به نام خدا

پروژه درس سیستم های هوشمند

Lorawan

دکتر سید احمد معتمدی

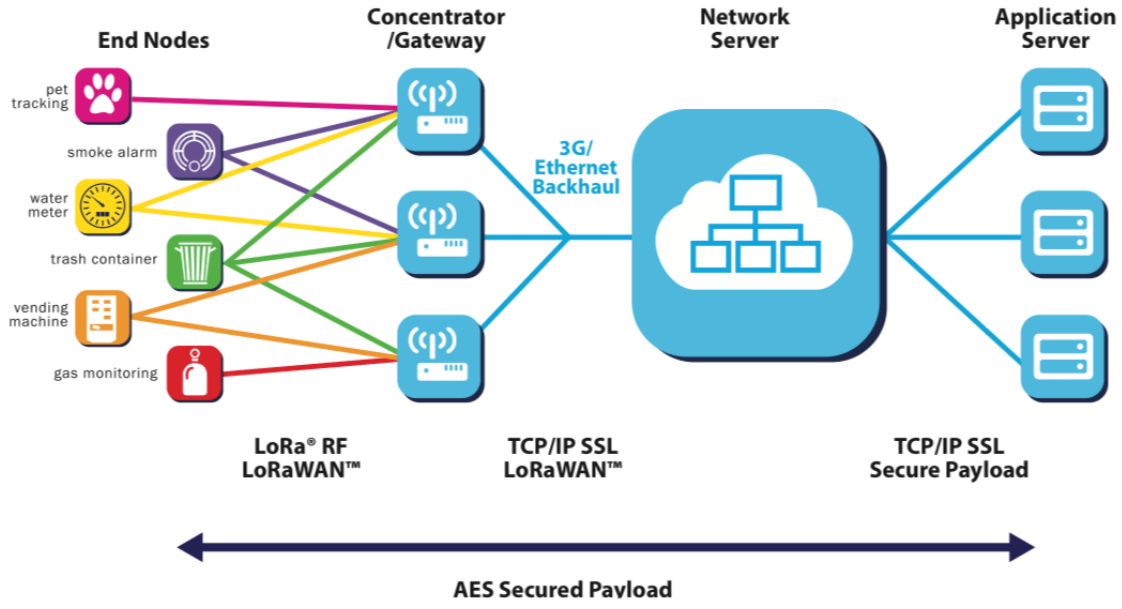
حسین غلامی

تعریف شبکه: LoRaWan

شبکه لورا (LoRaWAN) یکی از پروتکل‌های اصلی دوربرد توان پایین LPWAN ویژه اینترنت اشیا است. این فناوری به سیگنال‌ها اجازه می‌دهد تا حتی در سطوح پایین‌تر از نویز نیز منتشر و بازیابی شوند. تجهیزات مبتنی بر LoRaWAN می‌توانند تا سال‌ها فقط با یک باطری کار کنند. شاید مهمترین مشخصه شبکه لورا (LoRaWAN) که توانسته است در کنار مزیت‌های فنی این پروتکل زمینه رشد سریع آن را فراهم کند، رویکرد غیر انحصاری توسعه این پروتکل بر بستر یک جامعه آزاد و با مشارکت مجموعه‌های مختلف فناوری باشد.

معماری فنی شبکه لورا(LoRaWAN)

همان‌طور که اشاره شد، شبکه لورا (LoRaWAN) یک پروتکل ارتباطی LPWAN ویژه اینترنت اشیا در باندهای فرکانسی بدون نیاز به مجوز (ISM) است که می‌تواند محدوده وسیعی را با توان مصرفی پایین تحت پوشش قرار دهد. این فناوری توسط شرکت Semtech و جامعه‌ای از شرکت‌های بزرگ حوزه فناوری (همچون IBM, Cisco, HP, Foxconn) که LoRa Alliance نام دارد، توسعه یافته و پشتیبانی می‌شود.



معماری ساختار یک شبکه لورا (LoRaWAN) همان‌طور که در شکل بالا آمده است از دستگاه‌های انتهایی مبتنی بر لورا) سنسورها و عملگرها که اصطلاحا End-Device خوانده می‌شوند(، گیت‌وی‌ها(LoRaWAN Gateways) ، سرور شبکه و نهایتا اپلیکیشن و نرم‌افزار کاربر تشکیل شده است.

توپولوژی شبکه لورا به صورت ستاره‌ای (Star of Stars) است. دستگاه‌های انتهایی اطلاعات را به از طریق شبکه لورا (LoRaWAN) به گیت‌وی ارسال می‌کنند. پس از دریافت داده توسط گیت‌وی، گیت‌وی اطلاعات را بر روی یک لینک ارتباطی مبتنی بر اینترنت به سمت سرور شبکه می‌فرستد. این لینک ارتباطی می‌تواند توسط شبکهLTE/3G ، Ethernetو یا شبکه‌های داخلی طراحی شود. سپس اطلاعات توسط سرور شبکه در اختیار نرم‌افزار کاربران قرار می‌گیرد.

در حقیقت گیت‌وی و سرور شبکه مانند یک واسطه بین نرم‌افزار کاربر و دستگاه‌های انتهایی عمل می‌کند و امکان رسیدن داده به نرم‌افزار را فراهم کند. در شبکه لورا (LoRaWAN) داده‌ها به صورت کامل (End-to-End) بین دستگاه‌ها و اپلیکیشن کاربر از طریق رمزگذاری AESارسال می‌شود. از این رو امنیت اطلاعات کاربران نیز تضمین می‌شود.

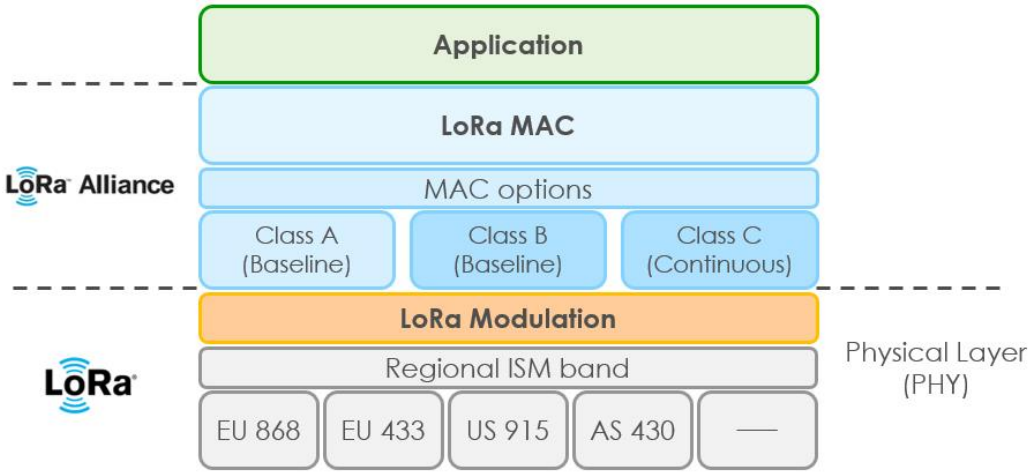
فناوری LoRaWAN با بکارگیری لینک متقارن، امکان ارتباط کاملا دو سویه را فراهم می‌کند؛ این مساله به ویژه در سرویس‌های اینترنت اشیاء که نیاز به ارسال دستورهای کنترلی از سمت سرور به تجهیزات انتهایی را دارند، بسیار با اهمیت است.

در LoRaWAN نرخ ارسال داده مبتنی بر پروتکل لایه فیزیکی LoRa 27 kb/s است و هر گیت‌وی می‌تواند داده‌های هزاران دستگاه انتهایی را جمع‌آوری کند. همچنین پوشش رادیویی هر گیت‌وی شبکه لورا (LoRaWAN) در مناطق باز و حومه شهر تا ۱۵ کیلومتر نیز می‌رسد.

لایه فیزیکی و مدولاسیون شبکه لورا (LoRaWAN)

اگرچه در بسیاری موارد شبکه LoRaWAN در کلام بطور مختصر لورا (LoRa) خوانده می‌شود، اما از نظر فنی، این دو متفاوت هستند. لورا پروتکل لایه فیزیکی یا مدولاسیون بیسیمی است که به منظور ایجاد لینک ارتباطی با ناحیه پوشش وسیع استفاده می‌شود. بسیاری از سیستم‌های پیشین به منظور رسیدن به توان پایین از مدولاسیون (FSK Frequency Shift Keying) در لایه فیزیکی بهره می‌بردند. اما لورا مبتنی بر مدولاسیون CSS (Chirp Spread Spectrum) است که علاوه بر فراهم آوردن خاصیت توان پایین مدولاسیونFSK ، ناحیه پوشش و نفوذپذیری را نیز به طور قابل توجهی افزایش می‌دهد. چندین دهه چنین مدولاسیونی به دلیل مقاوم بودن در برابر تداخل (Interference) و ناحیه پوشش وسیع، تنها در کاربردهای نظامی و ارتباطات فضایی استفاده می‌شد. اما LoRa اولین پیاده‌سازی کم هزینه و مناسب برای کاربردهای تجاری از چنین سیستمی است که در باندهای بدون نیاز به مجوز کار می‌کند.

پروتکل LoRaWAN در حقیقت پروتکلی در لایه بالاتر (MAC) است که بر پایه پروتکل LoRa توسعه یافته و کار می‌کند و امکان راه‌اندازی یک شبکه کامل را فراهم می‌سازد.



مدل توسعه شبکه لورا (LoRaWAN)

از نظر مدل توسعه، شبکه لورا (LoRaWAN) در مقابل شبکه SigFox استراتژی کاملا متفاوتی را اتخاذ کرده و مشارکت در تمام قسمت‌ها باز و ممکن است. عضویت در LoRa Alliance و مشارکت در توسعه و استفاده از استانداردهای این فناوری برای همه امکان‌پذیر است. هر شرکت سخت‌افزاری می‌تواند دستگاه‌های انتهایی و گیت‌وی را مطابق با استانداردهای شبکه لورا تولید کند. حتی تولید ماژول‌های رادیویی (لایه فیزیکی) که تا دو سال پیش تنها توسط Semtech انجام می‌گرفت، با فروش license به کمپانی‌های NXP وMicrochip ، از انحصار یک کمپانی خاص خارج شده است.

از این رو شبکه لورا (LoRaWAN) راهبرد توسعه بسیار منعطفی را پیش گرفته که در نتیجه آن به توسعه شبکه به یک شرکت خاص وابسته نیست. همین امر رشد این پروتکل را سرعت بخشیده است و علی‌رغم شروع دیرتر نسبت به SigFox هم‌اکنون در مناطق بیشتری پوشش رادیویی دارد. شبکه لورا تاکنون (سپتامبر ۲۰۱۸) در ۹۵ کشور وجود دارد و این روند توسعه در آینده سیاره‌ای هوشمند خواهد ساخت. همچنین مدل‌های متنوعی نیز از شبکه‌های کاملا خصوصی و خارج از بستر اینترنت، تا شبکه‌های عمومی با طرح‌های تجاری مختلف، بر بستر این فناوری شکل گرفته است. از سوی دیگر اکوسیستم باز شبکه لورا موجب شده است که این فناوری در بخش فنی نیز به سرعت توسعه و در این زمینه نسبت به سایر پروتکل‌ها پیشی گیرد.

تعریف پروژه

این پروژه برای درس سیستم های هوشمند دیجتال دانشگاه امیرکبیر توسط دکتر سید احمد معتمدی تعریف شد به این نحو است که در یک شبکه لورا ، میبایست ارتباط بین گره نهایی و بروکر فراهم شود و انتقال اطلاعات انجام شود.

برای این کار از از ماژول RN2483 ، stm32f103c8 استفاده شد. و به کمک command ها ماژول پیکربندی و انتقال اطلاعات انجام شد.

برای پیکر بندی ماژول و انتقال اطلاعات میتوان در 2 مود کاری ماژول عمل کرد

مد اول ABP ، مد دوم OTAA (هر کدام از مود ها شرح داده خواهد شد) پیاده سازی در مود ABP انجام شد.

مود ABP به این صورت است که hweui گره باید در گیت وی تعریف شده باشد

ولی در مود OTAA گیت وی تنظیم میشود و سپس گره ها به آن متصل میشوند.

دستورات در مود ABP برای پیکر بندی به شرح زیر است.

```
mac set nwkskey 12345678123456781234567812345678

ok

mac set appskey 12345678123456781234567812345678

ok

mac set devaddr 001B4D55

ok

mac set adr on

ok
```

radio set pwr 14

ok

mac save

ok

و همچنین دستورات در مود OTAA علاوه بر موارد فوق به شرح زیر است

mac set appkey 00112233445566778899AABBCCDDEEFF

ok

mac set appeui FEDCBA9876543210

ok

mac save

ok

حال در مود ABP در گیت وی باید گره ای جدید اضافه کرد و devEUI را برابر hweui گره قرار داد الان گره میتواند به گیت وی متصل شده و اطلاعات را منتقل کند.

برای مود OTAA تنها کافی است که گره ای جدید تعریف شود و تنظیمات در گره نهایی مطابق آن انجام شود.

آنگاه گره نهایی میتواند با دستور mac join abp یا mac join otaa به گیت وی متصل شود

و بعد از اتصال ، ack آن به صورت دستور accepted باز میگردد

حال میتوان اطلاعات را به دو صورت با تصدیق (ack) یا بدون آن ارسال کرد فرمت کلی ارسال به شکل زیر است

mac tx <type> <FPort> <payload>

که در آن نوع ارسال وپورت و داده مشخص شده

مثال:

mac tx uncnf 1 DEAD

ok

mac_tx_ok

حالت بدون تصدیق و mac_tx_ok به معنای ارسال از گره است.

mac tx cnf 1 BEEF

ok

mac_tx_ok

mac_rx 1

حالت با تصدیق و mac_rx 1 به معنای رسیدن بسته در طرف مقابل است.

پیاده سازی در میکروکنترلر stmf103c8

برای توضیح این بخش ابتدا یاداوری میشود که ، برای پیاده سازی از دو کانال uart استفاده شد و یکی برای ارتباط با ماژول RN2483 و دیگری برای پیاده سازی ترمینال و مشاهده ارتباط بین این دو آی سی

برای این کار ابتدا دو تابع send_command و send_terminal به شرح زیر نوشته شد:

```
void send_command(char * str){
	char t_str1;[100]
	sprintf(t_str1, str);
	HAL_UART_Transmit(&huart1,(unsigned char*)t_str1 ,strlen(t_str1),100);
	sprintf(t_str1,"\r\n");
	HAL_UART_Transmit(&huart1,(unsigned char*)t_str1 ,strlen(t_str1),100);
}
void send_terminal(char * str){
	char t_str1;[100]
	sprintf(t_str1, str);
	HAL_UART_Transmit(&huart2,(unsigned char*)t_str1 ,strlen(t_str1),100);
	sprintf(t_str1,"\r\n");
	HAL_UART_Transmit(&huart2,(unsigned char*)t_str1 ,strlen(t_str1),100);
}
```

همچنین ماشین حالتی نوشته شد که در هر ثانیه وضعیتش را بررسی میکند و به حالت بعدی میرود و دستورات را یکی پس از دیگری ارسال میکند:

```
switch (status){
                    case 0:
                        send_command(nwkskey_command);
                        HAL_UART_Receive_IT(&huart1,data_u,1);
                        status++;
                        break;
                    case 1:
                        if(recive_validity==0){
                            send_command(nwkskey_command);
                            send_terminal("nwskey send again");
                            HAL_Delay(1000);
                        }
                        else{
                            check_ok();
                            recive_validity=0;
                            send_terminal("ack of nwkskey");
                        }
                        break;
                    case 2:
                        send_command(appskey_command);
                        HAL_UART_Receive_IT(&huart1,data_u,1);
                        status++;
                        break;
                    case 3:
                        if(recive_validity==0){
                            send_command(appskey_command);
                            send_terminal("appskey send again");
                            HAL_Delay(500);
                        }
                        else{
                            check_ok();
                            recive_validity=0;
                            send_terminal("ack of appskey");
```

```
		}
		break;
		case 4:
		send_command(devaddr_command);
		HAL_UART_Receive_IT(&huart1,data_u,1);
		status++;
		break;
case 5:
		if(recive_validity==0){
			send_command(devaddr_command);
			send_terminal("devaddr send again");
			HAL_Delay(500);
		}
		else{
			check_ok();
			recive_validity=0;
			send_terminal("ack of devaddr");

		}
		break;
case 6:
		send_command("mac set adr on");
		HAL_UART_Receive_IT(&huart1,data_u,1);
		status++;
		break;
case 7:
		if(recive_validity==0){
			send_command("mac set adr on");
			send_terminal("adr on send again");
			HAL_Delay(500);
		}
		else{
```

```
                check_ok();
                recive_validity=0;
                send_terminal("ack of adr on");
        }
        break;
case 8:
        send_command(pwr_command);
        HAL_UART_Receive_IT(&huart1,data_u,1);
        status++;
        break;
case 9:
        if(recive_validity==0){
                send_command(pwr_command);
                send_terminal("radio pwr send again");
                HAL_Delay(500);
        }
        else{
                check_ok();
                recive_validity=0;
         send_terminal("ack of pwr 14");

        }
        break;
case 10:
        send_command("mac save");
        HAL_UART_Receive_IT(&huart1,data_u,1);
        status++;
        break;
case 11:
        if(recive_validity==0){
                send_command("mac save");
                send_terminal("mac save send again");
                HAL_Delay(500);
```

```
		}
		else{
			check_ok();
			recive_validity=0;
			send_terminal("ack of mac save");
			send_terminal("confguration is done");

		}
		break;
//_____________________________________________configurate done
case 12:
		send_command("mac join abp");
		HAL_UART_Receive_IT(&huart1,data_u,1);
		status++;
		break;
case 13:
		if(recive_validity==0){
			send_command("mac join abp");
			send_terminal("mac join abp send again");
			HAL_Delay(500);
		}
		else{
			check_ok();
			recive_validity=0;
			send_terminal("ack of mac join abp");

		}
		break;
case 14:
		if(recive_validity==0){
			send_terminal("wait for accepted");
			HAL_Delay(500);
```

```
                    if(binary_back_off==timeout){
                    send_terminal("connection is fail");
                    status--;
                    binary_back_off=0;
                    }

                    binary_back_off++;
          }
          else{
                    check_accepted();
                    recive_validity=0;
          }
          break;
case 15:
                    send_command("mac tx uncnf 1 salam");
                    send_terminal("send|salam|-udp");
                    HAL_Delay(1000);

          break;
```

در نهایت وارد وضعیت 15 شده و هر ثانیه یک بار اطلاعات را ارسال میکند.

وضعیت پایه ها را در شکل زیر مشاهده میکنیم: