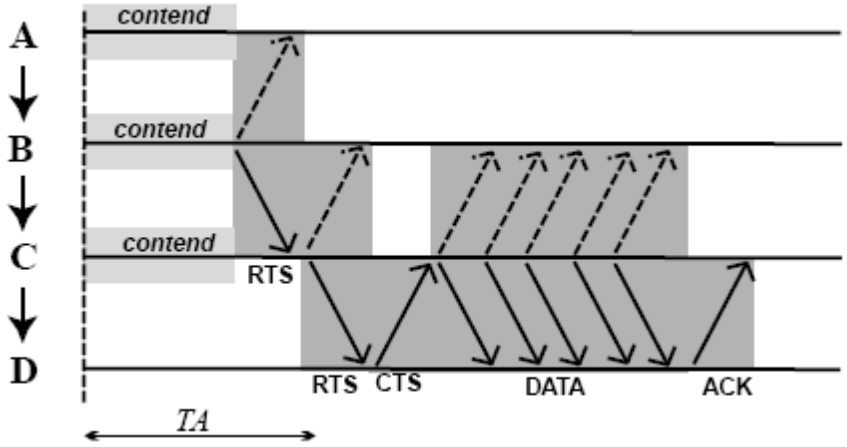
Để thông điệp FRTS (do C phát) không gây nhiễu dữ liệu trao đổi (giữa A và B) theo sau thông điệp CTS, dữ liệu phải được hoãn lại khoảng thời gian truyền FRTS. Để không mất kênh truyền, nút khởi tạo RTS ban đầu (Nút A) truyền một gói tin DS (Data-Send). Sau gói DS, nó ngay lập tức gửi dữ liệu bình thường. FRTS có cùng kích thước với DS, nó sẽ chỉ xung đột với gói DS mà không phải với gói dữ liệu. Gói DS bị mất, nhưng không có vấn đề gì vì nó không chứa đựng thông tin.

Với giải pháp FRTS, độ dài của khoảng thời gian TA phải được tăng thêm một khoảng bằng độ dài thông điệp CTS. Việc thực hiện giải pháp gửi sớm RTS sẽ làm tăng thông lượng cực đại trong truyền thông đẳng hướng. Tuy nhiên, vì

có DS và FRTS, mức tiêu thụ năng lượng cũng tăng thêm. Cũng có thể sử dụng kỹ thuật FRTS trong các dạng truyền thông khác nhưng chỉ khi muốn tăng thông lượng một cách chính đáng. Vì khi tải ở mức thấp thì tốc độ trao đổi dữ liệu cũng thấp do phải gia tăng xử lý thông tin điều khiển.

- Thực hiện ưu tiên gửi khi bộ đệm đầy (taking priority on full buffers):

Khi nào bộ đệm truyền/định tuyến của một nút gần đầy, thì việc gửi sẽ hợp lý hơn là tiếp tục nhận. Khi một nút nhận được RTS dành cho nó, ngay lập tức nó gửi gói RTS của chính mình cho nút khác, thay vì việc trả lời với một CTS như bình thường.



Hình 2.10: Thực hiện ưu tiên gửi khi bộ đệm đầy

Giải pháp này có hai hiệu quả, trước hết khi nút C khi trả lời B bằng thông điệp RTS khi bộ đệm của nó đầy, một mặt nó trả lời B rằng nó không muốn nhận, mặt khác đồng thời nó cũng thông báo cho D là nó muốn gửi dữ liệu. Như vậy xác suất mà vấn để ngủ sớm xảy ra sẽ thấp hơn. Hai là, thực hiện ưu tiên gửi khi bộ đệm đầy hình thành một giới hạn điều khiển luồng trong mạng có lợi cho những dạng truyền thông từ nút tới nút gốc. Trong Hình 2.10, nút B bị ngăn gửi cho đến khi nút C có đủ không gian bộ đệm.

Tuy nhiên, phải cẩn thận khi sử dụng giải pháp này, nó nguy hiểm trong trường hợp tải cao khi dạng truyền thông không phải là đẳng hướng. Khi những tất cả nút trong một mẫu truyền thông đa hướng bắt đầu giành quyền ưu tiên, cơ hội cho xung đột tăng nhanh chóng. T-MAC sử dụng một giới hạn: một nút có

thể chỉ sử dụng giải pháp này khi nó đã mất cạnh tranh ít nhất hai lần. Giới hạn này bảo vệ sự thực hiện trong một mẫu truyền thông đa hướng, trong khi vẫn tăng thông lượng cực đại trong mẫu đơn hướng.

2.4. Kết luận:

Mục đích chính của việc điều khiển thâm nhập môi trường (Medium Access Control - MAC) trong WSN là quản lý hiệu quả hoạt động của sóng vô tuyến trong mạng nhằm tiết kiệm năng lượng cho các nút mạng. Trong chương này chúng ta có thể nắm được các thông số cần quan tâm khi thiết kế giao thức MAC và các nguyên nhân gây ra hao phí năng lượng trong WSN. Từ đó tìm hiểu một số giao thức MAC đã được đề xuất cho WSN hoạt động như thế nào để có thể tiết kiệm năng lượng cho các nút mạng một cách hiệu quả nhất.

CHƯƠNG 3: THỰC NGHIỆM MÔ PHỎNG GIẢI THUẬT ĐIỀU KHIỂN THÂM NHẬP MÔI TRƯỜNG TRONG WSN

Chương trình thử nghiệm là chương trình khảo sát đo nhiệt độ môi trường trong mạng WSN được tiến hành trên một số nút mạng cảm biến CC1010 (sử dụng vi điều khiển CC1010 để xây dựng nút mạng trong mạng cảm biến) với phương pháp lập lịch tập trung. Lý do lựa chọn phương pháp này của em là do đây là một trong những phương pháp có thể tiết kiệm được năng lượng cho nút mạng không dây và dễ triển khai trên thực tế.

3.1. Chế độ lập lịch tập trung:

Lập lịch tập trung là một trong nhiều cách tổ chức hoạt động của các nút mạng trong mạng WSN. Với lập lịch tập trung, hoạt động của mạng sẽ do hoàn toàn nút cơ sở điều khiển. Tất cả các nút cảm nhận khác trong mạng sẽ hoạt động theo yêu cầu của nút mạng cơ sở, các yêu cầu điều khiển này đã được nút mạng cơ sở sắp xếp cố định và trở thành một lịch trình hoạt động cho mạng đó.

Cấu trúc của gói dữ liệu đã được sử dụng trong chương trình lập lịch tập trung với tất cả các nút mạng có dạng như sau:

7 byte	2 byte	16byte	2 byte	2 byte	2 byte	1 byte
Preamble	NútID	NútName	Data	Target	Type	CRC

Trong đó:

Preamble : các byte dẫn đường dùng để đồng bộ ngưỡng cho bộ thu RF

NútID: Địa chỉ của nút truyền.

NútName: Tên của nút truyền.

Data: Thông tin dữ liệu truyền về.

Target: Địa chỉ nút nhận.

Type: Loại gói dữ liệu (điều khiển/dữ liệu).

CRC : Byte chứa thông tin kiểm tra lỗi dư thừa vòng.

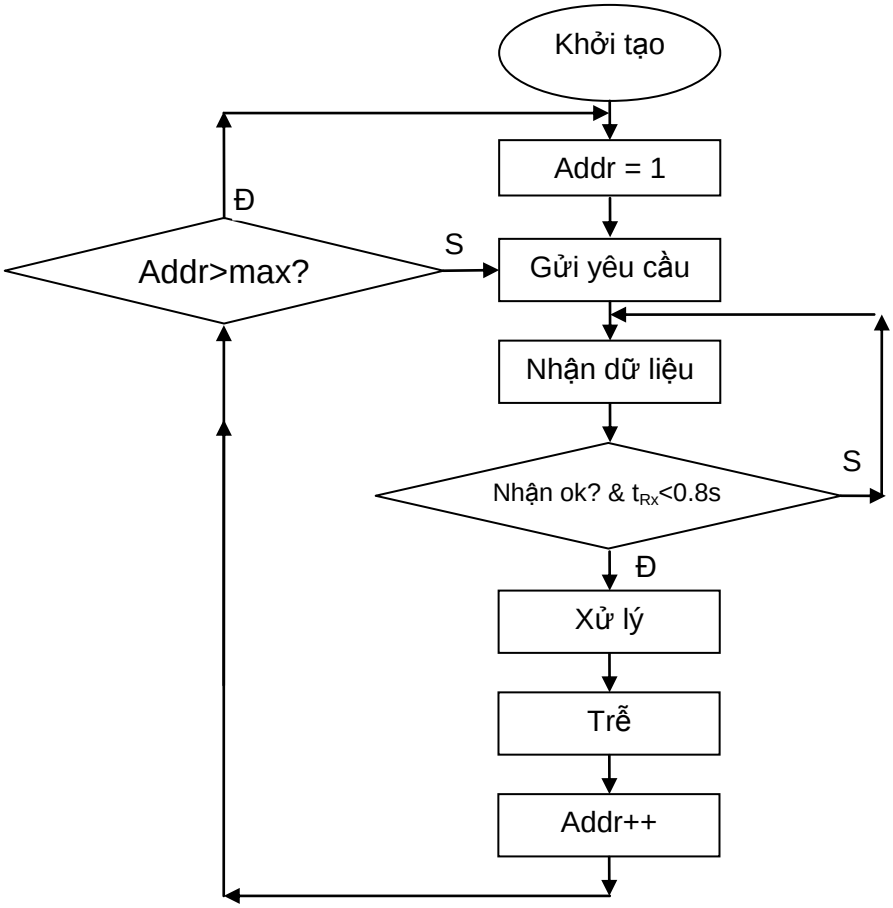
*** Hoạt động:**

- Nút cơ sở: Sẽ lần lượt yêu cầu từng nút mạng gửi dữ liệu truyền về. Mỗi nút mạng khi nhận được yêu cầu sẽ phải gửi dữ liệu phản hồi về cho nút cơ sở.

Ban đầu, nút cơ sở gửi yêu cầu tới một nút cảm nhận n (NútID=n). Sau khi gửi gói tin yêu cầu nút n ra ngoài môi trường thì các nút mạng khác cũng đều nhận được yêu cầu đó chứ không riêng nút n nhận được. Tiếp theo nút mạng cơ sở chuyển sang chế độ nhận dữ liệu từ nút mạng cảm nhận truyền về. Lúc này, nút mạng cơ sở sẽ nhận dữ liệu, nếu nhận không thành công thì nhận lại lần nữa. Tuy nhiên, quá trình nhận lại này sẽ được giới hạn về mặt thời gian. Tức là với mỗi một nút cảm nhận, nút cơ sở chỉ dành cho một lượng thời gian nhất định, nếu sau khoảng thời gian đó mà vẫn không nhận được gói dữ liệu truyền về thì tức là nút mạng được yêu cầu đó đã không nhận được yêu cầu hoặc là gói dữ liệu truyền vễ đã gặp phải lỗi truyền nào đó ngoài môi trường dẫn tới nút cơ sở không thể nhận được. Lúc này nút cơ sở sẽ phải bỏ qua nút đó để chuyển tiếp sang yêu cầu nút khác. Nếu trong thời gian đó mà nút cơ sở nhận được thành công gói dữ liệu của nút cảm nhận truyền về thì sẽ tiến hành xử lý gói tin và hiển thị hoặc lưu trữ dữ liệu nhận được đó. Sau khi xử lý xong gói tin, trước khi chuyển sang yêu cầu nút tiếp theo (n+1) thì nút cơ sở sẽ trễ một khoảng tdelay. Lý do phải có khoảng trễ này sẽ được giải thích tại phần hoạt động của nút cảm nhận. Như vậy là nút cơ sở đã hoàn thành việc yêu cầu và nhận gói tin từ một nút cảm nhận trong mạng. Sau đó nút cơ sở sẽ tăng thêm 1 vào địa chỉ của nút sẽ yêu cầu, tức là sẽ tiến hành nút cảm nhận tiếp theo. Nhưng nếu khi tăng địa chỉ nút yêu cầu mà vượt quá giới hạn đã đặt là số nút tối đa có trong mạng thì nút cơ sở lại quay lại yêu cầu nút đầu tiên (nútID=1). Quá trình được lặp lại liên tục, nút cơ sở yêu cầu và nhận dữ liệu của lần lượt từng nút trong mạng, sau khi hết một lượt lại tiến hành quay lại với nút đầu tiên.

```
while (TRUE) {
        for(i=1;i<max;i++)              // max là số nút mạng
        {
                tbcTransmit(i,0xFF);    // gửi yêu cầu cho nút I, loại gói tin điều
        khiển
                t = (int) sppGetTime(); // lấy thời gian trước khi nhận
                do{                     //Nếu nhận không thành công sẽ nhận lại
                                        cho //toi khi hết 800ms thì thôi.
                        tbcReceive();
         }while((RXI.status != SPP_RX_FINISHED)&&( ((int) sppGetTime()-
        t)<80));
                 tbcPrintTable();       // xử lý gói tin
                delay(10000);//Thời gian Master đợi các Slave tính toán, xử lý dữ
        liệu
        }
}
```

*** Giải thuật của nút cơ sở:**



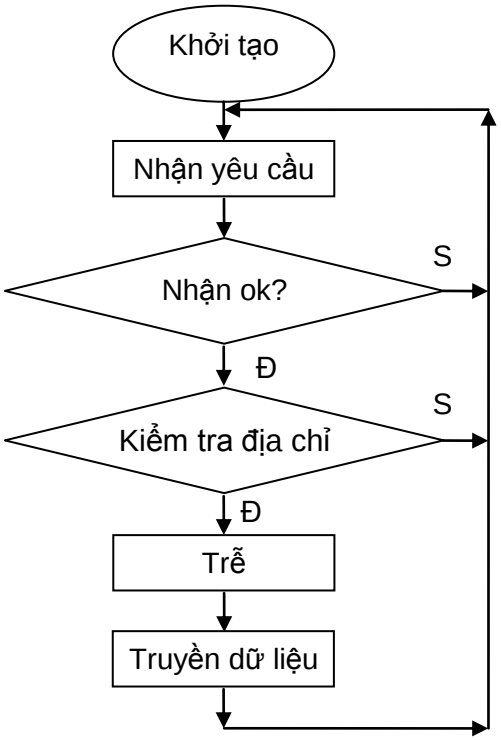
Hình 3.1: Giải thuật nút cơ sở

- Nút cảm biến: Luôn ở trạng thái sẵn sàng nhận yêu cầu của nút cơ sở. Ngay sau khi nhận được yêu cầu sẽ tiến hành gửi dữ liệu trở về.

Ban đầu, nút cảm nhận sẽ luôn ở trong trạng thái nhận dữ liệu, nếu nhận không thành công thì nhận lại cho tới khi nhận được gói dữ liệu thành công. Sau khi nhận được gói, nút cảm nhận tiến hành tách từng trường dữ liệu của gói đã nhận được và kiểm tra xem đó có phải là yêu cầu gửi dữ liệu của nút cơ sở yêu cầu chính nó hay không. Nếu đúng thì nút cảm nhận sẽ trễ một khoảng thời gian ngắn rồi mới tiến hành gửi dữ liệu trở về cho nút cơ sở. Rồi sau đó lại quay trở lại quá trình nhận yêu cầu để chờ tín hiệu yêu cầu lần tiếp theo. Nếu gói tin nhận được không phải là yêu cầu chính nó gửi dữ liệu về thì lúc này có thể có 2 khả năng: hoặc là đó gói tin yêu cầu của nút cơ sở gửi tới yêu cầu một nút khác, hoặc đó là một gói tin dữ liệu của một nút cơ sở khác đang truyền về. Khi đó, nó

sẽ quay trở lại việc nhận dữ liệu từ môi trường. Nhưng sau mỗi quá trình nhận thì nút cảm nhận lại phải tiến hành tách các trường dữ liệu trong gói tin nhận được và kiểm tra các trường đó. Các công việc này sẽ tốn một khoảng thời gian và khiến cho nút cảm nhận không được yêu cầu sẽ quay trở lại quá trình nhận yêu cầu muộn hơn nút cảm nhận vừa nhận được yêu cầu và đã gửi dữ liệu đi. Và cũng chính vì quá trình xử lý mất một khoảng thời gian mà tại nút cơ sở sau khi xử lý xong dữ liệu lại phải trễ đi một khoảng để khi gửi tin yêu cầu thì các nút cảm nhận khác có thể nhận được.

*** Giải thuật nút cảm biến:**



Hình 3.2 Giải thuật nút cảm biến

```
while (TRUE) {
        do{
                tbcReceive();
        }while(RXI.status != SPP_RX_FINISHED);
        if((nútTarget[n]==1)&& (nútType[n]==0xFF))
```

```
        {
                delay(20000);
                tbcTransmit(0xFFFF,0xFF00);
        }
}
```

3.2. Thiết lập thực nghiệm:

*** Các thiết bị thực nghiệm:**

- Các nút của mạng được xây dựng từ vi điều khiển CC1010.

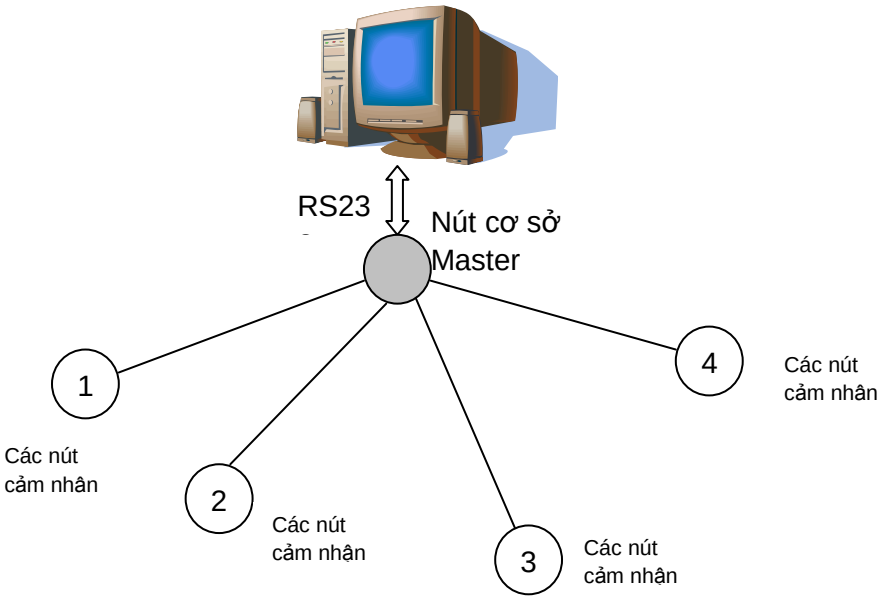
- Hệ thống WSN được kết nối với máy tính thông qua cáp nối RS232 để nạp phần mềm cho các nút mạng. Ngôn ngữ lập trình sử dụng là ngôn ngữ C cùng với chương trình dịch là Keil µVision2.0.

- Nguồn pin, đồng hồ đo.

Hình 3.1: Nút cảm biến sử dụng khối EM-CC1010

Hình 3.2: Nút mạng cảm biến có gắn mà hình hiển thị kết quả đo

*** Sơ đồ thực nghiệm và thuật toán:**



Hình 3.3: Sơ đồ thực nghiệm mạng WSN

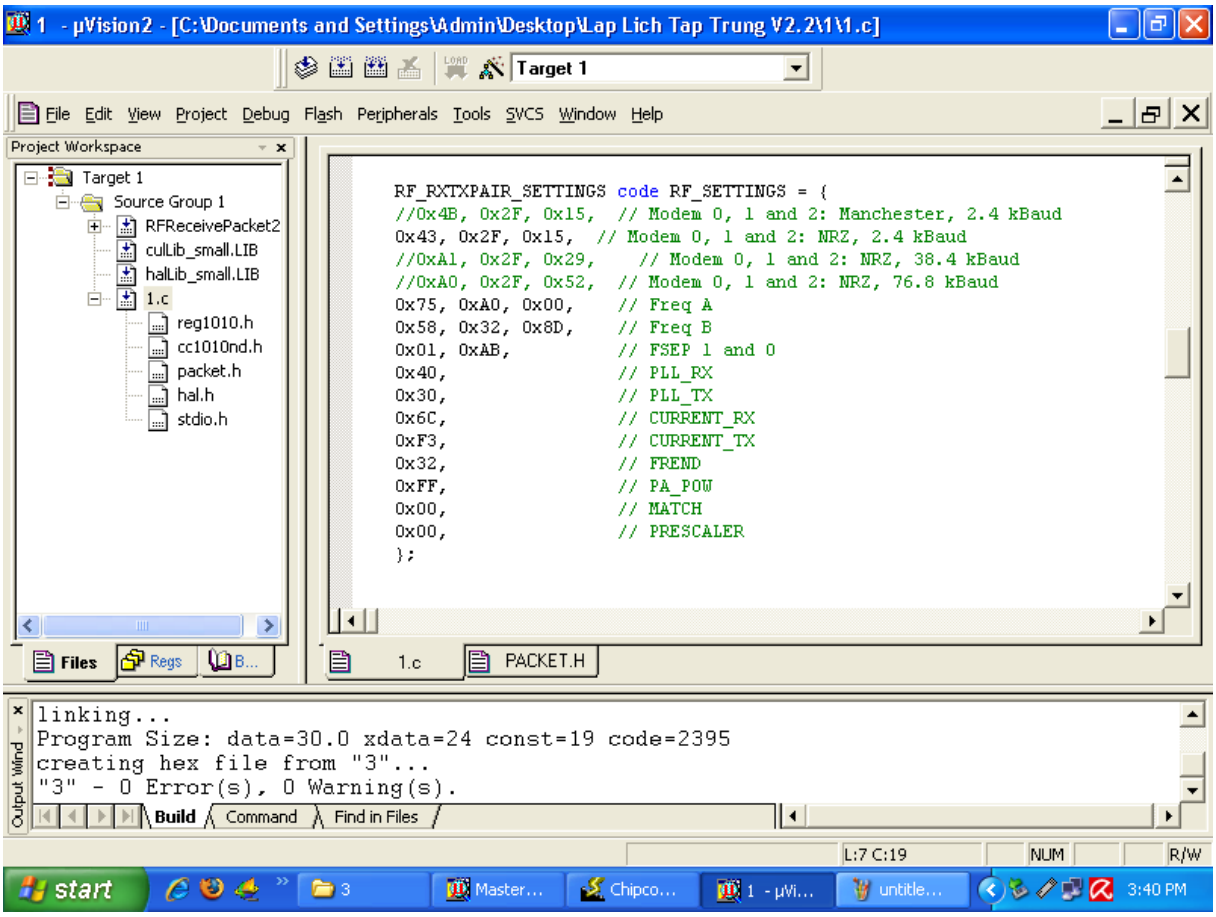
Nút CC1010 nối trực tiếp với máy tính qua cổng RS232 (gọi là nút cơ sở)

nhận kết quả từ nút cảm nhận (có thể di động xa nút cơ sở). Nút cơ sở sẽ phát tín hiệu yêu cầu thu thập dữ liệu từ các nút cảm nhận theo nguyên tắc hỏi vòng, các nút cảm biến sau khi nhận được yêu cầu từ nút cơ sở sẽ gửi trả lời.

*** Nạp chương trình cho các nút mạng:**

- Nối bản mạch với máy tính để nạp chương trình nhúng cho các nút mạng thông qua bản mạch này.

- Dùng trình biên dịch Keil µVision 2.0 để dịch chương trình thử nghiệm trên máy tính từ ngôn ngữ C sang mã máy ta được file .hex để nạp cho các nút.



Hình 3.4: Dịch chương trình nhúng bằng Keil µVision 2.0

- Bật nguồn pin của bản mạch vừa gắn nút mạng, mở chương trình Chipcon CC1010 Flash Programmer để nạp tệp .hex vừa dịch cho nút mạng.

Hình 3.5: Nạp chương trình nhúng cho các nút mạng

3.3. Tiến hành thực nghiệm:

Tiến hành đo dòng điện tiêu thụ 4 trạng thái khác nhau của nút mạng là: truyền, nhận, ngủ và khi không truyền – nhận.

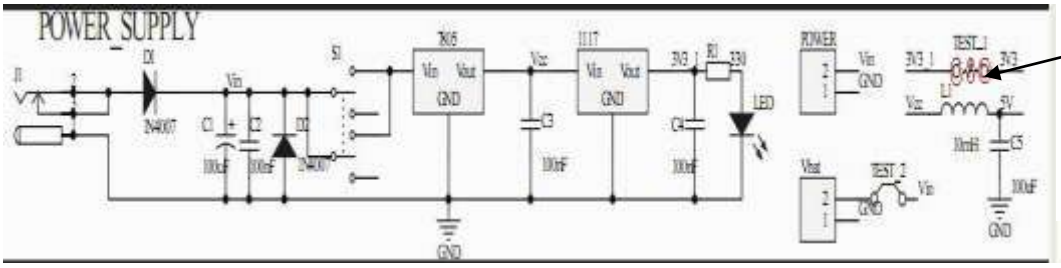
Cách đo dòng điện:

CC1010 sử dụng nguồn nuôi 3.3V, điện áp này được tạo ra từ jump test1 (được đánh dấu trên sơ đồ trong Hình 3.6).

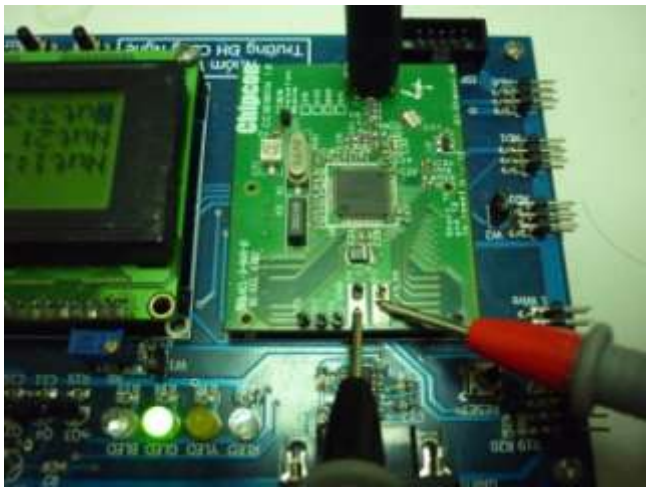
Trong sơ đồ CC1010MB thì chỉ có chip CC1010 sử dụng nguồn nuôi 3.3V nên dòng qua jump test 1 cũng chính là dòng tiêu thụ của chip CC1010.

Để đo dòng này ta đặt một đồng hồ đo dòng nối tiếp tại test1 như sơ đồ trong Hình 3.6.

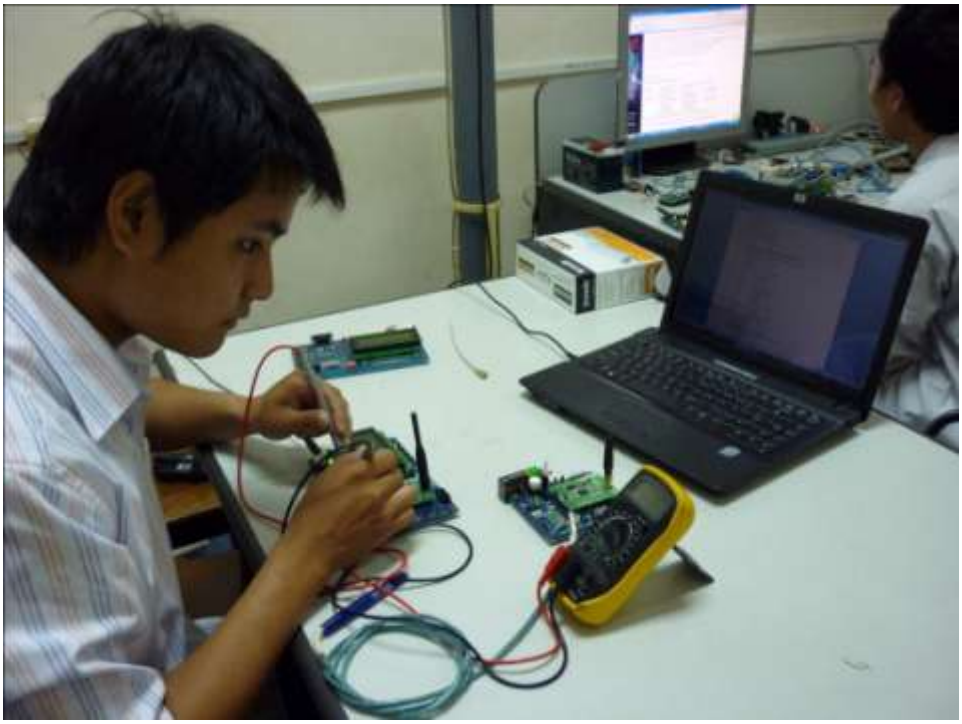
Đồng thời nạp chương trình cho CC1010 chạy trong từng chế độ riêng biệt, ta sẽ đo được dòng tiêu thụ của CC1010 trong chế độ tương ứng.



Hình 3.6: Sơ đồ chip CC1010, vị trí đánh dấu để đo dòng điện tiêu thụ



Hình 3.7: Vị trí đo dòng điện tiêu thụ trên CC1010



Hình 3.8: Đo dòng điện trên chip CC1010

a. Đo dòng điện tiêu thụ của nút mạng lúc mạng nhận dữ liệu:

Để có thể đo dòng điện tiêu thụ của nút mạng ở chế độ nhận dữ liệu ta sẽ phải làm nút mạng chỉ hoạt động ở chế độ này bằng cách nạp chương trình cho nút luôn ở chế độ nhận dữ liệu sau đó tiến hành đo cường độ dòng điện.

* Các bước tiến hành:

- Thiết lập chế độ hoạt động cho module RF.

- Cho phép module RF bắt đầu nhận tín hiệu.

- Chương trình lặp vô hạn.

* Kết quả:

Cường độ dòng điện khi nút mạng nhận dữ liệu là: 23mA

b. Đo dòng điện tiêu thụ của nút mạng lúc mạng truyền dữ liệu:

Cho nút mạng hoạt động ở chế độ nhận dữ liệu, nạp chương trình cho nó luôn ở chế độ phát tín hiệu, sau đó tiến hành đo cường độ dòng điện.

* Các bước tiến hành:

- Thiết lập chế độ hoạt động cho module RF.

- Cho phép module RF bắt đầu phát tín hiệu.

- Nhảy tại chỗ.

* Kết quả:

Cường độ dòng điện khi nút mạng truyền dữ liệu là: 35mA

c. Đo dòng điện tiêu thụ của nút mạng lúc mạng ngủ:

Cho nút mạng về chế độ ngủ, nạp chương trình cho nó luôn ở chế độ ngủ rồi tiến hành đo dòng điện.

Nút cảm biến sẽ ở chế độ ngủ ngay khi bắt đầu chạy.

* Kết quả:

Cường độ dòng điện khi nút mạng ngủ là: 0.18mA

d. Đo dòng điện tiêu thụ của nút mạng lúc mạng không truyền - nhận dữ liệu:

Tương tự như trên, ta cũng thiết lập nút mạng về chế độ không truyền - nhận dữ liệu bằng cách nạp chương trình cho nó ở chế độ không truyền – nhận dữ liệu rồi tiến hành đo.

* Các bước tiến hành:

- Thiết lập chế độ cho module ADC và kích hoạt ADC, không kích hoạt module RF, chỉ cho CPU ở chế độ chạy lệnh.

- Chương trình nhảy tại chỗ.

* Kết quả:

Cường độ dòng điện khi nút mạng không truyền – nhận là: 17mA

e. Đánh giá kết quả :

Từ các kết quả thu được ở trên ta nhận thấy chương trình thực hiện được việc tiết kiệm năng lượng rất rõ ràng. Dòng điện tiêu thụ ở chế độ nghỉ còn nhỏ 1% của dòng tiêu thụ khi nút mạng ở chế độ hoạt động. Vì vậy, nếu thời gian nút mạng ở trong chế độ nghỉ kéo dài sẽ tiết kiệm năng lượng rất nhiều. Tùy theo từng ứng dụng thực tế yêu cầu mà ta có thể tăng hoặc giảm thời gian nghỉ của nút mạng.

Phương pháp này rất hiệu quả đối với những mạng chỉ cần cung cấp thông tin một cách định kỳ như theo dõi thời tiết, mực nước, nhiệt độ tại những thời điểm trong ngày…vì thời gian nghỉ của mạng lớn và các nút chỉ hoạt động trong những khoảng thời gian được định sẵn. Căn cứ vào nhu cầu thực tế ta có thể can thiệp vào thời gian nút mạng nghỉ để có được hiệu quả tiết kiệm năng lượng nhất.

KẾT LUẬN

Nghiên cứu về mạng cảm biến không dây là một vấn đề mới còn lạ lẫm với nhiều người làm việc trong lĩnh vực công nghệ thông tin. Qua đồ án này, em đã trình bày một cách tổng quan về mạng cảm biến không dây cùng với những ưu điểm, tính năng ưu việt và ứng dụng đa dạng mà không phải mạng nào cũng có. Trong tương lai không xa thì mạng cảm biến sẽ nhanh chóng được phát triển, ứng dụng rộng rãi nhằm phục vụ tốt hơn đời sống, nhu cầu của con người. Với đồ án này, em mong rằng có thể góp một phần nhỏ bé của mình vào việc nghiên cứu về lĩnh vực khá mới mẻ này ở nước ta.

Trong phạm vi nhiệm vụ của đồ án tốt nghiệp này, em đã nghiên cứu những vấn đề cơ bản của việc thiết kế các giao thức điều khiển môi trường và các nguyên nhân gây hao phí năng lượng của mạng cảm biến không dây, tìm hiểu một số các giao thức thâm nhập môi trường và tiến hành thực nghiệm phương pháp lập lịch tập trung. Do đây là một lĩnh vực còn khá mới mẻ và kiến thức của em còn hạn chế nên đồ án tốt nghiệp của em không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự góp ý, phê bình của các thầy cô trong khoa để đồ án của em được hoàn thiện hơn.

Một lần nữa, em xin chân thành cảm ơn PGS.TS. Vương Đạo Vy – Khoa Điện tử viễn thông – Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội đã tận tình giúp đỡ em trong thời gian vừa qua.

TÀI LIỆU THAM KHẢO

Tài liệu Tiếng Việt

[1] Đỗ Thị Tuyết (2008), *"Nghiên cứu và mô phỏng giao thức định tuyến Pagasis trong mạng cảm biến"*, Đại học Bách khoa Hà Nội.

[2] Đỗ Duy Tân (2009), *"Wireless Sensor Networks, Kỹ thuật, Giao thức và Ứng dụng"*, Đại học Quốc gia TP.HCM.

[3] Phạm Mạnh Toàn (2009), *"Nghiên cứu về hiệu quả năng lượng của một số giao thức điều khiển thâm nhập môi trường trong mạng cảm biến không dây"*, Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội.

Tài liệu Tiếng Anh

[4] Thomas Hanselmann (2008), *"Sensor Networks"*.

[5] Bhaskar Krishnamachari (2005), *"Networking Wireless Sensors"*, Cambridge University

[6] Wei Ye, John Heidemann, Deborah Estrin (2002), "*An Energy-Efficient MAC Protocol for Wireless Sensor Networks*", University of California.

[7] Vanitha SivaSubramaniam (2003), *"Energy Efficient MAC Protocols For Ad Hoc Networks"*

PHỤ LỤC

Chương trình khảo sát đo nhiệt độ môi trường trong mạng WSN

Phần mềm viết cho Master bao gồm các file: CC1010MB.h, Master.c

Phần mềm viết cho Slave bao gồm các file: Slave.c

1. CC1010MB.h: File thư viện định nghĩa các chân I/O

```
#ifndef CC1010EB_H
#define CC1010EB_H
#include <Chipcon/reg1010.h>    // Include register definitions

//********** Constants ************
#define CC1010EB_CLKFREQ 14746
#define STRING_LENGTH    16

//********** LED macros ************
#define RLED P1_4              // last recieved false
#define YLED P1_5              // waitting for recieve
#define GLED P1_6   // last recieved ok
#define BLED P1_7   // transmitting
#define LED_ON       0
#define LED_OFF 1

// LED pin output enable macros
#define RLED_OE(x) {P1DIR=(x) ? P1DIR&~0x10 : P1DIR|0x10;}
#define YLED_OE(x) {P1DIR=(x) ? P1DIR&~0x20 : P1DIR|0x20;}
#define GLED_OE(x) {P1DIR=(x) ? P1DIR&~0x40 : P1DIR|0x40;}
#define BLED_OE(x) {P1DIR=(x) ? P1DIR&~0x80 : P1DIR|0x80;}

//********** Switch macros ************
#define SW1_PRESSED (!P1_7)
#define SW2_PRESSED (!P3_2)
#define SW3_PRESSED (!P3_3)
```

```
#define SW4_PRESSED (!P2_4)

//********** Button macros ***********
#define MENU_PRESSED (!P1_3)
#define SELECT_PRESSED (!P2_3)
#define UP_PRESSED (!P3_3)
#define DOWN_PRESSED (!P2_4)

//********** Misc macros *************
#define DCLK          P0_2
#define DIO           P0_1
#define DCLKIO_OE(b)    {P0DIR=(P0DIR&~0x06)|((b)?0x00:0x06);}

// PUT IN EXAMPLE RF_RXTXPAIR STRUCTURES FOR 434/868/915 !!!

#endif //CC1010EB_H
```

2. Master.c

```
#include <chipcon/reg1010.h>
#include "CC1010MB.h"
#include "PACKET.H"
#include <chipcon/hal.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

//LCD define
#define LCDPORT P2
sbit _RS=LCDPORT^0;
sbit _RW=LCDPORT^1;
sbit _E =LCDPORT^2;
#include "lcd.h"

// Node ID constants
```

```
#define Node_ID     0xFF
#define Node1       0x01
#define Node2       0x02
#define Node3       0x03
#define Node4       0x04
#define Max_Node    3

// Data type
#define Temp_LM35               0x01
#define Temp_DS18B20            0x02
#define Presure                 0x03

// warning level
#define Nguong1         25
#define Nguong2         30
#define Nguong3         35

// Protocol const
#define PREAMBLE_BYTE_COUNT     10
#define PREAMBLE_BITS_SENSE     16

#define CRC16_POLY              0x1021
#define CRC16_INIT              0xFFFF
#define CRC_OK                  0

PACKET xdata txDataBuffer;
PACKET xdata rxDataBuffer;

byte halRFReceivePacket2(byte timeOut, byte* packetData, byte maxLength);

// MAIN PROGRAM
void main(void) {
        unsigned int n;
        byte result;
```

```
        unsigned int node;
        char so[20];
        float fTemp;

// X-tal frequency: 14.745600 MHz
// RF frequency A: 868.277200 MHz Rx
// RF frequency B: 868.277200 MHz Tx
// RX Mode: Low side LO
// Frequency separation: 64 kHz
// Data rate: 2.4 kBaud
// Data Format: Manchester
// RF output power: 4 dBm
// IF/RSSI: RSSI Enabled

RF_RXTXPAIR_SETTINGS code RF_SETTINGS = {
   //0x4B, 0x2F, 0x15,    // Modem 0, 1 and 2: Manchester, 2.4 kBaud
   0x43, 0x2F, 0x15,    // Modem 0, 1 and 2: NRZ, 2.4 kBaud
   //0xA1, 0x2F, 0x29,    // Modem 0, 1 and 2: NRZ, 38.4 kBaud
   //0xA0, 0x2F, 0x52,    // Modem 0, 1 and 2: NRZ, 76.8 kBaud
   0x75, 0xA0, 0x00,    // Freq A
   0x58, 0x32, 0x8D,    // Freq B
   0x01, 0xAB,          // FSEP 1 and 0
   0x40,                // PLL_RX
   0x30,                // PLL_TX
   0x6C,                // CURRENT_RX
   0xF3,                // CURRENT_TX
   0x32,                // FREND
   0xFF,                // PA_POW
   0x00,                // MATCH
   0x00,                // PRESCALER
   };

   // Calibration data
   RF_RXTXPAIR_CALDATA xdata RF_CALDATA;
```

```
  // Initialize peripherals
  WDT_ENABLE(FALSE);
  RLED_OE(TRUE);
  YLED_OE(TRUE);
  GLED_OE(TRUE);
  BLED_OE(TRUE);

  BLED = LED_OFF;
  RLED = LED_OFF;
  GLED = LED_OFF;
  YLED = LED_OFF;

  // Set optimum settings for speed and low power consumption
  MEM_NO_WAIT_STATES();
  FLASH_SET_POWER_MODE(FLASH_STANDBY_BETWEEN_READS);

      UART0_SETUP(57600, 14746, UART_NO_PARITY | UART_RX_TX |
UART_POLLED);

      P2DIR=0x00;          // LCD port is output
      lcd_init();
      lcd_com(15);
      lcd_goto(1,1);
      lcd_puts("    Ket Qua:");
      lcd_goto(2,1);
      lcd_puts("Nut1:chua co du lieu");
      lcd_goto(3,1);
      lcd_puts("Nut2:chua co du lieu");
      lcd_goto(4,1);
      lcd_puts("Nut13:chua co du lieu");

  // Build packet
      // first 2bytes is Source of packet (here is Master)
      // next 2 bytes is Destination of packet (target node)
```

```
        // next 2 bytes is type of data or command (temperature or pressure...)
        // next 2 bytes is real data or command.
        // the transmiting function will add 2 byte CRC16(packet) at the end of packet
        txDataBuffer.packet.source = Node_ID;
        txDataBuffer.packet.target = 0x00;
        txDataBuffer.packet.type   = 0x00;
        txDataBuffer.packet.dat    = 0x00;

        n=0;
        node = 1;
   while (TRUE)
        {
                // Calibration
                halRFCalib(&RF_SETTINGS, &RF_CALDATA);

                        txDataBuffer.packet.target = node;
        BLED = LED_ON;

                        // Turn on RF, send packet
        halRFSetRxTxOff(RF_TX, &RF_SETTINGS, &RF_CALDATA);

        halRFSendPacket(PREAMBLE_BYTE_COUNT, txDataBuffer.buffer,
PACKET_LENGTH);

                        BLED = LED_OFF;
                        YLED = LED_ON;

        // Turn on RF, receive ACK, turn off RF
        halRFSetRxTxOff(RF_RX, &RF_SETTINGS, &RF_CALDATA);
        result = halRFReceivePacket2(80, rxDataBuffer.buffer, PACKET_LENGTH);  // wait
for 1s

        YLED = LED_OFF;
                        RLED = LED_OFF;
```

```
                GLED = LED_OFF;

        // Success/failure indicators
        if ((result))// == PACKET_LENGTH) && (rxDataBuffer.packet.source == node) &&
(rxDataBuffer.packet.target == Node_ID) )
                {
                        GLED = LED_ON;
                 fTemp = rxDataBuffer.packet.dat;
                if(rxDataBuffer.packet.type == Presure)                 // Cam bien ap suat
            {
                                fTemp *= 0.925;
                fTemp -= 10.2;
                                lcd_goto(node+1,1);
                                sprintf(so,"Do sau: %3.1f cm  ",fTemp);
                                lcd_puts(so);
                        }
                else if(rxDataBuffer.packet.type == Temp_DS18B20) // Cam bien
nhiet do so DS18B20
                fTemp /= 16;
                        else if (rxDataBuffer.packet.type == Temp_LM35)
                        {
                                fTemp -= 492;
                fTemp /= 8.192;
                        }
                        lcd_goto(node+1,1);
                        sprintf(so,"Nut%1d:%2.1f",node,fTemp);
                        lcd_puts(so);
                        lcd_data(223);
                        lcd_puts("C        ");

                        if(rxDataBuffer.packet.type                             ==
Temp_DS18B20||rxDataBuffer.packet.type == Temp_LM35)
                        {
                                if(fTemp > Nguong1 && fTemp < Nguong2)
```

```
                                {
                                        lcd_goto(node+1,12);
                                        lcd_puts("-BaoDong1");
                                }
                                else if(fTemp > Nguong2 && fTemp < Nguong3)
                                {
                                        lcd_goto(node+1,12);
                                        lcd_puts("-BaoDong2");
                                }
                                else if(fTemp > Nguong3)
                                {
                                        lcd_goto(node+1,12);
                                        lcd_puts("-BaoDong3");
                                }
                        }

                        // Send to PC
                        printf("%u\t",rxDataBuffer.packet.source);
                        printf("MYNAME\t");
                        printf("%u\t",rxDataBuffer.packet.target);
                        printf("%u\t",rxDataBuffer.packet.type);
                        printf("%u\t",rxDataBuffer.packet.dat);
                        printf("  END\n");
}
                else
                {
  RLED = LED_ON;
                        lcd_goto(node+1,1);
                        sprintf(so,"Nut%1d: time out      ",node);
                        lcd_puts(so);
                }
        // sleep_ms(100);
        halWait(100, 14746);  //delay 100ms
        node++;
```

```
                if(node > Max_Node) node = 1;
    }

} // end of main()

// Flash interrupt handler (do nothing)
// We need to handle the interrupt even though we do not do anything.
// If not, the program will not run correctly except under the debugger,
// which has its own Flash interrupt handler

void FlashIntrHandler(void) interrupt INUM_FLASH {

    INT_SETFLAG(INUM_FLASH, INT_CLR);
    return;
}

byte halRFReceivePacket2(byte timeOut, byte* packetData, byte maxLength) {
    byte receivedBytes, pkgLen, crcData, i;
    word crcReg;

    halConfigTimer23(TIMER3|TIMER23_NO_INT_TIMER,                    10000,
CC1010EB_CLKFREQ);

    INT_SETFLAG(INUM_TIMER3, INT_CLR);
    TIMER3_RUN(TRUE);

    RF_SET_PREAMBLE_COUNT(16);
    RF_SET_SYNC_BYTE(RF_SUITABLE_SYNC_BYTE);
    MODEM1=(MODEM1&0x03)|0x24;   // Make sure avg filter is free-running + 22 baud
settling time
    INT_ENABLE(INUM_RF, INT_OFF);
    INT_SETFLAG(INUM_RF, INT_CLR);
    RF_START_RX();
```

```
while (1) {
   // Check if 10 ms have passed
   if (INT_GETFLAG(INUM_TIMER3)) {
      // Clear interrupt flag and decrement timeout value
      INT_SETFLAG(INUM_TIMER3, INT_CLR);
                        if (timeOut && !--timeOut) {
                                 timeOut=255;
         break;                  // Timeout
                        }
   }
   // Check if sync byte received
   if (INT_GETFLAG(INUM_RF)) {
      EXIF &= ~0x10; // Clear the flag
      break;
   }
}

receivedBytes=0;
// Timeout or sync byte received?
if (timeOut!=255) {

   // Lock average filter and perform RSSI reading if desired
   RF_LOCK_AVERAGE_FILTER(TRUE);

   // Get length of package
   RF_WAIT_AND_RECEIVE_BYTE( pkgLen );
   pkgLen+=2;              // Add the two CRC bytes

   // Initialize CRC-16
   crcReg=CRC16_INIT;

   // Receive as many bytes as packet specifies + 2 crc bytes
   while (pkgLen--) {
      RF_WAIT_AND_RECEIVE_BYTE( crcData );
```

```
        // If there is space in the buffer at _packetData_ and we're not
        // currently receiving CRC, store the byte
        if ( pkgLen>=2 && maxLength ) {
           *packetData++=crcData;
           receivedBytes++;
           maxLength--;
        }

        // Calculate CRC-16 (CCITT)
        for (i=0; i<8; i++) {
                if ( ((crcReg&0x8000)>>8) ^ (crcData&0x80) )
                        crcReg=(crcReg<<1)^CRC16_POLY;
                else
                        crcReg=(crcReg<<1);
                crcData<<=1;
        }
     }

     // Check if CRC is OK
     if (crcReg != CRC_OK)
        receivedBytes=0;
  }

  TIMER3_RUN(FALSE);
  RF_SET_PREAMBLE_COUNT(RF_PREDET_OFF);
  return receivedBytes;
}
```

3. Slave.c

```
#include <chipcon/reg1010.h>
#include "CC1010ND.H"
#include "PACKET.H"
#include <chipcon/hal.h>
```

```
#include <stdio.h>

// Node ID constants
#define Node_ID                 0x03
#define Master                  0xFF

// Data type
#define Temp_LM35               0x01
#define Temp_DS18B20            0x02
#define Presure                 0x03

// Protocol const
#define PREAMBLE_BYTE_COUNT     10
#define PREAMBLE_BITS_SENSE     16

// Variable

PACKET xdata txDataBuffer;
PACKET xdata rxDataBuffer;

extern byte halRFReceivePacket2(byte timeOut, byte* packetData, byte maxLength);

//MAIN PROGRAM
void main(void) {
        byte result;

        // X-tal frequency: 14.745600 MHz
        // RF frequency A: 868.277200 MHz Rx
        // RF frequency B: 868.277200 MHz Tx
        // RX Mode: Low side LO
        // Frequency separation: 64 kHz
        // Data rate: 2.4 kBaud
        // Data Format: Manchester
        // RF output power: 4 dBm
```

```
        // IF/RSSI: RSSI Enabled

        RF_RXTXPAIR_SETTINGS code RF_SETTINGS = {
   //0x4B, 0x2F, 0x15,  // Modem 0, 1 and 2: Manchester, 2.4 kBaud
   0x43, 0x2F, 0x15,  // Modem 0, 1 and 2: NRZ, 2.4 kBaud
   //0xA1, 0x2F, 0x29,    // Modem 0, 1 and 2: NRZ, 38.4 kBaud
   //0xA0, 0x2F, 0x52,  // Modem 0, 1 and 2: NRZ, 76.8 kBaud
   0x75, 0xA0, 0x00,    // Freq A
   0x58, 0x32, 0x8D,    // Freq B
   0x01, 0xAB,          // FSEP 1 and 0
   0x40,                // PLL_RX
   0x30,                // PLL_TX
   0x6C,                // CURRENT_RX
   0xF3,                // CURRENT_TX
   0x32,                // FREND
   0xFF,                // PA_POW
   0x00,                // MATCH
   0x00,                // PRESCALER
   };

   // Calibration data
   RF_RXTXPAIR_CALDATA xdata RF_CALDATA;
        WDT_ENABLE(FALSE);
   // Setup UART0 with polled I/O
   UART0_SETUP(57600, CC1010EB_CLKFREQ, UART_NO_PARITY | UART_RX_TX |
UART_POLLED);
        // ADC setup
        halConfigADC(ADC_MODE_SINGLE   |   ADC_REFERENCE_INTERNAL_1_25,
CC1010EB_CLKFREQ, 0);
        ADC_SELECT_INPUT(ADC_INPUT_AD1);
        ADC_POWER(TRUE);
   // Initialize peripherals
   RLED_OE(TRUE);
   YLED_OE(TRUE);
```

```
GLED_OE(TRUE);
BLED_OE(TRUE);
      BLED = LED_OFF;
RLED = LED_OFF;
GLED = LED_OFF;
YLED = LED_OFF;
// Set optimum settings for speed and low power consumption
MEM_NO_WAIT_STATES();
FLASH_SET_POWER_MODE(FLASH_STANDBY_BETWEEN_READS);
// Ready for acknowledge
      txDataBuffer.packet.source = Node_ID;
      txDataBuffer.packet.target = Master;
      txDataBuffer.packet.type   = Temp_LM35;
      txDataBuffer.packet.dat    = 0x00;
// Calibration
halRFCalib(&RF_SETTINGS, &RF_CALDATA);
while (TRUE) {
   // RX
      YLED = LED_ON;
      // Turn on RF, receive test string packet,
      halRFSetRxTxOff(RF_RX, &RF_SETTINGS, &RF_CALDATA);
   result = halRFReceivePacket2(0, rxDataBuffer.buffer, PACKET_LENGTH);
      YLED = LED_OFF;
            GLED = LED_OFF;
            RLED = LED_OFF;
            BLED = LED_OFF;
            // Success/failure indicators
   if (result == PACKET_LENGTH)
            {
      GLED = LED_ON;
                  if(rxDataBuffer.packet.target == Node_ID)
                  {
                        BLED = LED_ON;
                        // Power up the ADC and sample the temperature
```

```
                    ADC_SAMPLE_SINGLE();
                    txDataBuffer.packet.dat= ADC_GET_SAMPLE_10BIT();

            halRFSetRxTxOff(RF_TX, &RF_SETTINGS, &RF_CALDATA);
            halRFSendPacket(PREAMBLE_BYTE_COUNT, txDataBuffer.buffer,
PACKET_LENGTH);
                    BLED = LED_OFF;
                }
    }
            else
        RLED = LED_ON;
  }

} // end of main()
```