Základní struktura aplikace

První podmínkou pro aplikaci je integrace knihovny do vaší aplikace – je potředa includovat patřičné hlavičkové soubory a říct překladači, kde se nachází zdrojové kódy aplikace.

Druhou podmínkou je zavolání funkce SYS_Init() před jakoukoliv jinou funkcí z knihovny LWM a volání funkce SYS_TaskHandler() co nejčastěji. Tato funkce se stará o obsluhu fyzické a síťové vrstvy, samotný příjem a odesílání dat. Z těchto podmínek vychází doporučená struktura aplikace:

```
static void APP_TaskHandler(void)
{
      // Put your application code here
}
int main(void)
{
      SYS_Init();
      while (1)
      {
      SYS_TaskHandler();
      APP_TaskHandler();
      }
}
```

Funkce APP_TaskHandler() se stará o spouštění uživatelského kódu. Tímto kódem může být cokoliv, v příkladech obsažených v knihovně se jedná obsluhu stavového automatu, v případě Peer2Peer programu se obsluhují dva stavy, APP_STATE_INITIAL a APP_STATE_IDLE. Pokud je aplikace ve stavu APP_STATE_INITIAL, spustí se funkce appInit() a následně aplikace přejde do stavu APP_STATE_IDLE.

```
static void APP_TaskHandler(void)
{
switch (appState)
{
case APP_STATE_INITIAL:
{
appInit();
appState = APP_STATE_IDLE;
} break;

case APP_STATE_IDLE:
break;

default:
break;
}
}
```

Konfigurace aplikace

Každá aplikace by měla prostřednictvím souboru config.h měnit výchozí parametry stacku, případně nastavovat parametry, které výchozí hodnoty nemají. Je také doporučeno konfigurovat zde parametry pro váš aplikační kód ve formátu APP_nazev.

Stack má následující parametry:

NWK_BUFFERS_AMOUNT – počet bufferů pro stack (pro příjem, odesílání a směrování). Teoreticky stačí jen jeden, pokud chcete potvrzování rámců, jsou potřeba 2. Doporučená hodnota je minimálně 3.

NWK_DUPLICATE_REJECTION_TABLE_SIZE – počet záznamů v tabulce duplicitního příjmu. Tato tabulka se využívá pro detekci a následné zahození duplicitních rámců, zejména z důvodu:

- Smyček v síti – vrátí se ten samý rámec
- Přeposlané broadcasty

Záznamy zůstávají v tabulce po dobu definovanou NWK_DUPLICATE_REJECTION_TTL v milisekundách. Během této doby nemohou být smazány. Pokud není v tabulce místo, nové příchozí rámce nejsou zpracovány.

NWK_DUPLICATE_REJECTION_TTL – viz výše. Čas by měl odpovídat alespoň dvojnásobku času propagace rámce na konec sítě a zpět.

NWK_ROUTE_TABLE_SIZE – počet záznamů ve směrovací tabulce. Číslo by mělo v ideálním případě odpovídat počtu uzlů v síti. Pokud ne, mělo by být co nejvyšší.

NWK_ROUTE_DEFAULT_SCORE – výchozí hodnota „kvality“ cesty. Počet selhání, po kterých je směrovací záznam odstraněn.

NWK_ACK_WAIT_TIME – doba čekání na potvrzení rámce, čas by měl opět odpovídat alespoň dvojnásobku času propagace rámce na konec sítě a zpět.

NWK_GROUPS_AMOUNT – počet skupin kterých může být uzel členem (multicast….)

NWK_ROUTE_DISCOVERY_TABLE_SIZE – počet záznamů v tabulce Route Discovery – počet souběžných route discovery request

NWK_ROUTE_DISCOVERY_TIMEOUT – doba platnosti záznamu v Route Discovery Table

NWK_ENABLE_ROUTING – povolení směrování, pokud je potřeba.

NWK_ENABLE_SECURITY – podpora šifrování na síťové vrstvě

NWK_ENABLE_MULTICAST – podpora multicastu

NWK_ENABLE_ROUTE_DISCOVERY – směrování pomocí AODV discovery, zároveň musí být povoleno NWK_ENABLE_ROUTING

NWK_ENABLE_SECURE_COMMANDS – zabezpečení samotné síťové vrstvy a jejích příkazů, nejen payloadu

SYS_SECURITY_MODE – výběr šifrovacího algortmu, pokud je šifrování povoleno:

- 0 – Hardware accelerated AES-128 (pouze pokud podporuje HW)
- 1 – Software XTEA (všechny platformy)

PHY_ENABLE_ENERGY_DETECTION – povelení přístupu k energy detection z PHY vrstvy

PHY_ENABLE_RANDOM_NUMBER_GENERATOR – povolení přístupu ke generátoru náhodných čísel

Application Endpoint – příjem dat

LWM stack umožňuje využívat jistou formu „socketů“ na síťové vrstve. V terminologii LWM jsou nazývány Application Endpoints.

Při odesílání dat, je situace jednoduchá – do pole dstEndpoint zapíšete patřičné číslo endpointu. Povoleny jsou hodnoty 1 – 16. Více viz odesílání dat.

Při příjmu je situace složitější, je potřeba patřičný socket/endpoint otevřít, registrovat. Je potřeba stacku říct, že rámec daným způsobem zpracujete, jinak ho zahodí.

Registrace endpointu je vhodné dělat při startu aplikace, provádí se následovně:

```
//Registrace endpointu

NWK_OpenEndpoint(CISLO_EP, funkceObluhy);

//obsluha prichozich ramcu
static bool funkceObluhy (NWK_DataInd_t *ind)
{

  // obsluha

  return true;
```

Přijatá data jsou obsažena proměnné *ind*, terá je typu *NWK_DataInd_t.* Ten de definován následovně (významy jednotlivých položek jsou zřejmé):

```
typedef struct NWK_DataInd_t
{
  uint16_t     srcAddr;
  uint16_t     dstAddr;
  uint8_t      srcEndpoint;
  uint8_t      dstEndpoint;
  uint8_t      options;
  uint8_t      *data;
  uint8_t      size;
  uint8_t      lqi;
  int8_t       rssi;
} NWK_DataInd_t;
```

Odesílání dat

K odesílání dat složí funkce *NWK_DataReq(data_k_odeslani)*, kde data_k_odeslani jsou struktury typu *NWK_DataReq_t*. Definice typu je uvedeny níže, významy položek jsou zřejmé.

```
typedef struct NWK_DataReq_t
{
  // service fields
  void         *next;
  void         *frame;
  uint8_t      state;

  // request parameters
  uint16_t     dstAddr;
  uint8_t      dstEndpoint;
  uint8_t      srcEndpoint;
  uint8_t      options;
#ifdef NWK_ENABLE_MULTICAST
  uint8_t      memberRadius;
  uint8_t      nonMemberRadius;
#endif
  uint8_t      *data;
  uint8_t      size;
  void         (*confirm)(struct NWK_DataReq_t *req);

  // confirmation parameters
  uint8_t      status;
  uint8_t      control;
} NWK_DataReq_t;
```

U položky options se jednotlivé volby dají kombinovat pomocí logického součtu (NWK_OPT_ACK_REQUEST | NWK_OPT_ENABLE_SECURITY). Jednotlivé volby jsou uveden v tabulce 5-2 v LWM developers guide.

Položka confirm je ukazatel na funkci provádějící obsluhu potvrzení daného requestu. Funkce může mít následující strukturu:

```
static void appDataConf(NWK_DataReq_t *req)
{
if (NWK_SUCCESS_STATUS == req->status)
// odeslano v poradku
else
// obsluha chyby, např. znovuodeslání
}
```

Jednotlivé chybové kódy jsou uvedeny v tabulce 5-3 v LWM developers guide.