Comunicação sem fio TinyOS

Saymon Castro de Souza

Orientador: Prof. Dr. José Gonçalves Pereira Filho

Agenda

- Introdução
- Interfaces básicas de comunicação
- Interfaces – Active Message

Introdução

- TinyOS **provê um conjunto de interfaces** para **abstrair** detalhes do serviço de **comunicação**.
- Por consequência, há um conjunto de **componentes** que **implementam** essas **interfaces**.

Introdução

- Visão geral
  - *message_t* é o buffer de mensagem do TinyOS
  - Envia o buffer de mensagem para o rádio
  - Recebe um buffer de mensagem do rádio.

Introdução

- Todas essas interfaces e componentes usam a mesma abstração *message buffer*.
  - message_t
    - É um tipo abstrato de dados (TAD)
    - É implementado como uma struct nesC.
    - Informações são lidas ou escritas por meio de funções.

Message Buffer

```
typedef nx_struct message_t {
  nx_uint8_t header[sizeof(message_header_t)];
  nx_uint8_t data[TOSH_DATA_LENGTH];
  nx_uint8_t footer[sizeof(message_footer_t)];
  nx_uint8_t metadata[sizeof(message_metadata_t)];
} message_t;
```

Os campos header, footer e metadata não devem ser acessados diretamente.
É recomendado acessar os campos desta estrutura apenas via Packet.

Interfaces básicas de comunicação

Interfaces Básicas de Comunicação

- Há um conjunto de interfaces que usam o *message_t* como estrutura de dados.
  - *Packet*
  - *Send*
  - *Receive*
  - *PacketAcknowledgements*
  - *RadioTimeStamping*

Interfaces básicas de comunicação

- *Packet*
  - Provê o acesso básico para o TAD *message_t*, tais como:
    - Prover comandos para **limpar** o conteúdo de uma mensagem.
    - Obter o **tamanho do *payload***
    - Obter o **ponteiro para a área de *payload*** da mensagem.

Interfaces Básicas de Comunicação

- *Send*
  - Fornece a interface básica para envio de mensagens sem endereço.
    - Provê comandos para **enviar** uma mensagem e **cancelar** um envio pendente.
    - Provê um evento para indicar se uma mensagem foi **enviada com sucesso ou não**.
    - Provê funções para obter a **capacidade máxima do *payload*** da mensagem, incluindo o **ponteiro para o *payload*** da mensagem.

Interfaces Básicas de Comunicação

- *Receive*
  - Fornece uma interface básica para recepção de mensagens.
    - Provê um evento para **receber mensagens**.
    - Provê comandos para obter o **tamanho do *payload*** da mensagem, incluindo o **ponteiro para o *payload*** da mensagem.

Informações Básicas de Comunicação

- *PacketAcknowledgements*
  - Fornece um mecanismo para solicitar **confirmações de pacotes.**
- *RadioTimeStamping*
  - Fornece informações de **data e hora de transmissão e recepção**.

Interfaces Active Message

Interfaces Active Message

- Como é muito comum ter **vários serviços** usando o **mesmo rádio** para se comunicar, o TinyOS fornece a camada *Active Message* (AM) para **multiplexar o acesso ao rádio**.
  - *AMPacket*
  - *AMSend*

Interfaces Active Message

- *AMPacket*
  - Semelhante ao Packet, fornece os acessadores AM básicos para o tipo de dado abstrato message_t. Essa interface fornece comandos para obter o endereço AM de um nó, um destino de pacote AM e um tipo de pacote AM. Comandos também são fornecidos para definir o destino e tipo de um pacote AM e verificar se o destino é o nó local.

Interfaces Active Message

- *AMSend*
  - Semelhante ao Send, fornece a interface básica de envio de mensagens ativas. A principal diferença entre o AMSend e o Send é que o AMSend recebe um endereço AM de destino em seu comando send.

Componentes

Componentes

- AMReceiverC
  - Fornece as seguintes interfaces: Receive, Packet, AMPacket.
- AMSenderC
  - Fornece AMSend, Packet, AMPacket, PacketAcknowledgements.

Referência

- http://tinyos.stanford.edu/tinyos-wiki/index.php/Mote-mote_radio_communication

Obrigado !