Introduction to SimpliciTI

Low-power RF protocol from Texas Instruments

Outline

- Overview – What is SimpliciTI?
- Device types and network topologies
- SimpliciTI software architecture
- Example: How to configure SimpliciTI devices
- Insight on packet format and addressing
- Supported hardware platforms
- Demonstration: Temp sensor network

What is SimpliciTI?

SimpliciTI is:

- Low Power: a TI proprietary low-power RF network protocol
- Low Cost: uses < 8K FLASH, 1K RAM depending on configuration
- Flexible: simple star w/ extendor and/or p2p communication
- Simple: Utilizes a very basic core API
- Versatile: MSP430+CC110x/2500, CC1110/2510, CC1111/CC2511, CC2430, CC2520
- Low Power: Supports sleeping devices

Application Areas

SimpliciTI supports:

- alarm & security: occupancy sensors, light sensors, carbon monoxide sensors, glass-breakage detectors

Analog Meter

- smoke detectors

- remote controls

- AMR: gas meters, water meters, e-meters

- home automation: garage door openers, appliances, environmental devices

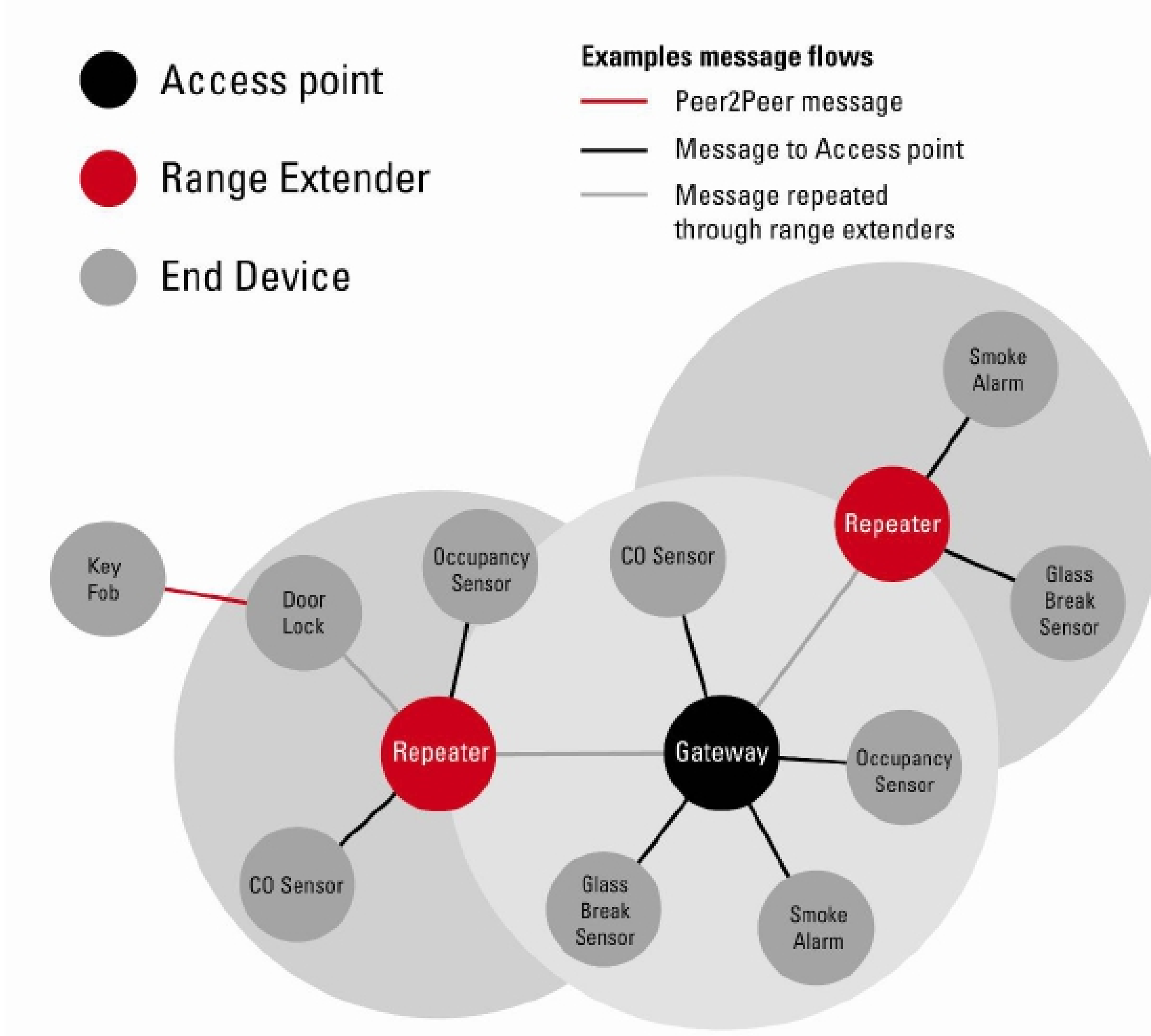
Outline

- Overview – What is SimpliciTI?
- Device types and network topologies
- SimpliciTI software architecture
- Example: How to configure SimpliciTI devices
- Insight on packet format and addressing
- Supported hardware platforms
- Demonstration: Temp sensor network

SimpliciTI Network topology
wireless sensing application

• Range can be extended through repeaters.

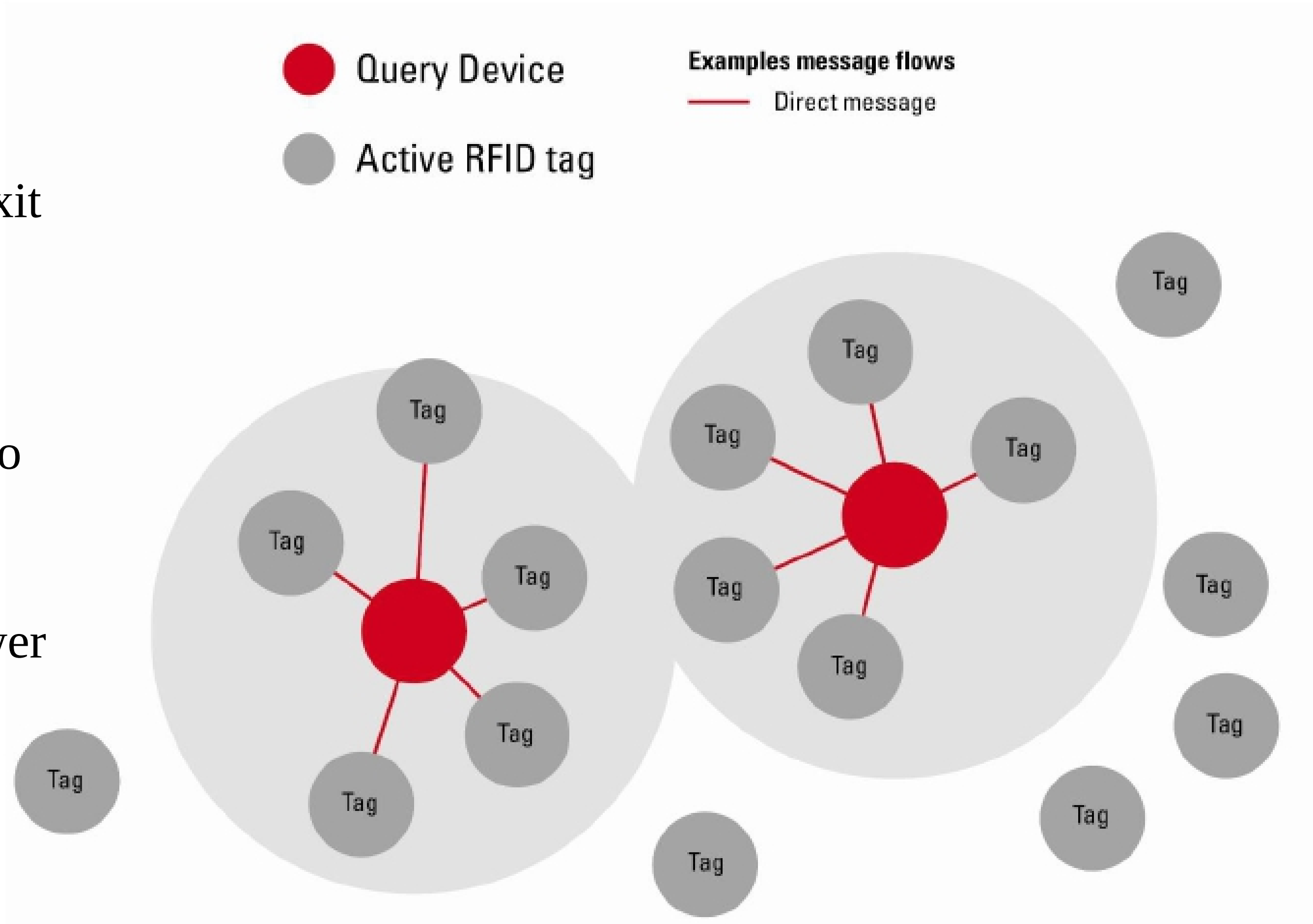
• The circles represent range of gateway and extended range of repeaters.



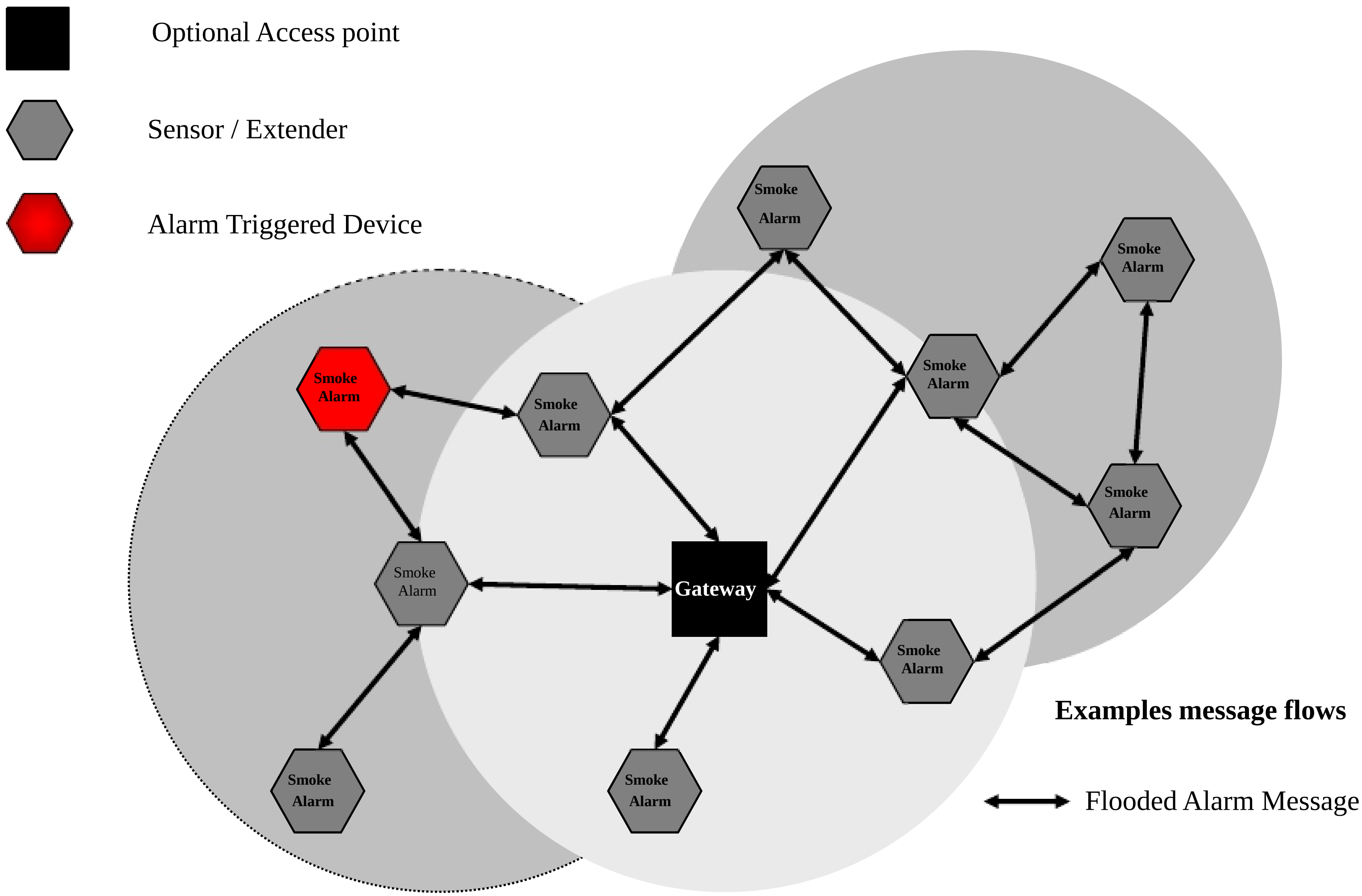
SimpliciTI Network topology
Active RF tags

• Active RF tags typically enter and exit the network ad-hoc.

• Tags must be able to quickly associate to the network while maintaining low power consumption.



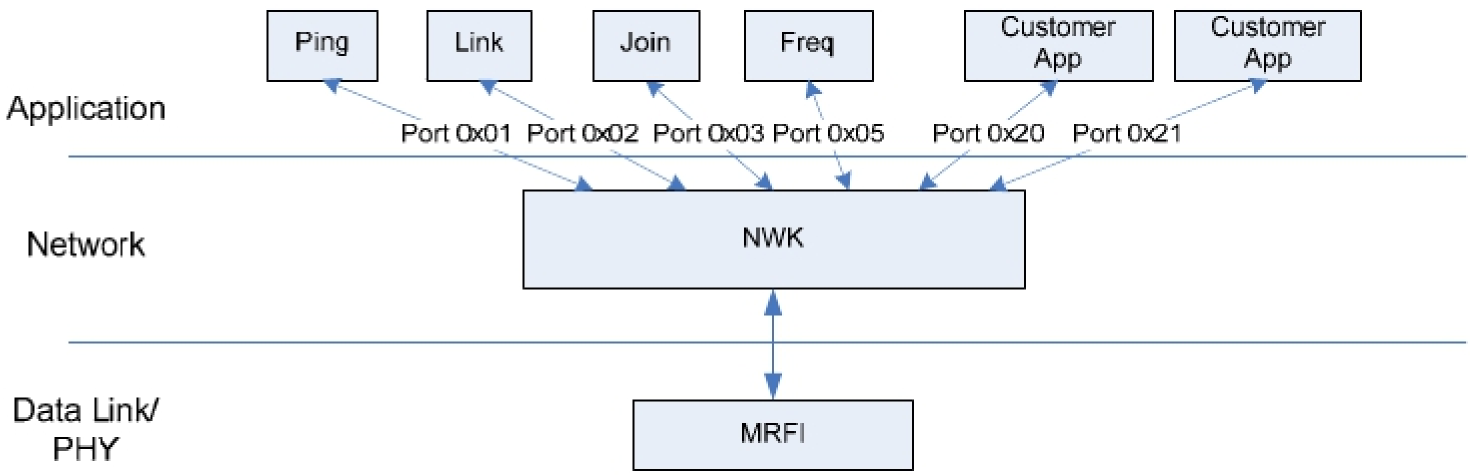
SimpliciTI Network topology
Smoke Detector System



Outline

- Overview – What is SimpliciTI?
- Device types and network topologies
- SimpliciTI software architecture
- Example: How to configure SimpliciTI devices
- Insight on packet format and addressing
- Supported hardware platforms
- Demonstration: Temp sensor network

Architectural Overview



- Layers
  - MRFI (“minimal RF interface”)
  - NWK
  - nwk applications (modules)
  - customer applications

- Network Support
  - init
  - ping
  - link / linklisten
  - nwk mgmt
  - send / receive
  - I/O

Application Programming Interface (API)

- initialization
  - smplStatus_t SMPL_Init(uint8_t (*callback)(linkID_t));
- linking (bi-directional by default)
  - smplStatus_t SMPL_Link(linkID_t *linkID);
  - smplStatus_t SMPL_LinkListen(linkID_t *linkID);
- peer-to-peer messaging
  - smplStatus_t SMPL_Send(lid, *msg, len);
  - smplStatus_t SMPL_Receive(lid, *msg, *len);
- configuration
  - smplStatus_t SMPL_Ioctl(object, action, *val);

Simple Configuration

- operational mode (type)
- power mode (sleep support)
- topology
- addressing / identification
- RAM allocation
  - packet size
  - buffer sizes
  - # supported links (connections)
- security tokens
- messaging (hop ct, repeaters)
- radio (freq, crypto key, modulation, CCA parameters)

```
/* FROM smpl_config.dat */

// Number of connections supported
-DNUM_CONNECTIONS=4

// Maximum size of application payload
-DMAX_APP_PAYLOAD=20

// size of low level queues for sent and received frames.
-DSIZE_INFRAME_Q=2
-DSIZE_OUTFRAME_Q=2

// default Link token
-DDEFAULT_LINK_TOKEN=0x01020304

// default Join token
-DDEFAULT_JOIN_TOKEN=0x05060708

// this device's address.
-DTHIS_DEVICE_ADDRESS="{0x79, 0x56, 0x34, 0x12}"

// device type
-DEND_DEVICE

// for End Devices specify the Rx type.
//-DRX_LISTENS
//-DRX_POLLS
//-DRX_NEVER
-DRX_ALWAYS
```

Runtime Configuration

- radio frequency
- encryption key
- app access to frame header
- app access to radio controls
- AP nwk mgmt control

Object	Description	Comments
IOCTL_OBJ_FREQ	Get/Set radio frequency	Frequency agility. May be used by **APP** or **NWK**.
IOCTL_OBJ_CRYPTKEY	Set encryption key	Customer may provide external means for user to set a non-default key. Requires reset to take effect.
IOCTL_OBJ_RAW_IO	Application layer access to the frame header to directly send or receive a frame.	This object is used for example to ping another device where the network address of the target device is supplied directly and not done through the connection table.
IOCTL_OBJ_RADIO	Application layer access to some radio controls.	Limited access to radio directly. For example, sleeping and awakening the radio and getting signal strength information.
IOCTL_OBJ_AP_JOIN	Access Point join-allow context	Interface to control whether Access Point will allow devices to join or not.

Outline

- Overview – What is SimpliciTI?
- Device types and network topologies
- SimpliciTI software architecture
- Example: How to configure SimpliciTI devices
- Insight on packet format and addressing
- Supported hardware platforms
- Demonstration: Temp sensor network

Example
How to configure Access Point

- star hub in the network (1 / net)
- always-on (acts as range extender)
- store and fwd for sleeping devices
- linking and token (link and join) mgmt
- AP can implement end device functionality (link listen, receive)

```
// Initialize the HW/Radio
BSP_Init();  // initialize the BSP (API subject to change)
SMPL_Init(0);

// Handle Linking
SMPL_LinkListen(&linkID1);

// Receive Messages
While (1) {
   while((SMPL_SUCCESS == SMPL_Receive(linkID1, msg, &len) {
      // do something
}}
```

Example
How to configure Range Extender

- always-on device
- repeats received frames (with limitations)
- limited to 4 / net (although flexible in design)

```
// Initialize the HW/Radio
BSP_Init();
SMPL_Init(0);

// No Linking or application level functionality
while(1) ;
```

Example
How to configure End Device

- poll for data
  - polling is Port specific
  - no data results in blank (empty) response
- API e.g. Sequence
  - Init (and Join)
  - Link (assumes listen)
  - Sample Temp
  - Send
- option to sleep

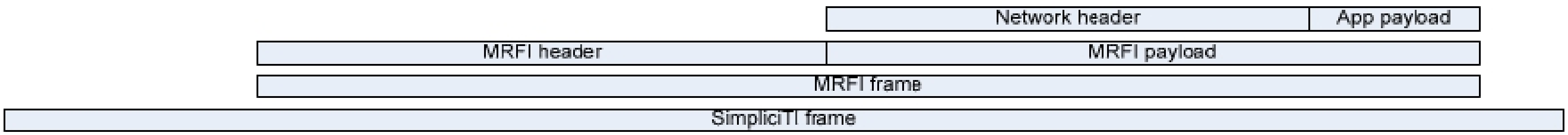
```
void main()
{
  linkID_t linkID;
  uint32_t    temp;

  // Initialize the board's HW
  BSP_Init();
  SMPL_Init(0);
  // link.
  SMPL_Link(&linkID);

  while (TRUE)
  {
    // sleep until timer.  read temp sensor
    MCU_Sleep();
    HW_ReadTempSensor(&temp);
    if (temp > TOO_HIGH)
    {
      SMPL_Send(linkID, "Hot!", 4);
    }
    if (temp < TOO_LOW)
    {
      SMPL_Send(linkID, "Cold!", 5);
}}}
```

Outline

- Overview – What is SimpliciTI?
- Device types and network topologies
- SimpliciTI software architecture
- Example: How to configure SimpliciTI devices
- Insight on packet format and addressing
- Supported hardware platforms
- Demonstration: Temp sensor network

Packet Format

PREAMBLE	SYNC	LENGTH	MISC	DSTADDR	SRCADDR	PORT	DEVICE INFO	TRACTID	App Payload	FCS
RD*	RD*	1	RD*	4	4	1	1	1	*n*	RD*

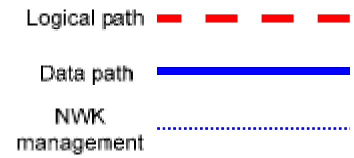


*RD: Radio-dependent populated by MRFI or handled by the radio itself

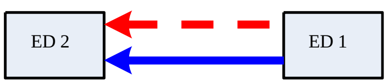
- preamble: hw sync
- sync: hw sync
- length: bytes non-phy
- dstaddr
- srcaddr
- port: app port number
- dev info: capabilities
- tractid: transaction nonce or seq num
- app pyld: 0 <= n <= 52 byte/113 byte (radio dependent)
- crc: must be valid

Addressing and Communication

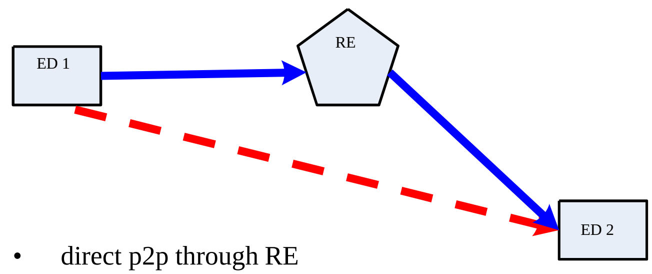
- net address = hw addr (4 byte) + app port
  - statically assigned hw addr
  - no address resolution mechanism



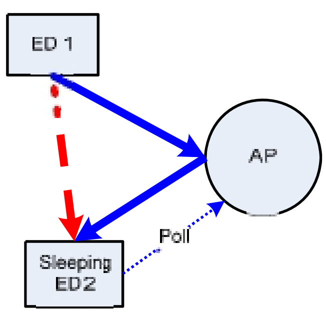
- byte 1: 0x00, 0xFF – reserved for broadcast
- communication topologies:



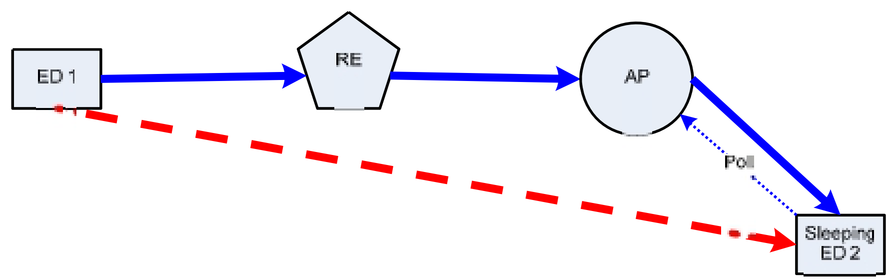
- direct peer-2-peer



- direct p2p through RE



- store and fwd p2p through AP



- store and fwd p2p through RE and AP

Additional Details

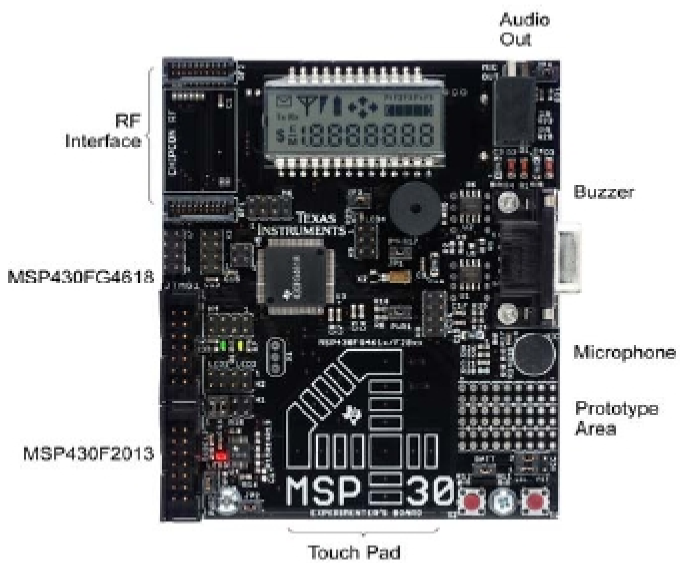
- CCS development environment
- minimal hw abstraction
- no driver support (UART, SPI, LCD, Timers)
- no heap utilization
- no runtime (nwk) context storage
- single thread (app), no tasks or scheduling
- nwk api is synchronous (does not return until operation is complete)
- retries and acks must be managed by app

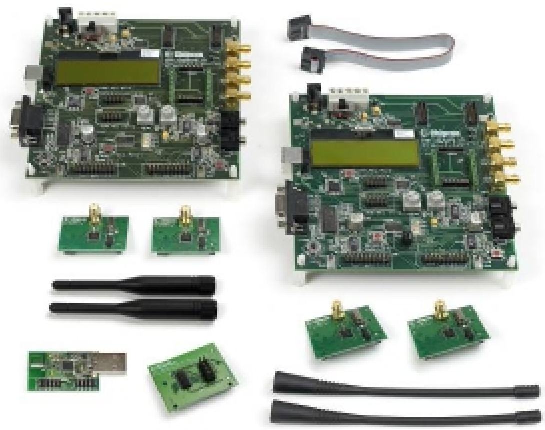
Outline

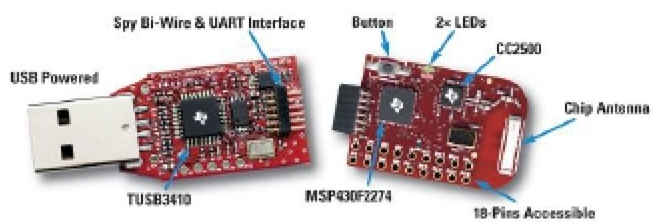
- Overview – What is SimpliciTI?
- Device types and network topologies
- SimpliciTI software architecture
- Example: How to configure SimpliciTI devices
- Insight on packet format and addressing
- Supported hardware platforms
- Demonstration: Temp sensor network

Hardware Support

- MSP-EXP430FG4618 Experimenters Board
  - (MSP430FG4618) w/ Socket Interface for CC110x / CC2500
- eZ430RF-2500
  - MSP430F2274 + CC2500
- CC2510-CC2511DK and CC1110 CC1111DK
- DSSS (MSP430 +CC2420, CC2430)
- CC2520



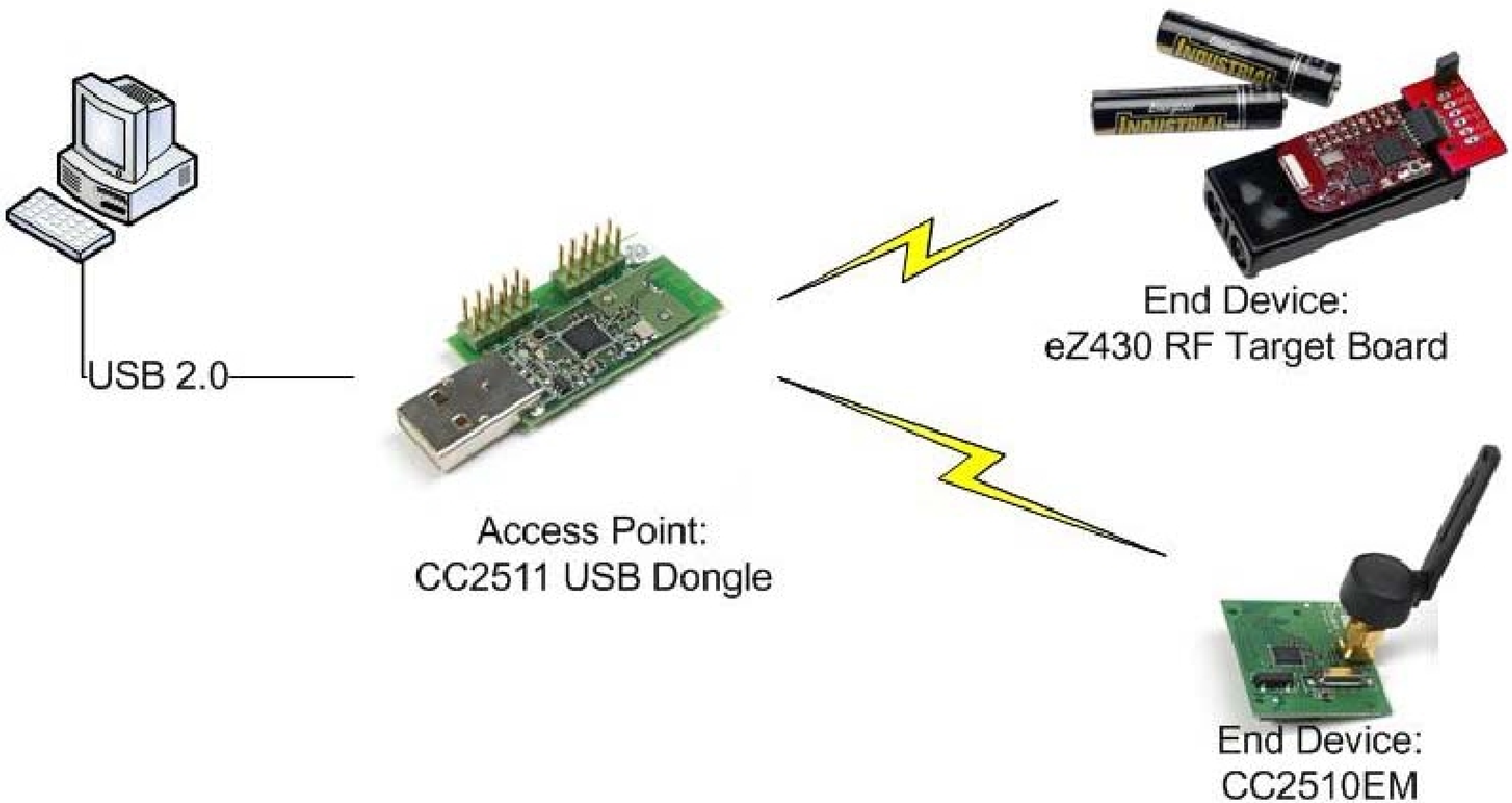




Outline

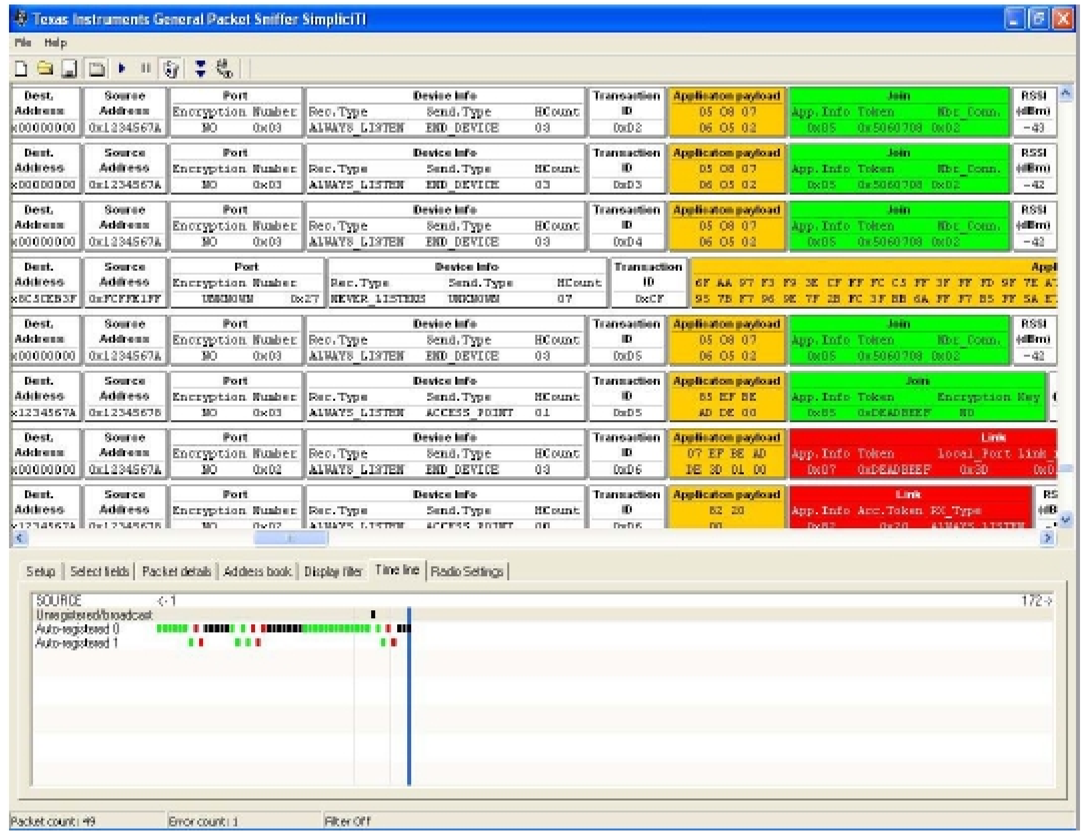
- Overview – What is SimpliciTI?
- Device types and network topologies
- SimpliciTI software architecture
- Example: How to configure SimpliciTI devices
- Insight on packet format and addressing
- Supported hardware platforms
- Demonstration: Temp sensor network

Example
Hardware configuration



Development Tools
Packet sniffer

- two end devices are reading their internal temperature sensor
- 1/sec they report their value to the access point
- the access point feeds the data to a terminal window on the PC via a virtual COM port
- all RF traffic can be monitored with the TI SimpliciTI packet sniffer



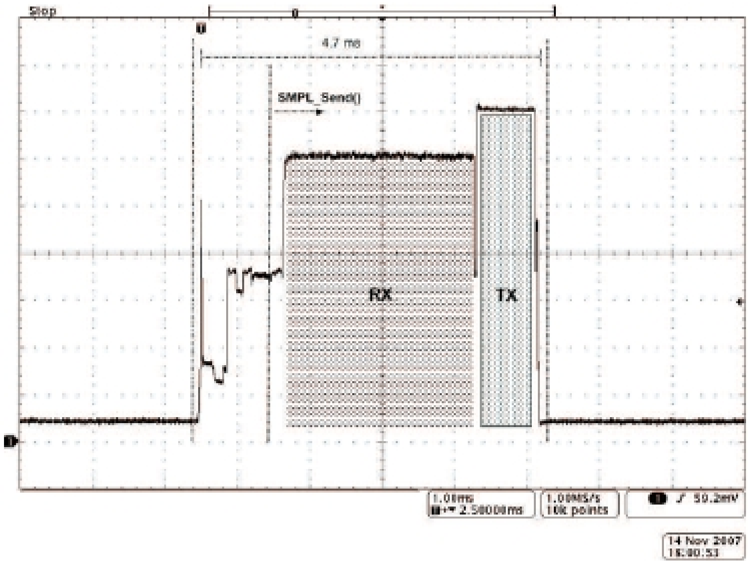
Packet sniffer screenshot

Current Consumption
How to estimate and measure?

- Guideline to SimpliciTI current consumption as presented in application note:

- Wireless Sensor Monitor Using the eZ430-RF2500.

- http://www.ti.com/litv/pdf/slaa378a



Available examples

Where	What	Notes
SimpliciTI distribution	SimpliciTI examples: - 2 ED with bi-di - AP as data hub - Cascading ED - Simple polling with AP	
eZ430-RF2500	- Temp.Sens network with PC gui	- Distributed with eZ430-RF2500. - Comes with app.note

www.ti.com/simpliciti