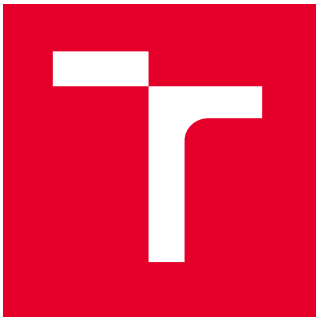


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

VYTVOŘENÍ LABORATORNÍ ÚLOHY PRO TECHNOLOGIE LPWA VYUŽÍVAJÍCÍ PROTOKOL LIGHTWEIGHT M2M (LWM2M)

LABORATORY DEMONSTRATOR FOR LPWA TECHNOLOGIES ENABLING THE COMMUNICATION VIA THE LIGHTWEIGHT M2M (LWM2M) PROTOCOL

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Radim Dvořák

AUTHOR

VEDOUCÍ PRÁCE — Ing. Pavel Mašek, Ph.D.

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Radim Dvořák ***ID:*** 186800

Ročník: 2 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Vytvoření laboratorní úlohy pro technologie LPWA využívající protokol Lightweight M2M (LWM2M)

POKYNY PRO VYPRACOVÁNÍ:

Cílem diplomové práce bude vytvoření laboratorní úlohy seznamující studenty s principem fungování Low-Power Wide Area (LPWA) technologií využívající Lightweight M2M (LWM2M)/CoAP protokol. Teoretická část práce bude obsahovat popis technologií NB-IoT a LTE Cat-M dle 3GPP Rel. 13/Rel. 14. Následně bude popsán zvolený protokol a rozdíly oproti aplikačním protokolům MQTT (SN). Cílem praktické části bude vytvoření prototypu zařízení obsahujícího zvolenou LPWA technologii. Následně bude vytvořena knihovna umožňující jednoduchý vývoj aplikací pro komunikaci skrze protokol LWM2M/CoAP. Vytvořená knihovna bude obsahovat detailní popis vytvořených funkcí a použitých programátorských rozhraní. Posledním krokem bude vytvoření laboratorní úlohy využívající vytvořenou knihovnu. Realizace bude probíhat v laboratoři Unilab na VUT v Brně ve spolupráci s VF.

DOPORUČENÁ LITERATURA:

[1] LIBERG, Olof. Cellular internet of things: From Massive Deployments to Critical 5G Applications. 1. Cambridge: Elsevier, 2019. ISBN 978-008-1029-022.

[2] LIBERG, Olof, Mařten SUNDBERG, Y.-P. Eric WANG, Johan BERGMAN a Joachim SACHS, [2018]. Cellular Internet of things: technologies, standards, and performance. San Diego, CA, United States: Academic Press, an imprint of Elsevier. ISBN 978-012-8124-581.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 24.5.2022

Vedoucí práce: Ing. Pavel Mašek, Ph.D.

Konzultant: Ing. Otto Zeman

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

S masivním nárůstem bezdrátových zařízení na trhu roste i poptávka po zajištění spolehlivého přenosu uživatelských dat. K zajištění spolehlivosti přenosu je nutné vybrat nejen vhodnou technologii pro samotný přenos dat, ale také vhodný přenosový protokol. Tato práce se zabývá popisem a srovnáním aktuálně dostupných LPWA (Low-Power Wide-Area) technologií licenčního spektra NB-IoT (Narrowband Internet of Things) a LTE-M (Long Term Evolution - Machine). Dále popisuje a srovnává aktuálně využívané přenosové protokoly pro implementaci v rámci M2M komunikace z hlediska parametrů komunikace a objemu přenesených dat. Také porovnává přenosové protokoly z hlediska zavedené režie, která do komunikace vnáší nadbytečná data pro zajištění spolehlivého přenosu, čímž zvyšuje celkový objem přenesených dat danou technologií. Práce se dále zabývá detailním popisem aplikačního protokolu LWM2M a také popisem programové knihovny v programovacím jazyce C++, která byla vytvořena pro účely této práce. Jako dodatek k této práci byla vytvořena laboratorní úloha využívající tuto knihovnu.

KLÍČOVÁ SLOVA

M2M, LPWA, licenční technologie, bezlicenční technologie, LoRaWAN, Sigfox, NB-IoT, LTE-M, UDP, TCP, MQTT, MQTT-SN, CoAP, LWM2M, mobilní technologie, 4G, 5G, 3GPP, mMTC, CIoT, IoT

ABSTRACT

As the number of wireless devices rises, so does the demand for reliable data transmission. To ensure the reliability of data transfers with such devices, suitable wireless technology, and a suitable transmission protocol have to be chosen according to the device capabilities and requirements. This thesis describes LPWA (Low-Power Wide-Area) technologies currently available on the market, both on licensed spectrum and non-licensed spectrum, focusing on licensed spectrum technologies (Narrowband Internet of Things, LTE-M) and also commonly used M2M enabled protocols for IoT devices. Furthermore, it compares those protocols from the communication parameters and data traffic perspectives with focus on overheads that further increase the amount of data transmitted but are necessary to ensure reliable data transmission. Lastly, it focuses on LWM2M protocol describing it to the detail and describes an LWM2M C++ library created as part of this thesis. In addition, a laboratory demonstrator was created utilizing this library.

KEYWORDS

M2M, LPWA, licensed spectrum technologies, non-licensed spectrum technologies, LoRaWAN, Sigfox, NB-IoT, LTE-M, UDP, TCP, MQTT, MQTT-SN, CoAP, LWM2M, mobile technologies, 4G, 5G, 3GPP, mMTC, CIoT, IoT

DVOŘÁK, Radim. *Vytvoření laboratorní úlohy pro technologie LPWA využívající protokol Lightweight M2M (LWM2M)*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 107 s. Diplomová práce. Vedoucí práce: Ing. Pavel Mašek, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Radim Dvořák
VUT ID autora:	186800
Typ práce:	Diplomová práce
Akademický rok:	2021/22
Téma závěrečné práce:	Vytvoření laboratorní úlohy pro technologie LPWA využívající protokol Lightweight M2M (LWM2M)

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Pavlu Maškovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále bych rád poděkoval Ing. Otto Zemanovi a Ladislavu Redayovi za konzultace a odbornou pomoc při řešení praktické části této práce.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

S neustále vyvíjejícími se technologiemi pro bezdrátový přenos masivně narůstá počet zařízení Internetu věcí vyžadující bezdrátovou konektivitu pro komunikaci M2M a tedy vzrůstají i nároky na přenosové technologie z hlediska počtu obsloužených zařízení a objemu přenesených dat. Pro obsluhu stále se zvyšujícího počtu zařízení bylo nutné neustále budovat komunikační infrastrukturu. V roce 2016 předložila standardizační skupina 3GPP (3rd Generation Partnership Project) iniciativu integrace zařízení IoT do již existujících mobilních sítí spravovaných operátory (licenční pásmo) a tedy eliminaci nutnosti budování nových komunikačních infrastruktur pro pokrytí technologiemi umožňující IoT (Internet of Things) komunikaci.

Technologie licenčního pásma umožňují přenos dat skrze vybudovanou infrastrukturu spravovanou operátorem bez nutnosti omezování komunikačních parametrů (vysílací střída, vysílací výkon, velikost zprávy), čímž je možné plně využít alokovaných přenosových kanálů, kde je kvalita služeb zajištěna operátorem oproti technologiím bezlicenčního pásma, kde více technologií může sdílet jeden komunikační kanál a může docházet k rušení komunikace.

Implementace IoT technologií pro přenos v licenčním frekvenčním pásmu, kde lze přenášet zprávy o vysokých objemech, umožnila implementaci stávajících přenosových protokolů i pro bezdrátová zařízení (MQTT (Message Queue Telemetry Transport), TCP (Transmission Control Protocol), UDP (User Datagram Protocol) a umožnila vytvoření nových přenosových standardů navržených pro potřeby masivní M2M (Machine to Machine) komunikace, tj., MQTT-SN (MQTT for Sensor Networks), CoAP (Constrained Application Protocol) a LWM2M (Lightweight M2M).

Tato práce se zabývá srovnáním technologií licenčního a bezlicenčního pásma, zejména pak technologiemi NB-IoT (Narrowband IoT) a LTE Cat-M pracující v licenčním pásmu, kdy je pozornost soustředěna na popis a následné srovnání přenosových protokolů využívaných v případě těchto technologií.

Samostatná část práce je věnována aplikačnímu protokolu LWM2M, vydaného společností OMA (Open Mobile Alliance), který umožňuje vzdálenou správu koncových zařízení a u kterého je momentálně předpoklad, že bude zastávat pozici jednoho z hlavních protokolů pro datové přenosy a správu zařízení v rámci technologií CIoT (Cellular IoT).

V rámci diplomové práce byla vytvořena uživatelská knihovna implementující aplikační protokol LWM2M pro nasazení v bezdrátových senzorových jednotkách Internetu věci. Knihovna implementuje protokol LWM2M dle standardu organizace OMA dle verze 1.0.

1 Technologie LPWA

LPWA (Low-power Wide Area) je skupina komunikačních technologií vyvinutých v důsledku neustálého nárůstu bezdrátových zařízení pro které je nutné připojení do světa internetu věcí (Internet of Things).

Technologie LPWA musejí umožňovat komunikaci velkého množství zařízení a poskytovat vysoké pokrytí signálem (několik km v zastavěných oblastech a desítky km v příměstských a mimo městských oblastí) a to na místech, kde je nedostatečné pokrytí konvenčních komunikačních technologií nebo daná technologie nesplňuje požadavky pro daný počet zařízení (GSM - Groupe Spécial Mobile, GPRS - General Packet Radio Service , LTE – Long Term Evolution) . Důraz u těchto technologií je také kladen na nízkou spotřebu energie, a tedy zvýšenou výdrž baterie ($>$ 10 let) a také nízkou cenu komunikačních modulů. Díky vysokému zastoupení měřících a senzorových jednotek je dále kladen důraz na spolehlivost přenosu s nízkou přenosovou rychlostí desítek až stovek b/s v závislosti na použité technologii. Díky své nízké přenosové rychlosti jsou LPWA technologie využívány téměř výhradně pro propojení zařízení do IoT (Internet of Things) a M2M (Machine-to-machine) komunikačních sítí.

Pro zvýšení spolehlivosti přenosu technologie LPWA využívají úzkopásmový přenos dat, díky kterému lze zvýšit odstup signálu od šumu SNR (Signal-to-Noise Ratio).

Technologie LPWA lze rozdělit na technologie využívající licenční pásmo, LTE-M (LTE for Machines) a NB-IoT (Narrowband Internet of Things), a bezlicenční pásmo (LoRaWAN, Sigfox).

1.1 Licenční vs bezlicenční pásmo

LPWA technologie využívající nelicenčního pásma (LoRaWAN, Sigfox aj.) využívají pro komunikaci jednoho z nelicenčních pásem. Tyto pásma se liší dle geografické lokace, ve které je zařízení/technologie aplikována. Pro evropský trh jsou nelicenční pásma ISM (Industrial, Scientific and Medical) definovaná Mezinárodní telekomunikační unií ITU (International Telecommunication Union) ve frekvenčních pásmech 169 MHz, 433MHz, 868 MHz a 2,4 GHz.

Výhodou bezlicenčního pásma je jeho bezplatné využití. Nevýhodou bezlicenčních pásem je masivní množství nestandardizovaných zařízení, a tedy vyšší využití komunikačního spektra. V určitém kmitočtovém pásmu lze provozovat více komunikačních technologií s různými parametry, a tedy může docházet k rušení komunikace. Z tohoto důvodu je nutné omezit vysílací výkon, omezit množství a délku zasílaných zpráv (Sigfox 140 Uplink, 4 Downlink), omezit využití kanálu daného zařízení, což

může mít za následek vyšší latenci přenosu zpráv, či jinak upravit přenos dat tak, aby nedocházelo k negativnímu ovlivnění ostatních komunikujících zařízení, popřípadě nedocházelo k zarušení komunikačního kanálu.

Technologie LPWA pracující v licenčním pásmu jsou integrovány do již fungujících sítí telekomunikačních operátorů. Díky tomu není zapotřebí budovat novou infrastrukturu pro pokrytí službami daných technologií a je zaručeno signálové pokrytí v oblastech, kde již fungují mobilní sítě předchozích generací. Technologie pracující v licenčním pásmu jsou také označovány jako technologie CIoT (Cellular Internet of Things) díky připojení do mobilních sítí fungující na buňkovém principu.

Výhodou technologií licenčního pásma je, že poskytovatel služeb má plnou kontrolu nad daným frekvenčním spektrem, které tak není přístupné pro další operátory či koncové uživatele využívající odlišné komunikační technologie, a tedy může zaručit vysokou kvalitu přenosu, zabezpečení dat, a stabilní přenosovou rychlost. Nevýhodou je nutnost standardizování připojení zařízení a zpoplatnění služeb pro připojení do sítě operátora [1].

Mezi CIoT jsou aktuálně dvě hlavní technologie (NB-IoT, LTE-M) u nichž se předpokládá nasazení do sítí 5G v rámci mMTC (massive Machine Type Communications), což umožní masivní nárůst zařízení IoT připojených do sítě. U těchto technologií, obzvláště LTE-M, je předpoklad, že by do budoucna mohly nahradit zastaralé technologie 2G a 3G [2].

LTE-M a NB-IoT jsou technologie vyvíjené na základě standardů definovaných organizací 3GPP (The 3rd Generation Partnership Project).

1.1.1 3GPP

3GPP je dohoda o spolupráci ve správě a vývoji celulárních mobilních technologií. Dohoda byla uzavřena v prosinci 1998 a sjednocuje sedm organizací vyvíjejících standardy pro mobilní komunikace [3].

Hlavním cílem organizace je standardizování a vytváření nových mobilních technologií. 3GPP průběžně publikuje nové technologie a úpravy stávajících technologií v tzv. „vydáních“ (Releases). Aktuálně se zabývá vytvářením standardů a implementací sítí páté generace 5G [3].

Obr. 1.1: Logo standardizační skupiny 3GPP

1.2 Technologie licenčního pásma (CIoT)

Jak již bylo zmíněno v předchozí sekci, technologie CIoT jsou technologie u kterých je předpokladem připojení do již existující celulární sítě spravované mobilním operátorem. Díky této skutečnosti se jedná o dlouhodobé řešení z hlediska globálního propojení zařízení IoT.

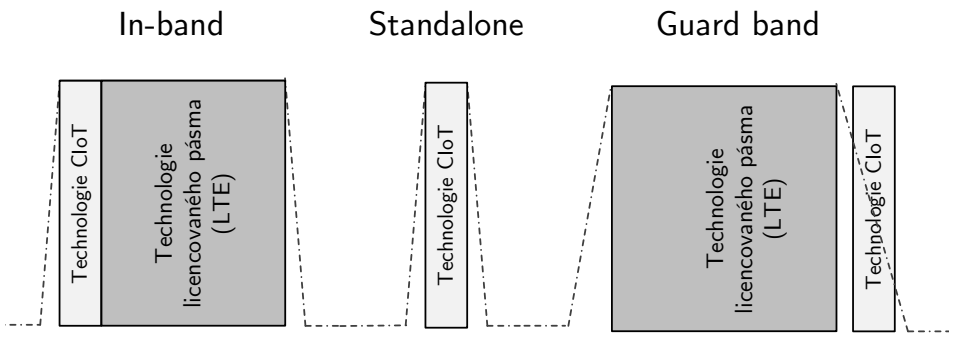
Šířka pásma technologií hraje důležitou roli při implementaci do stávajícího frekvenčního spektra. Mobilní operátoři, kteří chtějí technologie CIoT provozovat, musí řešit alokaci dostatečné šířky pásma pro danou technologii tak, aby vůbec, případně co nejméně, omezovala technologie, které již v daném frekvenčním spektru operují (Například LTE). Nasazení technologií do frekvenčního spektra lze řešit třemi způsoby:

- **Samostatné nasazení - Standalone deployment -** Daná technologie je nasazena do nevyužitého frekvenčního pásma. Je vhodná pro nasazení technologií s vyšší šířkou pásma [1].
- **Nasazení v ochranném pásmu – Guard band deployment** – Technologie je implementována do ochranného pásma stávající technologie operující v daném frekvenčním pásmu. Tato metoda nasazení je vhodná pro technologie s úzkou šířkou pásma [1].
- **Nasazení v pásmu jiné technologie – In-band deployment -** V této konfiguraci je technologie CIoT zasazena do frekvenčního spektra jiné technologie. V závislosti na šířce pásma technologie CIoT, je omezena efektivní šířka pásma nadřazené technologie a tedy je omezena i kvalita některých poskytovaných služeb (rychlost přenosu, zpoždění apod.) [1].

Aktuálně jsou k dispozici 2 hlavní technologie CIoT NB-IoT a LTE-M [2, 4]. V České republice je mobilním operátorem Vodafone od roku 2017 nasazena technologie NB-IoT pro komerční použití operující ve frekvenčním pásmu 800 MHz, kde je umístěna do ochranného intervalu (Guard band) technologie LTE, kterou Vodafone v tomto frekvenčním pásmu provozuje. Reálné nasazení technologie LTE-M je aktuálním tématem, kdy bude technologie umístěna jako standalone technologie ve frekvenčním pásmu 900 MHz. Do budoucna se zvažuje nasazení těchto dvou technologií ve frekvenčním pásmu 700 MHz, ve kterém aktuálně probíhá refarming frekvenčního spektra (přerozdělení částí frekvenčního spektra mobilním operátorům).

1.2.1 Metody úspory elektrické energie CIoT

V rámci úspory energie jsou skupinou 3GPP definovány 2 standardy pro úsporu elektrické energie PSM (Power Saving Mode) a eDRX (extended idle-mode Discontinuous Reception).



Obr. 1.2: Možné druhy nasazení technologií LPWA

Power Saving Mode (PSM)

Režim snížené spotřeby vydaný skupinou 3GPP ve vydání 12 (Release 12). Umožňuje maximální úsporu elektrické energie. Zařízení v módu PSM není schopné vysílat ani přijímat, přestává monitorovat paging[1], avšak nadále zůstává připojeno do komunikační sítě. Zařízení určuje dobu, která je nutná pro odeslání / přijetí zprávy a po kterou je nutné být aktivní [5].

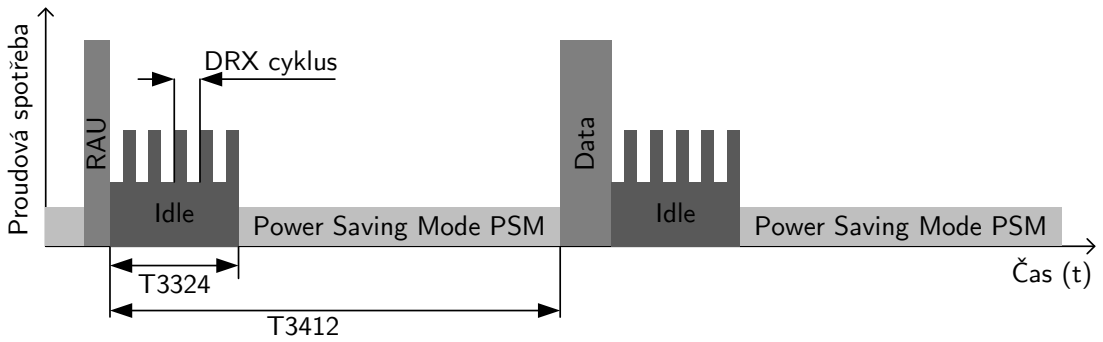
Zařízení mohou zažádat stanici o povolení režimu PSM během připojení, režimu TAU (Tracking Area Update) nebo při aktualizaci směrovací oblasti RAU (Routing Area Update). Koncové zařízení může žádost přijmout / odmítnout, popřípadě upravit některé parametry. Zařízení odvysílá stanici žádost o využití režimu PSM a po odvysílání svých zpráv přechází do režimu idle. Aktivuje 2 časovače T3324 a T3412. Časovač T3324 určuje dobu, po kterou zařízení naslouchá pagingu na kanále a po kterou je možné jej kontaktovat. Po vypršení časovače T3324 přechází do režimu PSM. Časovač T3412 určuje dobu trvání PSM + T3324. Po vypršení časovače T3412 je zařízení probuzeno a je připraveno odesílat/přijímat zprávy. Maximální délka trvání časovače T3412 je 413 dní a 8 hodin, časovače T3324 3 hodiny a 6 minut [5].

Extended idle-mode Discontinuous Reception (eDRX)

Režim snížené spotřeby vydaný ve vydání 13 (Release 13) skupiny 3GPP. Je založen na periodicky se opakujících časových intervalech, kdy je možné přijímat zprávy. V režimu eDRX je možné nastavení délky intervalů od 20,46 do 10485,76 sekund.

Zařízení si může u stanice vyžádat režim eDRX dovoluji-li mu to síť. Režim eDRX lze kombinovat s režimem PSM, dovoluje-li to síť. Stejně jako u PSM zařízení může žádat během připojení, TAU nebo RAU procedury a požadavek může být buď přijmut / odmítnut nebo upraven.

[1]Paging - Způsob signalizace stanice směrem ke koncovému zařízení o příchozí zprávě.



Obr. 1.3: Proudová spotřeba v režimech eDRX a PSM

1.2.2 NB-IoT

NB-IoT (Narrowband Internet of Things) je mobilní technologie spadající do kategorie standardů CIoT. Poprvé byla popsána skupinou 3GPP ve vydání 13 (Release 13) a později rozšířena navazujícím vydáním 14 (3GPP Release 14). Technologie NB-IoT byla navržena pro potřeby připojení velkého množství zařízení s nízkým nárokem na objem přenesených dat, kde lze tolerovat vyšší hodnoty přenosového zpoždění a s vyšším nárokem na pokrytí oblastí. NB-IoT cílí především na energeticky nenáročné koncové zařízení napájené z baterií s důrazem na cenu komunikačního modulu < 120 Kč ($< 5\$$) [4].

Obr. 1.4: Logo technologie NB-IoT

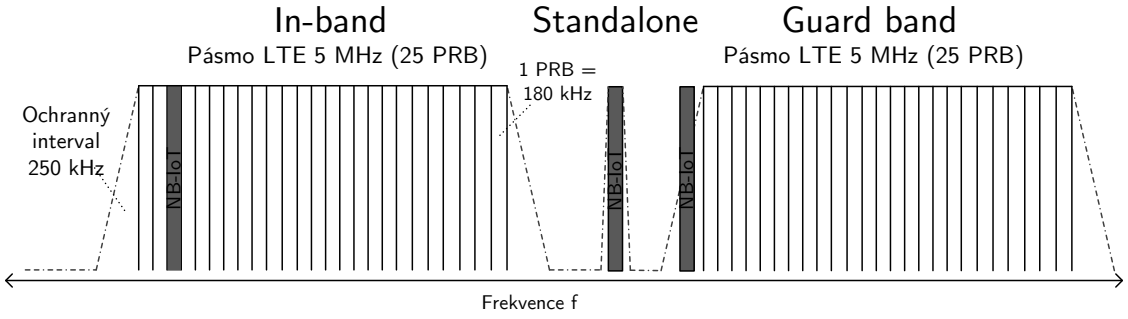
NB-IoT respektuje komunikační principy definované pro technologii LTE, a proto lze najít mnoho společných rysů včetně signalizačních mechanismů, které však byly pro použití v NB-IoT výrazně zjednodušeny s ohledem na energetickou a cenovou náročnost. Pro mnohonásobný přístup NB-IoT využívá ortogonální frekvenční dělení OFDMA (Orthogonal Frequency Division Multiple Access) a SC-FDMA (Single Carrier – Frequency Division Multiple Access). Technologie NB-IoT byla navržena pro šířku pásma 180 kHz, což přímo koresponduje se šířkou jednoho zdrojového bloku (Resource Block) technologie LTE, a tedy může využívat až 12 subnosných o šířce 12 kHz ve směru downlink i uplink. Maximální možná přenosová rychlost technologie NB-IoT (NB1) je 27 kb/s ve směru downlink a 32 kb/s ve směru uplink [4].

Technologie NB-IoT je navržena pro stacionární zařízení, tudíž nepodporuje funkci hand-over, jak tomu je u LTE a LTE Cat-M1.

Operační módy NB-IoT

Díky šířce pásma 180 kHz, což odpovídá jednomu zdrojovému bloku LTE, je možné nasazení technologie NB-IoT v následujících operačních módech:

- **Stand alone -** technologie NB-IoT je umístěna volného pásma technologie GSM. Pásmo GSM je rozděleno na bloky o šířce 200 kHz, čímž implementováním technologie NB-IoT do jednoho z těchto bloků efektivně vzniká 10 kHz ochranné pásmo z obou stran komunikačního kanálu [4].
- **Guard band -** komunikační kanál je umístěn do ochranného pásma nosných LTE sítě, a tedy je využito zdrojových bloků nevyužitých technologií LTE. V tomto pásmu jsou pro synchronizaci využity výhradně signály, které zcela náleží do ochranného intervalu [6].
- **In-band -** komunikační kanál je umístěn uvnitř pásma nosných technologie LTE a je využito zdrojových bloků LTE. V rámci tohoto operačního módu není možné využití kteréhokoliv zdrojového bloku, nýbrž pouze bloků definovaných v tabulce 1 [2][4].



Obr. 1.5: Operační módy NB-IoT

Rozšířené úrovně pokrytí NB-IoT ECL (Extended Coverage Level)

Jak již bylo zmíněno v předchozí kapitole, technologie NB-IoT je navržena tak, aby bylo možné k síti připojit i zařízení v prostorách s nízkou úrovní rádiového signálu (podzemní garáže, zařízení v odlehlých lokalitách). Z tohoto důvodu technologie definuje tři základní třídy rozdělení na základě rádiových podmínek daných naměřenou hodnotou MCL (Maximum Coupling Loss).

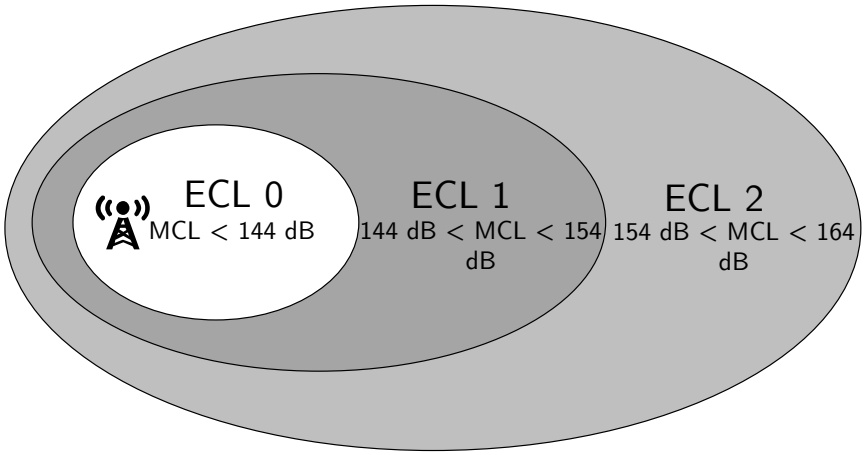
- **ECL 0 - Normal** - pro místa s normálními rádiovými podmínkami, MCL $<$ 144 dB.

- **ECL 1 - Robust** - pro místa zhoršeným pokrytím rádiovým signálem, 144 dB < MCL < 154 dB
- **ECL 2 - Extreme** - pro místa se špatnými rádiovými podmínkami, 154 dB < MCL < 164 dB [7, 8]

Kde MCL:

$$MCL_{dB} = P_{TX} - (NoiseFigure + SINR^2 + NoiseFloor) \tag{1.1}$$

Na základě přijaté síly signálu z koncového zařízení a síly signálu indikované zařízením, vysílací stanice (eNodeB– evolved Node B – název základní stanice pro systémy LTE) rozhodne o třídě, do které zařízení spadá. Přiřazená třída pokrytí ECL ovlivňuje počet opakování vyslání zprávy. [8]



Obr. 1.6: Grafické zobrazení tříd ECL

Rádiové rozhraní NB-IoT (Air interface)

NB-IoT využívá poloduplexní systém komunikace, tedy zařízení mohou v jeden moment zprávy pouze přijímat nebo odesílat.

Ve směru downlink využívá NB-IoT OFDM (Orthogonal Frequency Division Multiplexing) s modulací QPSK (Quadrature Phase Shift Keying, 2 bity v jednom stavu) nebo modulací BPSK (Binary Phase Shift Keying, 1 bit na jeden stav). Ve 180 kHz kanálu NB-IoT je 12 subnosných po 15 kHz nazývané jako tóny (tones). Ve směru downlink může přijímat data v 1 ms subrámci (2 sloty) pouze jedno zařízení, a tedy může využít všech 12 tónů [1].

Ve směru uplink je zachováno řízení přístupu LTE SC-FDMA (Single Carrier-Frequency Division Multiple Access) s modulacemi QPSK a BPSK. V případě, že

[2]SINR - Signal to interference plus Noise Ratio.

zařízení dokáže přijímat data ze sítě, ale není schopno vysílat dostatečným výkonem pro přijetí zprávy na eNodeB (například z důvodu špatného pokrytí nebo nedostatečné antény), může zařízení použít tóny po 3,75 kHz. Tím se může vysílací výkon soustředit na užší šířku kanálu což zvýší pravděpodobnost přijetí signálu na stanici eNodeB. K této situaci může dojít u zařízení ve třídě ECL 2. Na rozdíl od downlinku, kde jedno zařízení využívá všech 12 subnosných, u uplinku může být jednomu zařízení alokováno 3, 6 nebo 12 subnosných, čemuž se říká „multitóní přenos“ (Multitone Transmission) [1].

LTE Cat-NB2

V navazujícím vydání 3GPP 14 (Release 14) byla popsána nová kategorie koncových zařízení technologie NB-IoT, LTE Cat-NB2. Dosavadní technologie NB-IoT (LTE Cat-NB1) byla rozšířena o některé, dosud nepodporované služby a dosud poskytované služby byly vylepšeny [9]. Byla zvýšena celková přenosová rychlost na 160 kb/s pro downlink a 127 kb/s pro uplink.

Mezi nejzásadnější změny ve vydání 14 patří:

- Představena podpora lokazilace pro technologii NB-IoT za pomocí protokolu LPP (Location and Positioning Protocol), který využívá metody OTDOA (Observed Time Difference of Arrival) a E-CID (Enhanced Cell Identity) [9].
- Definice nové výkonové třídy o výkonu 25,12 mW (14 dBm, Power Class 6) pro rozšíření možností úspory elektrické energie.
- Rozšíření technologie o podporu multicastových přenosů
- Představení podpory pro mobilitu
- Implementace mechanismů RAI (Release Assistance Indication), který umožňuje dřívější uvolnění spojení mezi koncovým zařízením a stanicí nejsou-li očekávaná žádná další data pro uplink i downlink.

V navazujícím vydání 3GPP 15 (Release 15) byla přidána podpora pro TDD (Time Division Multiplexing) a podpora pro WUS (Wake Up Signal). V běžném provozu, v režimech DRX nebo eDRX, koncové zařízení v pravidelných intervalech monitoruje, zda nepřichází paging ze stanice (viz. kapitola eDRX) na kontrolních kanálech NPDCCH (Narrowband Physical Downlink Control Channel) a PDSCH (Physical Downlink Shared Channel). S implementací WUS, již koncové zařízení nemusí na těchto kanálech naslouchat. Díky tomu je možné odpojit hardwarovou část zodpovědnou za dekódování těchto kanálů, což má za následek další snížení spotřeby elektrické energie. ENB může zaslat koncovému zařízení signál WUS, tedy příkaz k aktivování monitoringu kanálů NPDCCH a PDSCH pro případný příchozí paging [9].

1.2.3 LTE-M

Technologie LTE-M jsou technologie založené na standardu LTE popsané ve vydání 3GPP 13. Předchůdcem technologií LTE-M byla technologie Cat-1, poprvé představena ve vydání skupiny 3GPP 8 (Release 8). Byla vyvinuta pro zjednodušení LTE zařízení s ohledem na jeho sníženou spotřebu zařízení a pro které není požadována přenosová rychlost vyšší než 10 Mbit/s. Pro celkové zjednodušení komplexity mohou být Cat-1 zařízení opatřena pouze jednou anténou, tedy nemusí podporovat MIMO (Multiple Input, Multiple Output) [1].

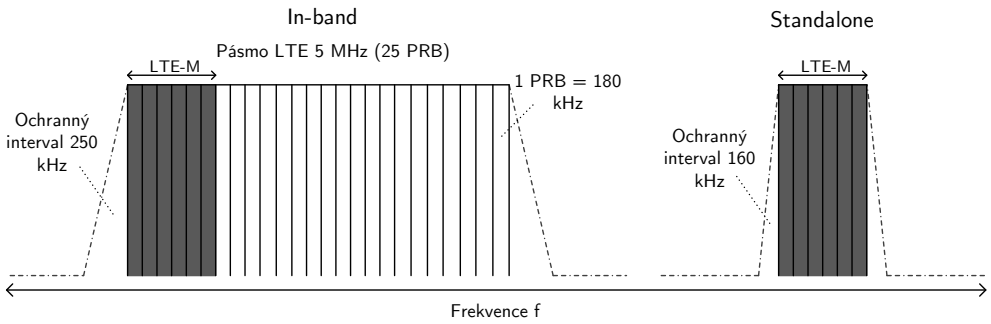
Další technologií, navazující na technologie Cat-1 je technologie Cat-0 vydaná ve vydání 3GPP 12 (Release 12), která dále zjednodušuje komplexitu zařízení zavedením volitelného poloduplexního provozu a omezením přenosové rychlosti na 1 Mbit/s. Zařízení Cat-0 využívají pouze jednu anténu, SISO (Single Input, Single Output). Zároveň technologie Cat-0 nově implementuje režim PSM pro úsporu elektrické energie [1].

Ve vydání 3GPP 13 (Release 13) byla popsána technologie první technologie eMTC LTE-M, Cat-M1 (dříve pouze Cat-M), u které došlo k výrazným změnám oproti předchozím technologiím. Technologie LTE Cat-M1 dále snižuje složitost zařízení až o 80 % oproti Cat-1 a zužuje šířku pásma z 20 MHz na 1,4 MHz [1]. Tímto krokem technologie LTE Cat-M1 vylepšuje úroveň pokrytí sítě minimálně o 15 dB, čímž umožňuje komunikaci s vyšším dosahem a vyšší kvalitou. Zároveň zachovává kompatibilitu se staršími systémy LTE (legacy). LTE Cat-M1 je navržena s ohledem na sníženou spotřebu zařízení ($>$ 10 let), a proto implementuje režimy pro snížení spotřeby PSM a eDRX. Pro potřeby nové generace LPWA technologií byla také ve vydání 3GPP 13 definována nová úroveň vysílacího výkonu 20 dBm (100 mW, Power Class 5) a 23 dBm (200 mW, Power Class 3).

Obr. 1.7: Logo technologie LTE-M

Technologie LTE Cat-M1 je navržena pro přímou implementaci do již existujícího pásma LTE a využívá šesti fyzických zdrojových bloků (PRB) technologie LTE se šířkou pásma 180 kHz. Cat-M1 může využívat operačních módů In-band a standalone. Celková šířka pásma Cat-M1 je 1,08 MHz bez ochranných pásem (6. 180

kHz) a 1,4 MHz s ochrannými pásmy (+2 . 160 kHz na každé straně pásma LTE Cat-M1) [10].



Obr. 1.8: Operační módy LTE Cat-M1, Stand alone a in-band

I přes snížení složitosti zařízení technologie Cat-M1 zachovává relativně nízké zpoždění přenosu dat (v řádu desítek ms) a podporu pro mobilní zařízení, včetně funkce hand-over. Díky vyšší přenosové rychlosti oproti ostatním technologiím CIoT je možné technologii Cat-M1 využít pro přenos hlasu s využitím VoLTE (Voice over LTE) [12, 11].

Rozšíření pokrytí LTE Cat-M1 CE (Coverage Enhancement)

Ve vydání 3GPP 13 byly definovány dva operační módy pro zařízení LTE Cat-M1 s cílem zajistit vyšší pokrytí rádiovým signálem oproti předchozím technologiím LTE nejméně o 15 dB [11, 13].

- **CE Mode A** – Je povinný pro všechny zařízení technologie LTE Cat-M1 a je uzpůsoben pro potřeby zařízení v běžných rádiových podmínkách
- **CE Mode B** – Je volitelný mód, který je vhodný pro zařízení pracující ve ztížených podmínkách z hlediska pokrytí rádiovým signálem [12].

Mód A umožňuje plné využití služeb podporovaných technologií Cat-M1. Mód B umožňuje zvýšení úrovně pokrytí rádiovým signálem, avšak za cenu zvýšení zpoždění přenášených dat a omezení rychlosti přenášení dat. Zařízení v módu B také nemohou využívat některé podporované služby, jako VoLTE a mobilita, což vyplívá z omezení přenosové rychlosti a zvýšení latence. Z tohoto důvodu je využití módu B vhodné u stacionárních zařízení, u kterého se nepředpokládá vysoký objem přenesených dat (např. senzory) [14].

Daný mód je volen základnovou stanicí eNodeB na základě vyhodnocení naměřené kvality rádiového signálu [13]. Zvýšení pokrytí je dosaženo počtem opakování zpráv. Maximální možný počet opakování zpráv se liší v závislosti na módu zařízení (mód A /B) a také na kanále, pro který je zapotřebí zprávu opakovat. Maximální možný počet repetic pro módy a kanály jsou uvedeny v tab. 2 [12].

Kanál LTE-M	Repetice v módu A	Repetice v módu B
PSS/SSS	1	1
PBCH	1	5
MPDCCH	16	256
PDSCH	32	2048
PUSCH	32	2048
PUCCH	8	32
PRACH	32	128

Tab. 1.1: Počty repetic pro jednotlivé kanály v módech LTE-M.

LTE Cat-M1 disponuje dle specifikace 3GPP úrovně pokrytí MCL až 160 dB (někde uvedeno 155 dB), což je o 16 dB více než u broadband technologií LTE. Rozložení tříd pokrytí ECL na základě hodnot MCL je závislé na operátorovi provozující danou síť [6].

Rádiové rozhraní LTE Cat-M1

Technologie LTE Cat-M1 podporuje jak plně duplexní přenos s využitím kmitočtového dělení FD FDD (Full Duplex – Frequency Division Duplexing) a časového dělení TDD (Time Division Duplexing), tak poloduplexní přenos dat s kmitočtovým dělením HD-FDD (Half Duplex-Frequency Division Duplexing), od čehož se odvíjí i maximální možná teoretická přenosová rychlost (1 Mb/s pro plně duplexní přenos a 375 kb/s pro poloduplexní přenos pro uplink i downlink) [6].

Cat-M1 zachovává původní modulační schéma z LTE. Pro downlink využívá OFDMA s modulacemi QPSK a 16QAM (16 Quadrature Amplitude Modulation – 4 bity v jednom stavu). Pro režim uplink je využita metoda mnohonásobného přístupu SC-FDMA s modulacemi QPSK a 16QAM [6].

LTE Cat-M2

Ve vydání 3GPP 14 (Release 14) byla definována nová kategorie koncového zařízení LTE Cat-M2. Jde o rozšíření stávající technologie LTE Cat-M1 o dosud nepodporované služby, či o vylepšení dosavadních služeb poskytovaných technologií LTE Cat-M1 [14].

Změny pro technologie LTE-M ve vydání 3GPP 14:

- Rozšíření dosavadních protokolů pro lokalizaci zařízení LPP (Location and Positioning Protocol), zejména metod pro lokalizaci OTDOA (Observed Time Difference of Arrival) a E-CID (Enhanced Cell Identity) [14, 10].

- Rozšíření šířky pásma na 5 MHz (využívá 25 PRB LTE), čímž je zvýšena maximální teoretická propustnost technologie (DL 4 MB/s a UL 7 MB/s v plně duplexním přenosu) [14, 10].
- Optimalizace a vylepšení služeb založených na přenosu dat v reálném čase (VoLTE) pro zařízení v CE módu A. Přidání podpory mobility i ve stavu RRC Idle a podpory multicastových komunikací. [14].

V navazujícím vydání 3GPP 15 (Release 15) byla definována nová výkonová třída zařízení pro LTE-M o výkonu 25,12 mW (14 dBm, Power Class 6), pro účely snížení spotřeby koncového zařízení [14].

Za účelem další optimalizace spotřeby zařízení, došlo ve vydání 15 k implementaci WUS (Wake-up Signal) [14, 10]. Funkčnost tohoto systému je totožná s implementací u technologie NB-IoT s rozdílem implementace na proprietárních kanálech technologie LTE-M (MPDCCH, PDSCH).

1.3 Technologie bezlicenčního pásma

Technologie bezlicenčního pásma pracují v nelicenčních pásmech ISM 169 MHz, 433 MHz, 868 MHz a 2,4 GHz. Výhodou technologií bezlicenčního pásma je bezplatné využití frekvenčního spektra. Vzhledem k hojnému počtu různých technologií komunikujících ve stejných kmitočtových spektrech je zapotřebí technologie bezlicenčních pásem omezovat na přenosových parametrech, aby nedocházelo k zarušení komunikačního kanálu, či ovlivňování ostatních zařízení. Techniky použité pro omezení zařízení jsou

- Omezení vysílacího výkonu
- Omezení střídy vysílání ToA (Time on Air).
- Omezení maximálního objemu přenášených dat v jedné zprávě.

Mezi nejaktuálnější technologie LPWA bezlicenčního pásma jsou technologie LoRaWAN a Sigfox.

1.3.1 Sigfox

Technologie Sigfox je technologie pracující v bezlicenčním frekvenčním pásmu 868 MHz v Evropě. Jedná se o proprietární technologii stejnojmenné společnosti. Technologie Sigfox byla poprvé představena roku 2009 a jedná se o vůbec první technologii specificky zaměřenou pro komunikaci zařízení IoT. Technologie Sigfox byla původně navržena pouze pro přenos zpráv ve směru uplink, avšak při pozdějších úpravách byl implementovaný i přenos ve směru downlink. Komunikační dosah technologie Sigfox je odhadován v rozsahu 10-30 km [6].

Obr. 1.9: Logo Sigfox

Rádiové rozhraní Sigfox

Technologie Sigfox pro komunikaci využívá frekvenční pásmo o šířce 192 kHz. Pro samotný přenos zpráv je však využito velice úzké šířky pásma 100 Hz, což má za následek nízké energetické nároky pro vyslání zprávy, zvýšení úrovně pokrytí a kvality signálu vůči šumu, za cenu nízké přenosové rychlosti 100 b/s. Pro mnohonásobný přístup využívá technologie Sigfox RFTDMA (Random Frequency and Time Division Multiple Access) založené na systému ALOHA, kde zařízení vysílají na zcela náhodných frekvencích a náhodných časech. Zařízení technologie Sigfox mohou vysílat kdykoliv bez předchozí synchronizace a vysílají každou zprávu 3x na náhodných frekvencích a náhodných časech. Tím je zajištěna odolnost vůči vnějšímu rušení. Zprávy přenesené technologií Sigfox nejsou žádným způsobem šifrovány [6].

Využitá modulace pro směr uplink je modulace DBPSK (Differential BPSK), pro downlink modulace GFSK (Gaussian Frequency Shift Keying) [6].

Omezení technologie Sigfox

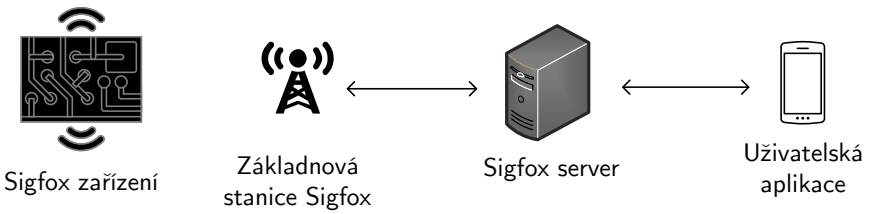
Díky nasazení technologie Sigfox v bezlicenčním frekvenčním spektru, je technologie Sigfox omezena na některých parametrech přenosu.

- **Omezení velikosti zprávy** – Velikost zpráva (bez záhlaví) je u technologie Sigfox omezena ve směru uplink na 12 B a ve směru downlink 8 B.
- **Omezení denního počtu zaslaných zpráv** – Počet zpráv odeslaných technologiích Sigfox je omezen na 140 zpráv ve směru uplink za den, ve směru downlink 4 zprávy za den.
- **Omezení využití kanálu -** Využití komunikačního je u technologie Sigfox omezeno střídou 1 %. V časovém úseku jedné hodiny je tedy možné vysílat nanejvýše 36 s.
- **Omezení maximálního vysílacího výkonu -** Vysílací výkon je u technologie Sigfox omezen na 14 dBm ve směru uplink a 27 dBm ve směru downlink.

Architektura Sigfox

Ačkoliv je technologie Sigfox provozována v nelicenčním frekvenčním spektru je její architektura spravována výhradně společností Sigfox nebo skrze její integrátory, kteří

dále distribuují technologii Sigfox po jednotlivých zemích a územních celcích. V České republice je Sigfox integrátorem společnost SimpleCell. Uživatelská data jsou přenesena technologií Sigfox na základnovou stanici, odkud jsou dále přeneseny konvenčními technologiemi na servery společnosti Sigfox. Veškerá uživatelská data jsou tak uložena na serverech společnosti Sigfox a přistupovat k nim lze pomocí webové aplikace společnosti Sigfox [15].



Obr. 1.10: Architektura sítě technologie Sigfox

1.3.2 LoRaWAN

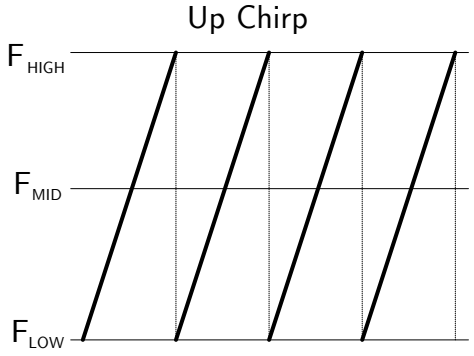
Standard LoRaWAN (LOng RAnge Wide Area Network), představený roku 2015, byl vyvinutý za účelem poskytování M2M připojení IoT zařízení, komunikujících na dlouhou vzdálenost. LoRaWAN je spravovaný uskupením LoRa Alliance a jedná se o specifikaci protokolů linkové vrstvy vybudované nad modulací fyzické vrstvy LoRa. V Evropě je technologie LoRaWAN provozována v kmitočtovém spektru 868 MHz. Komunikační dosah technologií LoRaWAN je odhadován v rozsahu 10-50 km [6].

Obr. 1.11: LoRaWAN logo

Rádiové rozhraní LoRaWAN

Technologie LoRaWAN využívá pro komunikaci frekvenčních pásem o šířce 125 kHz, 250 kHz a 500 kHz. V Evropě je využito téměř výhradně šířka pásma 125 kHz. LoRaWAN využívá modulace LoRa a dosahuje přenosových rychlosti 300 b/s až 50 kb/s v závislosti na využitých přenosových parametrech. Přenášené zprávy jsou u technologie LoRaWAN šifrovány symetrickou šifrou AES-128 [6].

Modulace LoRa využívá techniku rozprostření spektra s využitím modulace CSS (Chirp Spread Spectrum), vytvořenou pro zlepšení odolnosti proti rušícím vlivům a snížení potřebného vysílacího výkonu. Základním prvkem této modulace je jeden chirp, definovaný jako lineární změna frekvence mezi hranicemi šířky pásma. Jeden chirp se skládá z určitého počtu čipů, definovaných hodnotou SF (Spreading Factor), což ovlivňuje i dobu vysílání ToA (Time on Air) a tedy i spolehlivost přenosu. Každý čip reprezentuje specifickou hodnotu signálu v rámci modulace [6].



Obr. 1.12: Grafické zobrazení chirpů modulace LoRa

Třídy zařízení LoRaWAN

Technologie LoRaWAN definuje tři třídy koncových zařízení v závislosti na plánování zpráv ve směru downlink.

- **Třída A(ll)** – je základní implementací pro všechna koncová zařízení umožňující obousměrnou komunikaci. Každé koncové zařízení musí podporovat tuto třídu. Po zaslání zprávy ve směru uplink je možné odeslání zprávy ve směru downlink ve dvou krátkých časových slotech s rozestupem 1 s. Jedná se o nejúspornější třídu vhodnou pro bateriově napájená zařízení.
- **Třída B(eacon) -** umožňuje obousměrnou komunikaci. Po zaslání dat ve směru uplink jsou podobně jako u třídy A otevřeny dvě krátká časová okna pro příjem dat downlink. Navíc však dochází k otevření dalších oken v předem definovaných intervalech. Pro otevření oken v přesně definovaných časových intervalech odesílá základnová stanice (brána) synchronizační rámce.
- **Třída C(ontinuous)** – umožňuje obousměrnou komunikaci otevřením kontinuálního okna pro příjem zpráv ve směru downlink. Tato třída není vhodná pro bateriově napájená zařízení [6].

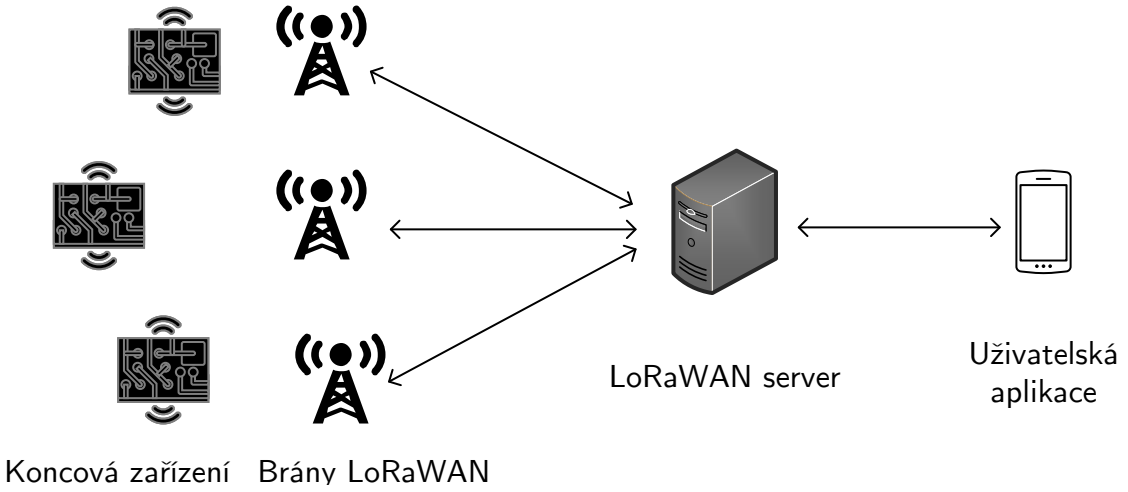
Omezení technologie LoRaWAN

Technologie LoRaWAN je díky nasazení v bezlicenčním frekvenčním spektru omezena na některých parametrech přenosu.

- **Omezeni využití kanálu -** Využití komunikačního je u technologie LoRaWAN omezeno střídou 0,1 %, 1 % a 10 % v závislosti na využitém kanálu a směru komunikace.
- **Omezení maximálního vysílacího výkonu** – Vysílací výkon je u technologie LoRaWAN omezen na 14 dBm ve směru uplink a 27 dBm ve směru downlink.

Struktura LoRaWAN

Struktura LoRaWAN je složena z koncových zařízení, základnových stanic (bran) a serveru. Ve směru uplink odesílá koncové zařízení zprávu, která je přijatá několika bránami. Na základě času přijetí zprávy je určena nejbližší brána (lokalizace zařízení), která je později využita pro přenos zpráv ve směru downlink. Brána přeposílá zprávu konvenčními přenosovými technologiemi na LoRaWAN server, kde je možné ke zprávě přistupovat. Na rozdíl od technologie Sigfox, technologie LoRaWAN umožňuje tvorbu privátních sítí [16].



Obr. 1.13: Architektura LoRaWAN

1.4 Srovnání LPWA technologií

Technologie	Sigfox	LoraWAN	LTE Cat-NB1	LTE Cat-NB2	LTE Cat-M1	LTE Cat-M2
Spektrum	ISM	ISM	Licencované	Licencované	Licencované	Licencované
Frekvence	868/915 MHz	433/868/915 MHz	700-2100 MHz	700-2100 MHz	700-2100 MHz	700-2100 MHz
Typ technologie	Proprietární	PHY: Proprietární MAC:Open Source	Open LTE	Open LTE	Open LTE	Open LTE
Šířka pásma	100,600 Hz	125, 250, 500 kHz	200 kHz	200 kHz	1,4 MHz	1,4 MHz
Vysílací výkon	UL 14, DL 27 dBm	14 dBm	23 dBm	23 dBm	23 dBm	23 dBm
Max velikost dat	UL 12 B, DL 8 B	242 B	1600 B	1600 B	1600 B	1600 B
Rychlost UL	0,6 kb/s	11 kb/s	62,5 kb/s	159 kb/s	1,3 Mb/s	7 MB/s
Rychlost DL	0,6 kb/s	21,9 kb/s	27,2 kb/s	127 kb/s	1 Mb/s	7 MB/s
Spotřeba	Tx: 44 mA Rx: 12 mA PSM: < 1 uA	Tx: 240 mA Rx: 46 mA PSM: < 1 uA	Tx: 240 mA Rx: 46 mA PSM: 3 uA	240 mA Rx: 46 mA PSM: 3 uA	Tx: 360 mA Rx: 70 mA PSM: 8 uA	Tx: 360 mA Rx: 70 mA PSM: 8 uA
Výdrž baterie	> 10 let	> 10 let	> 10 let	> 10 let	> 5 let	> 5 let
Zabezpečení	Žádné / AES-128	AES-128	LTE	LTE	LTE	LTE

Tab. 1.2: Srovnání přenosových technologií.

2 Protokoly transportní vrstvy v LPWA přenosech

Pro přenos dat mezi aplikacemi uživatelů jsou v transportní vrstvě modelů TCP/IP a OSI definovány protokoly zajišťující služby pro přenos dat. Protokoly transportní vrstvy zajišťují případnou režii a doručení dat k příslušnému aplikačnímu procesu. Mezi základní služby protokolů transportní vrstvy se řadí zajištění doručení zpráv z pohledu požadavků aplikace. V transportní vrstvě jsou definovány dva univerzální a nejběžněji využívané protokoly pro přenos dat aplikací, UDP (User Datagram Protocol) a TCP (Transmission Control Protocol). Tyto dva protokoly jsou nativně podporovány ve velké části komunikačních modulů technologií licenčního pásma CIoT. Kromě protokolů UDP a TCP se mimo jiné lze také setkat s protokoly DCCP (Datagram Congestion Control Protocol), SCTP (Stream Control Transmission Protocol), RSVP (Resource Reservation Protocol), které jsou aplikovány ve specifických scénářích a nejsou vhodné pro použití v systémech LPWA.

2.1 User Datagram Protocol

Protokol UDP, představený roku 1980, zprostředkovává přenos dat mezi koncovými účastníky na transportní vrstvě modelu OSI a TCP/IP [17]. Protokol UDP je často označován jako nespolehlivý protokol, jelikož na rozdíl od ostatních protokolů transportní vrstvy, nevyžaduje po příjemci odeslání potvrzovací zprávy o přijetí (best-effort). Pro přenos dat přes protokol UDP (datová jednotka UDP – datagram) není potřebné vytvoření spojení mezi odesílatelem a příjemcem. Díky nespojovanému a nepotvrzovanému nedochází k žádné režii, která by zásadně zvýšila objem přenesených dat přes komunikační médium a protokol UDP je tak vhodné použít pro aplikace běžící v reálném čase, které nevyžadují 100 % spolehlivost přenosu dat (například hlasové aplikace), nebo pro aplikace citlivé na objem přenesených dat (vybrané technologie LPWA). Vyžaduje-li aplikace režii, může využít jiný protokol transportní vrstvy, případně musí potřebnou režii zajišťovat protokol vyšší vrstvy modelu OSI a TCP/IP [17].

Díky absenci jakékoliv režie na transportní vrstvě, obsahuje záhlaví protokolu UDP pouze informace o zdrojovém a cílovém portu (2+2 B), délku dat přenášených v datagramu (2 B) a kontrolní součet (2 B). Záhlaví UDP tak navyšuje velikost přenášených dat o 8 B.

Bity 0-15	Bity 16-31
Zdrojový port	Cílový port
Celková délka	Kontrolní součet

Obr. 2.1: UDP datagram

2.2 Transmission Control Protocol

Protokol TCP, představený roku 1974, je v dnešní době nejvíce využívaným protokolem transportní vrstvy v IP sítích. TCP zajišťuje spojovaný a spolehlivý přenos aplikačních dat (datová jednotka TCP – segment), tedy přenos přes prvotně vytvořené spojení a s potvrzováním přijatých zpráv [17]. Proto je vhodné protokol TCP použít u aplikací, u kterých je vyžadována 100 % spolehlivost doručení dat. Díky rozsáhlé režii naopak není vhodný pro použití u aplikací běžící v reálném čase s nárokem na nízké zpoždění přenosu.

Záhlaví protokolu TCP je výrazně složitější oproti protokolu UDP. Kromě zdrojového a cílového portu zde lze nalézt pořadové číslo bajtu (Sequence Number), jehož kontrolou je adresát schopen detekovat duplicitu, ztrátu nebo příjem zpráv ve špatném pořadí. Pořadové číslo potvrzovaného bajtu indukuje odesílateli, že zpráva byla přijata [20]. Další důležitou položkou jsou příznakové bity, které slouží k řízení spojení. Celková délka záhlaví je minimálně 20 B, díky čemuž není vhodný pro použití v systémech citlivých na objem přenesených dat [17].

<table>
<tr><td colspan="8">Bity 0-15</td><td colspan="8">Bity 16-31</td></tr>
<tr><td colspan="8">Zdrojový port</td><td colspan="8">Cílový port</td></tr>
<tr><td colspan="16">Pořadové číslo odesílaného bajtu</td></tr>
<tr><td colspan="16">Pořadové číslo potvrzovaného bajtu</td></tr>
<tr><td colspan="2">Délka záhlaví</td><td colspan="3">Rezerva</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="8">Délka okna</td></tr>
<tr><td colspan="8">Kontrolní součet</td><td colspan="8">Ukazatel naléhavých dat</td></tr>
<tr><td colspan="16">Volitelné položky záhlaví</td></tr>
<tr><td colspan="16">Aplikační data</td></tr>
</table>

Obr. 2.2: Záhlaví TCP

3 Protokoly aplikační vrstvy v LPWA přenosech

Protokoly aplikační vrstvy jsou protokoly zajišťující formátování a obsah uživatelských dat pro přenos mezi systémy a je-li zapotřebí i případnou režii a potvrzování příjmu zpráv (protokoly využívající protokol UDP na transportní vrstvě) [17]. Pro potřeby technologií LPWA, kde se uvažuje připojení menších, mobilních zařízení s omezeným výpočetním výkonem, je vhodné volit takové protokoly, aby zajišťovali spolehlivý přenos uživatelských dat, ale zároveň nekladly vysoké nároky na výpočetní kapacitu zařízení (co nejvíce odlehčené – lightweight – protokoly). Mezi běžně používané protokoly aplikační vrstvy u technologií LPWA se řadí MQTT (Message Queuing Telemetry Transport), či jeho varianta pro senzorové sítě MQTT-SN (Message Queuing Telemetry Transport - Sensor Networks), protokoly CoAP (Constrained Application Protocol) a LWM2M (Lightweight Machine to Machine).

3.1 Message Queuing Telemetry Transport

Protokol MQTT je odlehčený, textově založený (text based) protokol přestavený roku 1999 a standardizovaný roku 2013 organizací OASIS (Organization for the Advancement of Structured Information Standards). Protokol MQTT využívá protokol TCP na transportní vrstvě, čímž je zajištěna spolehlivost přenosu dat. Princip fungování protokolu MQTT je založen na topicích (tématech) do kterých mohou jednotlivá zařízení odesílat data (PUBLISH) nebo data po přihlášení se k odběru topiku (SUBSCRIBE) vyčítat (metoda Publish-Subscribe).

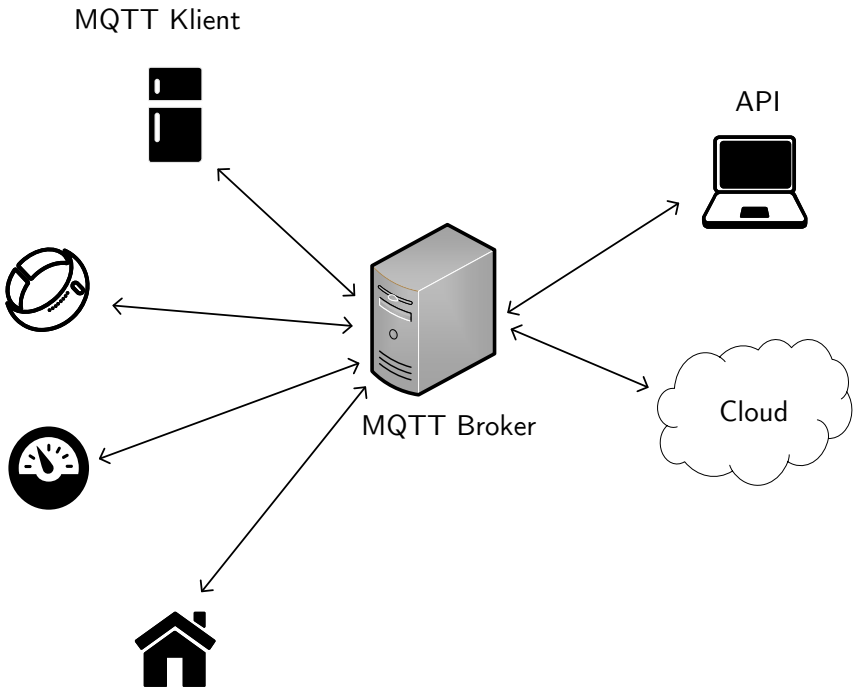
3.1.1 Struktura komunikačního řetězce MQTT

MQTT definuje tři základní entity v komunikačním řetězci.

- **MQTT Broker (server) -** Broker je centrálním bodem infrastruktury MQTT, se kterým komunikují jednotlivá zařízení (klienti). V každé infrastruktuře protokolu MQTT musí existovat minimálně jeden MQTT Broker. Broker je realizován uživatelskou aplikací (Mosquitto aj.). MQTT Broker je zodpovědný za příjem zpráv od publikujících zařízení (publishers) a následné přeposlání zprávy uživatelům, kteří daný topik odebírají (subscribers). Broker také zprostředkovává propojení sítě MQTT s externími prvky, například uživatelská API (Application Programming Interface), externí cloud apod.
- **MQTT Publisher -** Publisher je koncové zařízení (klient) odesílající data na MQTT Broker. Jedno zařízení může publikovat data do několika topiků, v

jedné zprávě však pouze do jednoho.

- **MQTT Subscriber -** Koncové zařízení (klient) přihlášené k odběru topiku. V případě publikace nové zprávy do topiku, ke kterému je klient přihlášen, je mu zpráva přeposlána metodou PUBLISH ze strany MQTT Brokeru.



Obr. 3.1: Struktura komunikačního řetězce MQTT

3.1.2 Funkcionalita MQTT

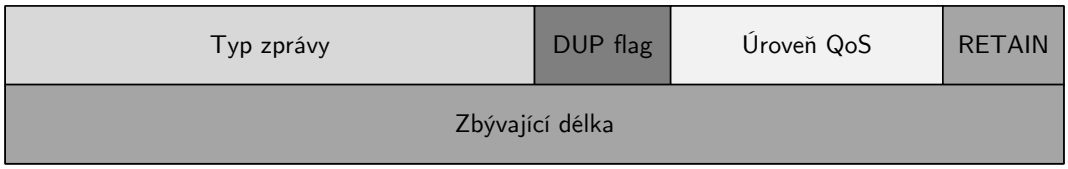
Pro organizaci datových toků využívá protokol MQTT rozdělení do tzv. topiků. Topik je řetězec znaků kódovaných v UTF-8 (Unicode Transformation Format), unikátních pro daný topik [18]. Struktura topiků připomíná složkový systém operačních systému a lze větvit do více úrovní za pomocí oddělovače „/“. Minimální počet znaků v jednotlivých úrovní je jeden znak [18]. Příklady topiků:

VUT/FEKT/T12/SC/5/Unilab/teplota
VUT/FEKT/T12/SC/5/Unilab/svetlo1
VUT/FEKT/T12/SC/5/Unilab/svetlo2
VUT/FEKT/T8/010/teplota

Do topiku mohou koncová zařízení přispívat (publish) nebo z nich odebírat (subscribe). Subscriber přihlášený do topiku dostává pouze zprávy v něm publikované,

čímž je zajištěno, že daný subscriber nepřijímá další zprávy, o které nemá zájem a které publikují ostatní klienti v ostatních topicích. Informace o topicích schraňuje MQTT Broker. V případě, že na Broker příjde zpráva PUBLISH nebo SUBSCRIBE na neexistující topik, je daný topik Brokerem automaticky vytvořen [19].

Uživatelská data jsou zapouzdřeny v záhlaví protokolu MQTT, které se skládá z minimálně dvou povinných B (fixní hlavička - fixed header). Fixní hlavička obsahuje důležité informace o typu zprávy, délce zprávy a požadované kvalitě služby QoS [18].



Obr. 3.2: Fixní záhlaví protokolu MQTT

- Typ zprávy je reprezentován prvními čtyřmi bity zprávy MQTT. Typ zprávy určuje, o jakou zprávu se jedná a také určuje strukturu záhlaví. MQTT definuje celkově 15 typů zpráv z nichž nejvýznamnějšími jsou zprávy CONNECT, CONNACK, PUBLISH, SUBSCRIBE a DISCONNECT.
- Příznakový bit duplikované zprávy DUP (Duplicate) - bit č. 3. Umožňuje indikaci opětovně odeslané zprávy.
- Úroveň QoS (bity 1 a 2). MQTT definuje 3 úrovně :
  - 0 (00) – Nejvíce jednou (At most once) – zpráva je odeslána právě jednou bez ohledu na její doručení.
  - 1 (01) – Alespoň jednou (At least once) – je vyžadováno potvrzení přijetí zprávy. Pokud nepřijde potvrzení, je zpráva vyslána několikrát, což může vést k několikanásobnému přijetí zprávy.
  - 2 (10) – Přesně jednou (Exactly once) – je vyžadováno potvrzení přijetí zprávy. Na základě ID zprávy (Message ID) je zpráva přijata pouze jednou.
- Retain (bit 0) – Je-li při metodě publish nastavený tento bit, je poslední zpráva v topiku uložena na MQTT Brokeru. V případě připojení nového subscribera je mu zpráva ihned po připojení odeslána. Subscriber tak nemusí čekat na novou publikaci.
- Poslední položkou povinného záhlaví je zbývající délka zprávy v bajtech, včetně variabilního záhlaví. Pole zbývající délky může být až 4 bajty dlouhé, což je určeno nejvíce významnými bity. Pokud je nejvíce významný bit nastavený na logickou hodnotu 1, následuje další bajt pole zbývající délky. Velikost zprávy je pak určena ze zbývajících sedmi bitů. Bajty jsou řazeny od nejméně vý-

znamného bajtu a teoreticky je možné přenést zprávu až o velikosti 256 MB [18]. Například hodnoty bajtů 0x80 a 0x32 reprezentují velikost zprávy 6400 B a hodnota 0x7F zprávu o velikosti 127 B. Rozdělení zprávy pro přenos řeší protokol nižší vrstvy (TCP segmentace).

Po fixní hlavičce následuje variabilní záhlaví (variable header), jehož délka a zda je vůbec přítomno závisí na typu zprávy. Data obsažená ve variabilní hlavičce se liší v závislosti na typu zprávy, požadovaném zabezpečení a požadované QoS. Položkami variabilního záhlaví může být identifikační číslo zprávy (u potvrzovaných zpráv) nebo název protokolu, verze protokolu a příznaková pole určující další strukturu datové části (payloadu) u zpráv CONNECT. U zpráv PUBLISH je součásti variabilního záhlaví také název topiku, do kterého chce uživatel publikovat [18].

Následuje datová část (payload), kde jsou v závislosti na typu zprávy uvedeny samotná uživatelská data (zprávy PUBLISH), doplňující informace jako je uživatelské identifikační číslo (Client ID – povinné pole), jméno a heslo (zpráva CONNECT), popřípadě topik, ke kterému se chce klient připojit (zprávy SUBSCRIBE) [18].

3.1.3 Popis komunikace MQTT

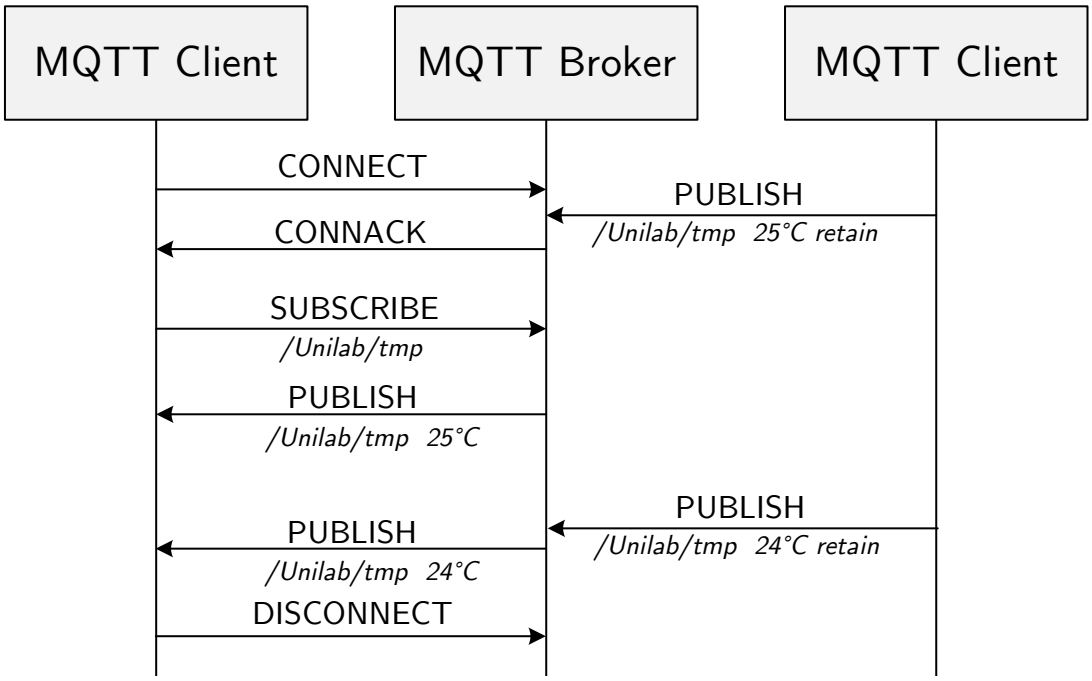
Před samotným odesíláním dat se musí klient připojit k MQTT Brokeru. Pro navázání komunikace s MQTT Brokerem musí klient odeslat zprávu CONNECT, kde je uvedený protokol a jeho verze, kontrolní bity, povinné pole Client ID, a je-li vyžadováno i klientské přihlašovací jméno a heslo pro přihlášení k Brokeru. Zpráva CONNECT je Brokerem zpracována a nazpět je klientovi odeslána zpráva CONNACK, která obsahuje informaci o výsledku pokusu o připojení (přijato/nepřijato a důvod selhání). Po úspěšném připojení může klient publikovat zprávy nebo se přihlásit k odběru jednotlivých topiků [18].

Klient, který se chce přihlásit k odběru topiku odesílá Brokeru zprávu SUBSCRIBE obsahující ID zprávy, informace o topiku/topicích a také o požadované QoS pro jednotlivé topiky. V jedné zprávě SUBSCRIBE se lze přihlásit k odběru několika topiků. Po zpracování zprávy odesílá Broker zprávu SUBACK, ve které informuje klienta o přidělených QoS při jednotlivé topiky. Pokud daný topik na Brokeru neexistuje, je automaticky vytvořen. Chce-li klient zrušit odběr topiku, odesílá zprávu UNSUBSCRIBE s ID zprávy a informacemi o topiku/topicích, ze kterých se chce odhlásit. Jako odpověď je mu nazpět odeslána zpráva UNSUBACK [18].

Pro publikování zprávy odesílá klient zprávu PUBLISH s informací o topiku, do kterého chce publikovat a daty, která chce publikovat. V závislosti na nastaveném QoS je přijetí zprávy potvrzeno zprávou PUBACK, či nikoliv. Po přijetí zprávy je provedena Brokerem kontrola existence topiku. Pokud daný topik neexistuje, je Brokerem automaticky vytvořen. Pokud již existuje, provede Broker kontrolu zaří-

zení, která daný topik odebírají. Těmto zařízením je následně publikovaná zpráva odeslána ve zprávě PUBLISH od Brokeru [18].

Pro odpojení klienta z Brokeru je odeslána zpráva DISCONNECT, která není nijak potvrzována.



Obr. 3.3: Příklad průběhu komunikace MQTT mezi Brokerem a klienty A a B s QoS 0.

3.1.4 Zabezpečení MQTT

Protokol MQTT nevynucuje pro svoji funkčnost žádné zabezpečení, ale podporuje několik běžně využívaných metod pro autentizaci klientů a zabezpečení přenosu dat. Autentizace klientů:

- **Na základě znalosti -** Pro úspěšné připojení k MQTT Brokeru musí klient prokázat znalost klientského ID (Client ID) a volitelně uživatelského jména a hesla ve zprávě CONNECT. Na základě Client ID je pak možné omezovat klienta v přístupu do jednotlivých topiků pro zvýšení bezpečnosti [20].
- **Na základě certifikátu** – Při připojování k MQTT Brokeru je vyžadován certifikát vydaný Brokerem.

Šifrování komunikace:

- **Šifrování dat na úrovni uživatelské aplikace -** Uživatel může zašifrovat svá data před předáním dat protokolu MQTT. Předpokladem je, že klienti přihlášení k odběru daného topiku znají klíč pro dešifrování dat. Tímto způsobem

se dají zašifrovat pouze uživatelská data (payload). Ostatní části protokolu MQTT tak zůstanou nezašifrované, včetně zaslaného hesla při připojování.

- **Šifrování pomocí TLS (Transport Layer Security)** – Data jsou šifrovány na 6. vrstvě modelu OSI. Jedná se o nejbezpečnější variantu zabezpečení, jelikož protokol TLS šifruje celou zprávu MQTT, nikoliv pouze zasílaná data. Případný útočník tak nemůže odposlechnout jakékoliv informace.

Pro nezašifrovanou komunikaci využívá protokol MQTT port 1883. Pro zašifrovanou komunikaci protokolem TLS využívá 8883 a port 8884, kde je kromě protokolu TLS vyžadována i autentizace certifikátem typu X.509 [20].

3.2 MQTT for Sensor Networks

MQTT for Sensor Networks (MQTT-SN) je protokol představený v roce 2008 společností IBM a od roku 2013 standardizovaný společností OASIS [21]. Protokol MQTT-SN vychází z protokolu MQTT upraveného pro potřeby bezdrátových senzorových jednotek Internetu věcí (IoT) - M2M komunikace. MQTT-SN zachovává základní principy komunikace MQTT (publish-subscribe) s ohledem na potřeby a limitace bateriově napájených zařízení (umožnění přerušovaného spojení, omezení objemu přenášených dat, předpoklad nízké výpočetní kapacity zařízení) [21]. Na rozdíl od protokolu MQTT, MQTT-SN využívá nespojované protokoly transportní vrstvy (např. UDP nebo ZigBee) [21].

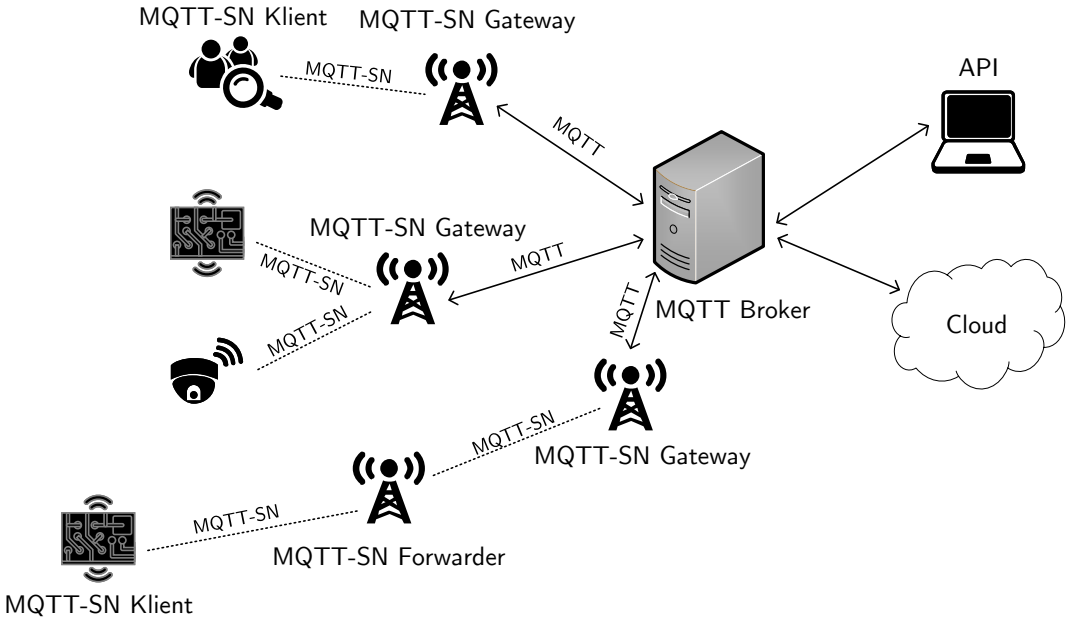
3.2.1 Struktura komunikačního řetězce MQTT-SN

Protokol MQTT-SN definuje oproti protokolu MQTT tři základní entity komunikačního řetězce.

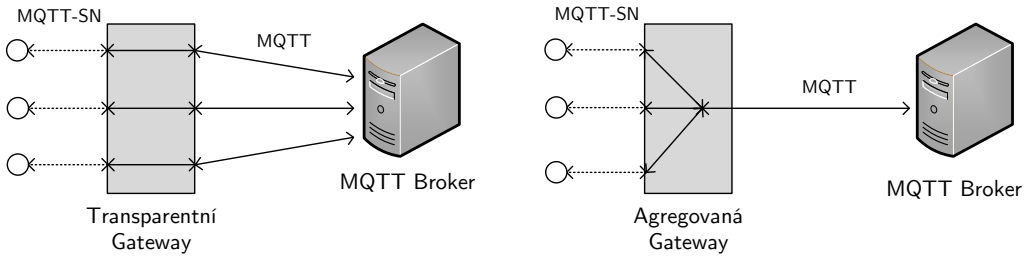
- **MQTT-SN Klient -** Koncové zařízení. Připojují se do sítě MQTT skrze MQTT-SN Gateway.
- **MQTT-SN Gateway (GW) -** Entita řídící komunikaci mezi koncovými účastníky. Gateway může být přímo zabudována nebo propojena s MQTT Brokerem. Pro komunikaci s Brokerem je využito protokolu MQTT. Gateway zprostředkovává „překlad“ mezi protokoly MQTT-SN a MQTT. Z hlediska topologie a funkcionality lze Gatewaye rozdělit na dva druhy:
  - **Transparentní Gateway -** Pro každého připojeného MQTT-SN klienta je vytvořeno samostatné MQTT spojení k MQTT Brokeru, čímž je docíleno téměř transparentní komunikace mezi klientem a MQTT Brokerem (end-to-end).

– **Agregovaná Gateway -** Mezi Gatewayí a Brokerem je vytvořeno pouze jedno MQTT spojení. Veškerá spojení klientů končí na GW, která rozhoduje, které zprávy budou přeposlány na MQTT Broker.

- **MQTT-SN Forwarder -** V případě, že v dané síti není přímo připojena žádná Gateway nebo je pro klienta nedostupná, je zpráva přijata Forwarderem. Forwarder zprávu zabalí do svého záhlaví a přepošle nezměněnou na nejbližší Gateway. V opačném směru komunikace příjme data od Gatewaye, odstraní záhlaví a nepozměněná data odešle klientovi [21].



Obr. 3.4: Architektura sítě MQTT-SN



Obr. 3.5: Transparentní vs agregovaná GW

3.2.2 Funkcionalita MQTT-SN

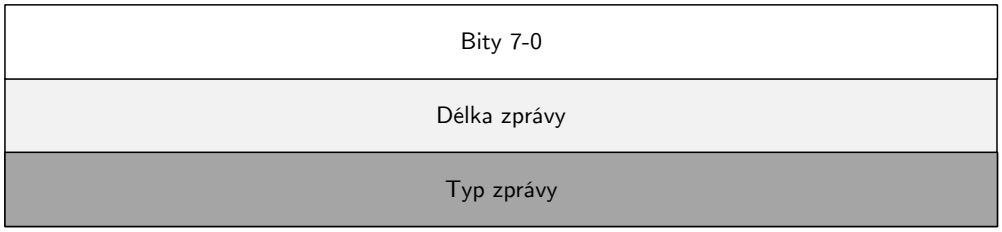
Protokol MQTT-SN zachovává původní funkcionalitu protokolu MQTT s drobnými odlišnostmi s ohledem na stavbu připojených zařízení. Změny oproti MQTT jsou implementovány s ohledem na nízkou předpokládanou výpočetní kapacitu klientů a vzhledem k charakteru komunikace (bezdrátová).

Pro snížení objemu přenášených dat se řetězce znaků pro identifikaci topiků nahrazují 16bitovým identifikátorem, unikátním pro daný topik (Topic ID / Topic Alias). Topic Alias vždy generuje GW a může být dynamicky generovaný nebo předdefinovaný. Topic Alias může klient získat zprávami REGISTER a SUBSCRIBE [21].

MQTT-SN implementuje mechanismy pro vyhledávání GW (Gateway Discovery). Každá GW periodicky odesílá multicast zprávy ADVERTISE obsahující informace o dané GW, které může klient následně využít. Pokud klient nepřijal zprávu ADVERTISE, může odeslat multicastovou zprávu SEARCHGW. GW, která tuto zprávu příjme odešle nazpět zprávu GWINFO obsahující informace o GW [21].

Pro potřeby malých senzorových jednotek byla definována nová třída QoS, QoS -1 (taktéž nazývána QoS 3), která umožňuje klientovi publikovat bez předchozího ustanovení spojení. V MQTT-SN může klient obdržet zprávu PUBACK i při nastavené QoS 0, pokud došlo k chybě [21].

Záhlaví protokolu MQTT-SN, stejně jako u MQTT, je rozděleno na fixní a variabilní. Fixní záhlaví je minimálně 2 B dlouhé, v závislosti na délce zprávy pak může být až 4 B dlouhé. Obsahuje informace o celkové délce zprávy a o typu zprávy. Typ zprávy určuje strukturu variabilního záhlaví [21].



Obr. 3.6: Záhlaví protokolu MQTT-SN

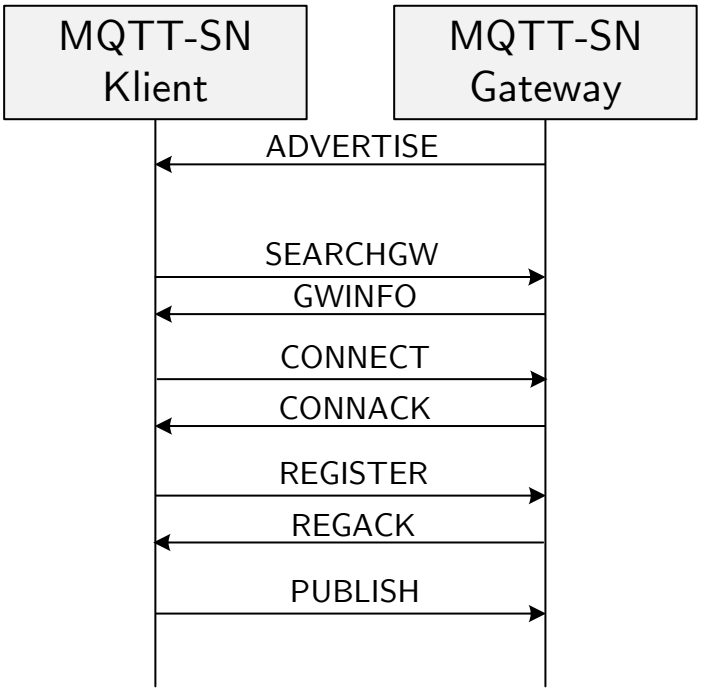
3.2.3 Popis komunikace MQTT-SN

V první fázi komunikace musí dojít k ustanovení spojení mezi klientem a GW (Gateway Discovery). Pokud klient nepřijal zprávu ADVERTISE, odesílá zprávu SEARCHGW. Na tuto zprávu odpovídá GW zprávou GWINFO. Po vyhledání GW následuje samotné připojení. Stejně jako u MQTT klient odesílá zprávu CONNECT se

všemi relevantními informacemi. Navíc uvádí i informaci o maximální délce zprávy, kterou je ochoten akceptovat. Po přijetí zprávy GW odesílá zprávu CONNACK s informacemi o daném procesu (úspěch / neúspěch a důvod). Klient následně může publikovat, či odebírat topiky.

Při prvním publikování zprávy musí klient odeslat zprávu REGISTER ve které uvede topik do kterého chce odeslat data – musí se „registrovat“. GW odpovídá zprávou REGACK ve které uvede 16bitový Topic Alias pro daný topik. Tyto zprávy lze vynechat zná-li klient Topic Alias dopředu. Se znalostí Topic Aliasu může klient odeslat zprávu PUBLISH.

Přihlášení k odběru topiku zprávou SUBSCRIBE je možné jak na základě znalosti Topic Aliasu, tak i na uvedení celého znakového řetězce identifikujícího daný topik. Stejně jako u MQTT je zpráva SUBSCRIBE potvrzována zprávou SUBACK, která obsahuje Topic Alias daného topiku, pro další komunikaci. Klient tak při další komunikaci nemusí odesílat celý řetězec topiku, čímž dochází ke snížení objemu přenesených dat.



Obr. 3.7: Ukázka komunikace pro publishování zprávy.

3.2.4 Zabezpečení MQTT-SN

Protokol MQTT-SN implementuje základní zabezpečení protokolu MQTT. Autentizace uživatelů probíhá stejně jako u protokolu MQTT na základě znalosti, či uži-

vatelského certifikátu.

Jelikož protokol MQTT-SN nevyužívá na transportní vrstvě protokol TCP není možné provoz zabezpečit protokolem TLS. Pro protokoly pracující s protokolem UDP na transportní vrstvě byla vyvinuta alternativa DTLS (Datagram Transport Layer Security), která nevyžaduje spojovaný přenos. DTLS je založeno na protokolu TLS a umožňuje autentizaci a šifrování přenášených dat.

Z pohledu šifrování je díky charakteru zařízení připojených do sítí MQTT-SN vhodné volit šifrovací algoritmy nenáročné na výpočetní výkon (AES, ChaCh20) společně s lightweight autentizačními protokoly (Poly1305) [22].

3.3 Constrained Application Protocol

Constrained Application Protocol (CoAP) je odlehčený webový protokol aplikační vrstvy modelu TCP/IP představený roku 2014 ve vydání RFC7252 organizací IETF (Internet Engineering Task Force). Protokol CoAP je navržený speciálně pro „constrained“ zařízení, tedy zařízení omezené na výpočetním výkonu, velikosti pamětí ROM, RAM a Flash, omezené na datových objemech nebo zařízení s omezenými možnostmi z hlediska napájení (bateriově napájená zařízení). Protokol CoAP je založený na modelu žádost-odpověď (request-response) a podporuje základní funkce webových služeb. CoAP je v mnoha ohledech podobný webovému protokolu HTTP (Hypertext Transfer Protocol), což byl záměr autorů pro snadný překlad a interakci mezi těmito protokoly (například přes proxy server). Po vzoru HTTP i protokol CoAP implementuje architekturu REST (Representational state transfer)[1], která je optimalizovaná pro potřeby M2M komunikace. Pro minimalizaci objemu přenesených dat je protokol CoAP vystavěn na protokolu UDP na transportní vrstvě, čímž je zajištěn nespojovaný přenos [23].

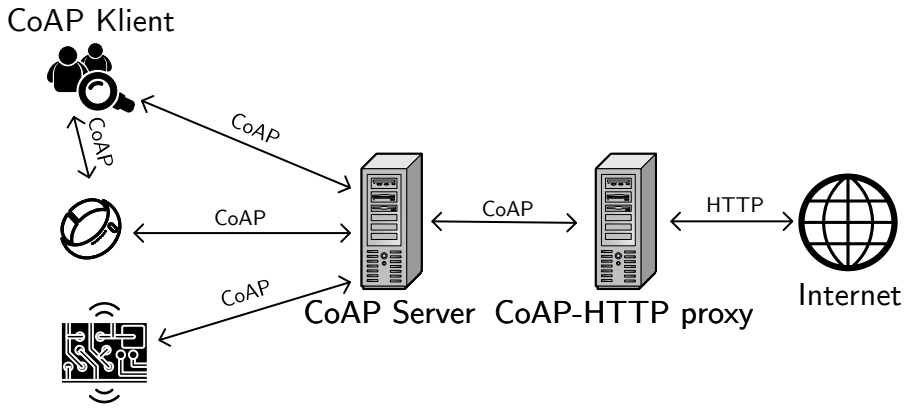
3.3.1 Struktura komunikačního řetězce CoAP

Protokol CoAP obecně definuje dvě entity v komunikačním řetězci:

- **CoAP klient -** Klientské zařízení implementující protokol CoAP. Zařízení generující požadavky.
- **CoAP server -** Zařízení zprostředkovávající přístup k datům. Generuje odpovědi na žádosti klientů a generuje žádosti o sledování dat (observe).

Vzhledem k charakteru protokolu CoAP je možná přímá komunikace mezi klienty (peer-to-peer), kdy jeden z klientů působí jako server a druhý jako klient. Tohoto scénáře je často využíváno v M2M komunikacích [23].

[1]Metodika přístupu k datům založená na řetězcích URI (Uniform Resource Identifier)



Obr. 3.8: Struktura komunikace protokolu CoAP

3.3.2 Funkcionalita protokolu CoAP

Protokol CoAP často pracuje se zdroji (resource). Zdrojem se rozumí cíl žádosti protokolu CoAP (data). Zdroje jsou definovány pomocí identifikátoru URI, který je přiložený ve zprávě CoAP. Příklad tvaru URI :

VUT/FEKT/T12/SC/5/Unilab/teplota

CoAP je založený na modelu žádost odpověď. Žádosti a odpovědi jsou rozlišovány na základě kódového označení ve tvaru „x.yy“, kde „x“ zastupuje třídu dané zprávy a „yy“ detail. CoAP, po vzoru protokolu HTTP, implementuje základní metody pro přístup a správu dat. Pro tyto metody je rezervovaná třída 0 [23].

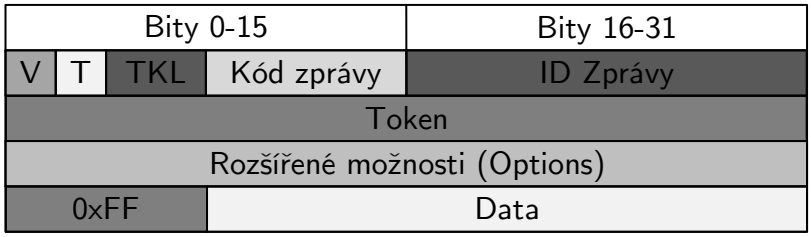
- **2.yy - Úspěch**
  - **0.yy - Žádost**
    - 0.01 - GET - Čtení zdroje
    - 0.02 - POST - Zápis/spuštění zdroje
    - 0.03 - PUT - Zápis zdroje
    - 0.04 - DELETE - Smazání zdroje/obsahu
  - 2.01 - Created - Vytvořeno
  - 2.02 - Deleted - Smazáno
  - 2.04 - Changed - Změněno
  - 2.05 - Content - Obsah přiložený ve zprávě
- **4.yy - Chyba na straně klienta**
  - 4.00 - Bad Request - Špatný požadavek ze strany klienta
  - 4.03 - Forbidden - Zakázáno
  - 4.04 - Not Found - Zdroj nenalezen
- **5.yy - Chyba na straně serveru**
  - 5.00 - Internal Server Error - Vnitřní chyba na serveru

- 5.01 - Not Implemented - Požadavek na serverem nepodporovanou službu
- 5.03 - Service Unavailable - Služba je aktuálně nedostupná

Jelikož je protokol CoAP založený na protokolu UDP, implementuje mechanismy pro potvrzování zpráv. Protokolem CoAP je možné odeslat 2 typy zpráv:

- **Potvrzovaná zpráva** - Confirmable (**CON**) - Po přijetí této zprávy je vyžadována odpověď potvrzující přijetí zprávy (ACK).
- **Nepotvrzovaná zpráva** - Non-confirmable (**NON**) - Tato zpráva je odeslána příjemci, přičemž potvrzování není vyžadováno. Pokud je přijata nepotvrzovaná žádost, odpověď by měla být odeslána také v nepotvrzované zprávě (nemusí být pravidlem) [23].

Záhlaví protokolu CoAP se skládá z povinných 4 B, které jsou přítomny v každé zprávě CoAP, obsahující základní informace o protokolu a o typu zprávy.



Obr. 3.9: Záhlaví protokolu CoAP

První dva bity záhlaví obsahují informaci o verzi protokolu. Bity musejí být nastaveny na hodnotu 01_2, jinak je daná zpráva neplatná. Třetí a čtvrtý bit jsou určeny pro identifikaci typu zprávy.

- 00_2 - Potvrzovaná zpráva CON
- 01_2 - Nepotvrzovaná zpráva NON
- 10_2 - Potvrzení zprávy ACK
- 11_2 - Reset

Poslední čtyři bity prvního bajtu jsou rezervovány pro délku tokenu. Může reprezentovat hodnoty 0-8 bajtů.

Druhý bajt obsahuje kód zprávy, tedy informace o žádosti, úspěchu, či chybě. Tato část je navíc rozdělena do dvou částí, kde první tři bity reprezentují třídu a zbylé detail daného kódu zprávy (2.05 je reprezentována jako hodnota 69).

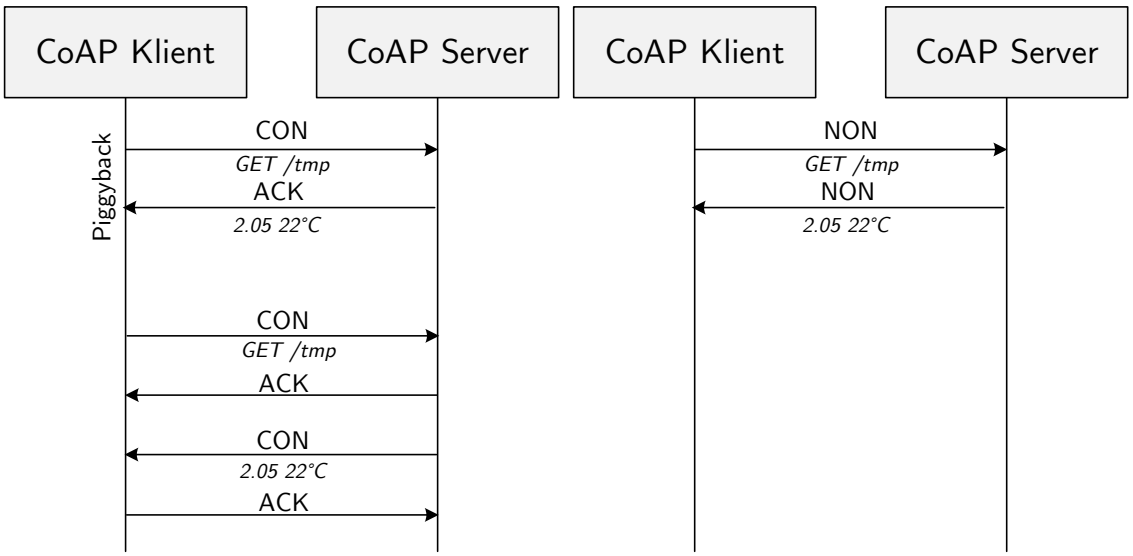
Zbylé 2 bajty povinného záhlaví reprezentují ID zprávy (Message ID), což je náhodné číslo generované odesílatelem požadavku. ID zprávy slouží pro identifikaci potvrzovaných zpráv (odpověď ACK musí mít stejné ID zprávy).

Nepovinnou položkou je pole Token. Pole Tokenu může být až 8 bajtů dlouhé. Token slouží pro identifikaci toku zpráv. Zprávy, které vyžadují odeslání několika zpráv musí mít stejný Token.

Rozšířené možnosti (options) slouží pro specifikaci nastavení. Mezi hlavní položky pole options patří specifikace URI zdroje, popřípadě specifikace datového formátu přiložených dat. Poslední položkou jsou uživatelská data [23].

3.3.3 Popis komunikace CoAP

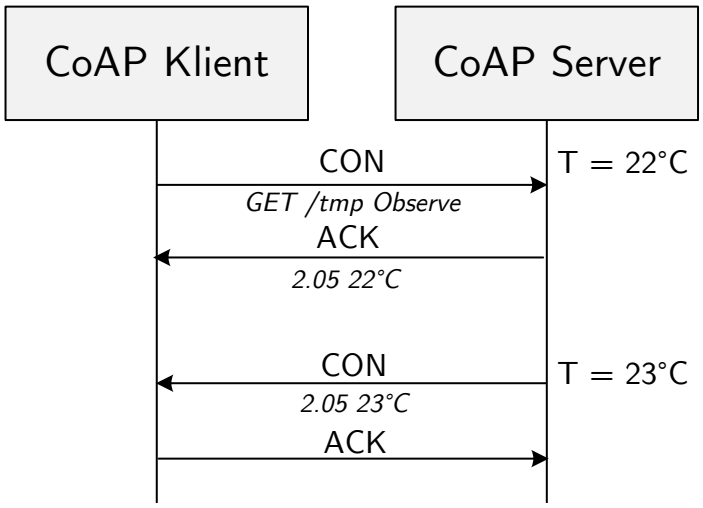
Komunikace protokolu CoAP je založena na modelu request-response. Klient zašle serveru požadavek, na který mu následně přijde od serveru odpověď. V závislosti na nastavení serveru a složitosti žádosti, může u potvrzovaných zpráv docházet k tzv. piggy-back odpovědi, což znamená, že odpověď na požadavek je obsažen přímo ve zprávě ACK odeslané nazpět klientovi. V opačném případě je odesláno pouze potvrzení a po zpracování požadavku je odeslána nová zpráva se stejnou hodnotou tokenu [23].



Obr. 3.10: Komunikace CoAP - potvrzovaná a nepotvrzovaná zpráva

V úpravě protokolu CoAP RFC7641 byla přidána podpora pro sledování zdrojů (observe). Klient může zažádat o sledování zdroje metodou GET, kde v rozšířených možnostech uvede požadavek na sledování zdroje. Při každé změně sledované hodnoty je klientovi zaslána zpráva s aktuální hodnotou sledovaného zdroje. V případě odstranění zdroje je klientovi odeslána zpráva korespondující s danou událostí (např. 4.04 Not found) [24].

Pro komunikaci využívá protokol CoAP na transportní vrstvě port 5683.



Obr. 3.11: Metoda observe

3.3.4 Zabezpečení protokolu CoAP

Protokol CoAP sám o sobě neimplementuje žádné mechanismy pro zabezpečení dat nebo autentizaci uživatelů. Zabezpečení je tedy nutné řešit jinými způsoby.

Zabezpečení a autentičnost uživatelských dat je možné řešit protokolem DTLS, který je kompatibilní s transportním protokolem UDP a který umožňuje autentizaci účastníků komunikace za pomocí certifikátů a šifrování komunikace [23]. Protokol CoAP zabezpečený protokolem DTLS je možné najít pod označením CoAPS (CoAP Secure – CoAP over DTLS).

Další alternativou je využití bezpečnostního protokolu OSCORE (Object Security for Constrained RESTful Environments). Protokol OSCORE, specifikovaný v RFC 8613, je navržený pro zabezpečení komunikace mezi koncovými účastníky využívajícími architekturu REST (CoAP, HTTP), primárně pro zařízení s omezenými možnostmi komunikace (nízká výpočetní náročnost, nízká spotřeba). Přidaná řídící pole do zprávy se mohou blížit hodnotám až 11-13 bajtů, což činí protokol OSCORE vhodným pro aplikaci na trhu IoT zařízení. OSCORE zabezpečuje nejen uživatelská data, ale i kritická pole pro signalizaci protokolu CoAP, včetně použité přístupové metody. Protokol OSCORE využívá pro šifrování algoritmy AEAD (Authenticated Encryption with Associated Data), což jsou symetrické šifrovací algoritmy schopné data nejen zabezpečit, ale zajistit autentizaci daných zpráv na základě hashovacích funkcí. Díky těmto algoritmům je protokol OSCORE schopný poskytnout nejen ochranu uživatelských dat, ale i ochranu před různými útoky (např. útok opakováním) [25, 26].

Protokol OSCORE je možné kombinovat s protokolem DTLS pro ještě vyšší zabezpečení přenosu dat, ovšem za cenu zvýšení objemu přenesených dat souvisejících

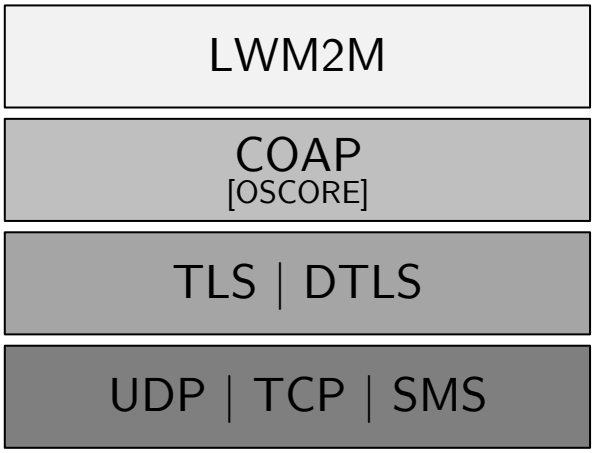
s nutnou režií [26].

3.4 Lightweight Machine to Machine

Protokol Lightweight Machine to Machine (LWM2M) je protokol představený roku 2017 normalizační organizací OMA (Open Mobile Alliance). Protokol LWM2M byl vytvořen za účelem sjednocení M2M komunikačních scénářů IoT zařízení a umožnění komunikace mezi zařízeními různých výrobců, kteří pro komunikaci dosud využívali své proprietární protokoly. Protokol LWM2M umožňuje vzdálenou správu zařízení (aktualizace firmwaru, diagnostika, správa konektivity, aktualizace přístupových údajů) a zprostředkování služeb (výčet senzorů, konfigurace) pro IoT zařízení. Protokol LWM2M je vystavěný na protokolu CoAP, který zajišťuje základní přístupové metody (GET, POST, PUT, DELETE) a zajišťuje režii přenosu. V komunikačním řetězci využívá protokol LWM2M dvě entity, LWM2M klient, zodpovědný za shromažďování dat a LWM2M server zodpovědný za správu zařízení a vyčítání a zápis dat. Funkcionalita protokolu LWM2M je založena na tzv. objektech, což jsou entity s pevně definovanou strukturou, které umožňují výčet a zápis dat, ale také mohou plnit funkci spouštění procesu, či nastavení. K objektům protokolu LWM2M je přistupováno skrze definované (well-defined) řetězce URI. Zabezpečení komunikace protokolu LWM2M může být zajištěno protokolem DTLS nebo v novější verzi LWM2M 1.1 bezpečnostním protokolem OSCORE [27, 28].

4 Lightweight M2M

Protokol LWM2M je aplikační protokol pro správu IoT zařízení přestavený roku 2017 ve verzi 1.0. Aktuální verze protokolu LWM2M 1.2 byla vydána organizací OMA v roce 2020. Protokol LWM2M sjednocuje strukturu datových modelů a způsob komunikace mezi IoT zařízeními různých výrobců. Protokol LWM2M je vystavěný na protokolu CoAP, kdy využívá přístupových metod protokolu CoAP pro správu tzv. objektů [27].

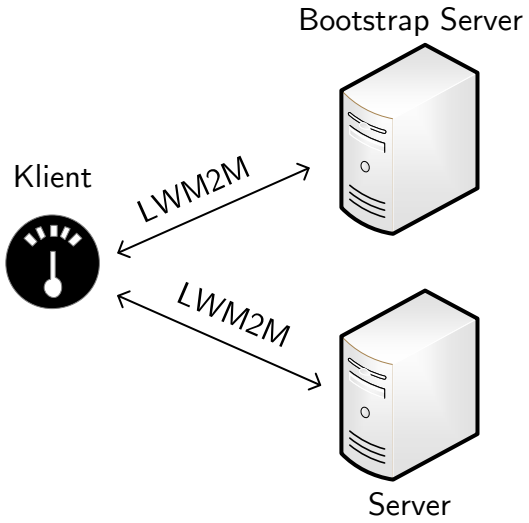


Obr. 4.1: Komunikační model protokolu LWM2M.

4.1 Struktura komunikačního řetězce LWM2M

Protokol LWM2M definuje 3 entity komunikačního řetězce.

- **LWM2M Klient -** LWM2M klient je zařízení, zodpovědné za ukládání a správu objektů. Klient komunikuje se serverem, dle doručených žádostí provádí operace a odesílá vyžádané informace na server.
- **LWM2M Server -** LWM2M server je zařízení zodpovědné za výčet a úpravu objektů nacházejících se na klientském zařízení.
- **LWM2M Bootstrap Server - Bootstrap server je zodpovědný za prvotní nastavení klienta a poskytování informací potřebných k úspěšné registraci na LWM2M server.**



Obr. 4.2: Model komunikační struktury protokolu LWM2M.

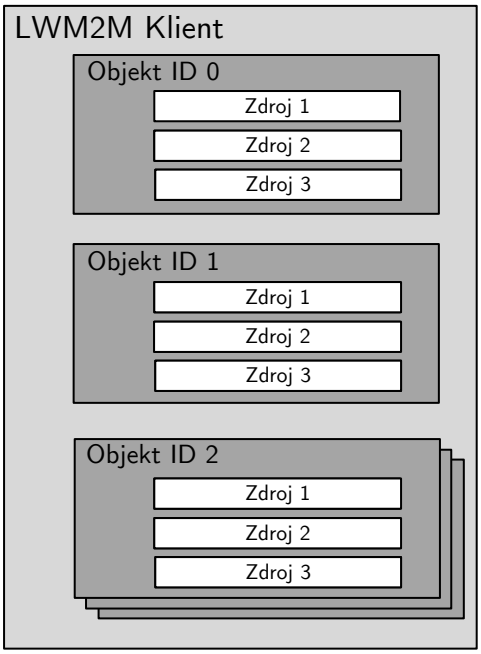
4.2 Datový model protokolu LWM2M

Pro výčet a zápis dat, správu zařízení a diagnostiku, využívá protokol LWM2M tzv. objektů. Objekt lze definovat jako základní entitu protokolu LWM2M obsahující uskupení zdrojů pozn pod čáru: zdrojem mohou být jakákoliv relevantní data např.: teplota v místnosti (22), jednotka teploty (°C) apod., které mezi sebou mají logickou vazbu. Struktura jednotlivých objektů je pevně stanovena organizací OMA. Objekty jsou uloženy na klientském zařízení kde k nim může být přistupováno kdykoliv ze strany serveru [27, 29].

4.2.1 Stavba objektu LWM2M

Každý objekt reprezentuje pevně definovanou strukturu zdrojů s různým zaměřením pro datovou komunikaci, bezpečnost a správu zařízení.

Objekt je reprezentován 16bitovým identifikačním číslem (Object ID) unikátním pro daný objekt. Každý objekt s odpovídajícím Object ID musí mít stejnou strukturu definovanou, či schválenou a registrovanou organizací OMA. V závislosti na daném objektu, může mít objekt několik aktivních instancí. Každá instance objektu je reprezentována 16bitovým číslem instance objektu (Object Instance ID) začínající od hodnoty 0 [29]. Každá instance objektu pak sdružuje zdroje specifické pro daný objekt.



Obr. 4.3: Datový model protokolu LWM2M.

4.2.2 Zdroje objektů LWM2M

Každá instance objektu sdružuje určitý počet logicky spolu souvisejících zdrojů. Zdroje jsou reprezentovány 16bitovým identifikačním číslem (Resource ID), které je v rámci instance objektu unikátní a předem pevně definované organizací OMA. Dle metod přístupu lze zdroje dělit na

- **Pouze pro čtení - Read only -** Zdroje sloužící pouze ke čtení, do kterých nelze zapisovat
- **Pouze k zápisu - Write only -** Zdroje sloužící pouze pro zápis, nelze z nich vyčítat data – vhodné například pro objekty pro zabezpečení
- **Pro čtení i pro zápis - Read and Write -** Zdroje, do kterých lze zapisovat, ale i z nich vyčítat data
- **Spustitelné zdroje - Executable -** Zdroje, které jsou spustitelné. Například spuštění měření, uživatelské aplikace či jiné operace.

Zdroje objektů mohou být povinné (mandatory – klient musí zajistit, že daný objekt bude tento zdroj obsahovat) nebo volitelné (optional). Každý zdroj má také dle specifikace OMA přiřazené vlastnosti jako datový typ (řetězec znaků - String, Číslo - Integer), zda je možné specifikovat více instancí zdroje (pole hodnot) nebo jednotky, ve kterých je hodnota uvedena [27, 29].

4.2.3 Atributy objektů LWM2M

Atributy jsou metadata, které mohou být přiřazeny jednotlivým objektům, jeho instancím nebo zdrojům. Hodnoty atributů jsou závislé na daném serveru a mohou nést informace o nastavení požadovaných akcí ze strany klienta (reportování dat a notifikace). Typy atributů jsou:

- **O Atribut -** Může být nastavený pro konkrétní objekt.
- **OI Atribut -** Může být nastavený pro konkrétní objekt nebo jeho instanci
- **R Atribut -** Může být nastavený pro konkrétní objekt, jeho instanci nebo pro zdroj objektu, případně pro danou instanci zdroje (pole hodnot).

Atributy jsou dále děleny na základě klasifikace:

- **Atributy vlastností - Properties -** Atributy popisující vlastnosti dané entity, které mohou být užitečné pro zpracování dat na LWM2M serveru. Mezi hlavní atributy vlastností patří atributy:
  - **Velikost - Dimension -** nese informaci o velikosti daného zdroje. Například velikost pole v 8bitové číselné podobě Integer.
  - **Krátké ID server - Short Server ID -** Nese informace o ID LWM2M serveru v 16bitové číselné hodnotě.
  - **URI serveru - Server URI -** Informace o URI serveru v datovém řetězci String.
  - **Verze objektu - Object Version -** Informace o verzi objektu v datovém řetězci String.
  - **"Enabler Version"- Nese informaci o aktuální podporované verzi protokolu LWM2M.**
- **Atributy pro notifikace - Notification -** Jsou atributy primárně určené pro nastavení zasílání (reportování) změn jednotlivých zdrojů. Nastavením příslušných atributů lze nastavit:
  - **Minimální periody zaslání zprávy -** Klientskému zařízení je umožněno zaslání zprávy o změně zdroje až po vypršení určitého intervalu
  - **Maximální periody zaslání zprávy -** Nedojde-li ke změně dat ve zdroji ve stanoveném intervalu, je přesto zaslána notifikační zpráva.
  - **Limity pro hodnotu zdroje -** Překročí-li datová hodnota zdroje nastavenou hodnotu, je zaslána notifikační zpráva [27].

4.2.4 Přístup ke zdrojům LWM2M

Z pohledu serveru je možné ke zdrojům objektů přistupovat na základě řetězce URI zaslaného v poli volitelných možností protokolu CoAP. Řetězec URI je vytvořen na základě identifikačního čísla objektu (Object ID), identifikačního čísla instance

objektu (Object Instance ID) a identifikačního čísla zdroje (Resource ID). Řetězec URI má následující podobu.

/Object ID/Object Instance ID/Resource ID

Příkladem může být výčet teploty z teplotního senzoru. Objekt registrovaný OMA pro teplotní čidla má Object ID = 3303 a ID zdroje pro hodnotu teploty je Resource ID = 5700. LWM2M server tedy odesílá požadavek s URI */3303/0/5700* [27].

Byla-li definice objektu změněna v navazujících verzích protokolu LWM2M, musí být součástí řetězce URI i informace o aktuální verzi daného objektu.

4.2.5 Povinné objekty LWM2M

Většina objektů LWM2M definovaných a registrovaných organizací OMA nemusí být na serveru ani klientovi implementovány, ale existují i objekty, které jsou pro implementování protokolu LWM2M povinné. Tyto objekty jsou:

- **LWM2M Security (Object ID 0)** – Objekt obsahující utajované informace ohledně připojení k různým serverům. Zdroje tohoto objektu jsou z bezpečnostních důvodů určeny pouze k zápisu, a to pouze v rozhraní bootstrap.
- **LWM2M Server (Object ID 1)** – Objekt obsahující veřejné informace o připojení k serveru
- **LWM2M Device (Object ID 3)** – – Objekt obsahující základní informace o zařízení včetně sériového čísla a výrobce.

Využívá-li klient více spojení na více serverů najednou, musí implementovat objekt Access Control (Object ID 2) [29].

4.3 Rozhraní protokolu LWM2M

Pro komunikaci mezi klientem a serverem rozlišuje protokol LWM2M čtyři rozhraní (interfaces)

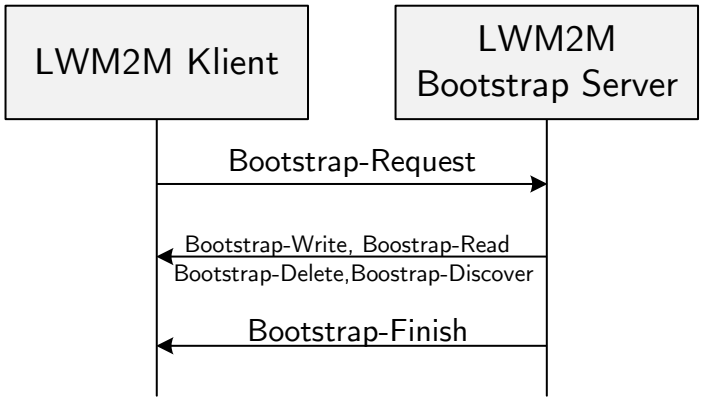
- **Rozhraní prvotní konfigurace - Bootstrap Interface**
- **Registrační rozhraní - Registration Interface**
- **Rozhraní správy zařízení a služeb - Device Management and Service Enablement Interface**
- **Rozhraní pro zasílání zpráv - Information Reporting Interface**

4.3.1 Rozhraní Bootstrap

Pro úspěšnou registraci k serveru musí být klientskému zařízení poskytnuty informace o konfiguraci serveru, o objektech, které server vyžaduje pro úspěšnou registraci a přihlašovací údaje. Tuto funkcionalitu zajišťuje rozhraní Bootstrap. Bootstrap rozhraní definuje několik metod [27].

- **Tovární Bootstrap -** Základní informace pro zařízení jsou přednastaveny výrobcem v paměti flash.
- **Bootstrap z extérního uložiště -** Potřebné informace jsou uloženy na uložišti (např. SD karta), ze kterého si zařízení může informace stáhnout.
- **Bootstrap inicializovaný klientem -** Potřebné informace jsou uloženy na tzv. Bootstrap serveru. Požadavkem pro tuto metodu je přednastavení potřebných údajů pro kontaktování Bootstrap serveru na klientském zařízení.
- **Bootstrap inicializovaný serverem -** Bootstrap server zašle požadavek o vyvolání Bootstrapu inicializovaného klientem

Při bootstrapu inicializovaného klientem je zaslána žádost Bootstrap-request a následně jsou Bootstrap serverem odeslány požadované informace. Procedura prvotní konfigurace je ukončena zprávou Bootstrap-Finish. Bootstrap serverem může být „obyčejný“ server nebo server speciálně určený pro tuto proceduru [27].



Obr. 4.4: Komunikace LWM2M rozhraní Bootstrap

Mapování metod protokolu CoAP pro Bootstrap rozhraní

Pro komunikaci v rozhraní prvotní konfigurace zastávají přístupové metody protokolu CoAP následující funkce:

- **Metoda POST -** Zastupuje zahájení a ukončení procedury v závislosti na předaných parametrech

  - **POST /bs?ep=Název klienta -** zastupuje zprávu zahájení procedury Bootstrap-Request.
  - **POST /bs -** zastupuje ukončení procedury.
- **Metoda PUT -** reprezentuje zápis ze strany serveru – Bootstrap-Write.
- **Metoda DELETE -** reprezentuje žádost o smazání záznamu/objektu - Bootstrap-Delete.
- **Metoda GET -** reprezentuje žádost o čtení stávajícího nastavení Bootstrap-Read.

4.3.2 Registrační rozhraní

Registrační rozhraní je využito pro registraci klienta k jednomu nebo více serverů. Každý server musí podporovat všechny operace registračního rozhraní. Při registraci provádí klient operaci „Register“ ve které specifikuje potřebné údaje pro registraci (podporované objekty, existující objekty, název klienta). Po vytvoření registrace je udržováno spojení mezi klientem a serverem, kdy klientské zařízení, na základě dohodnuté periody, zasílá serveru zprávy „Update“. Operace Register a Update informují server o dostupnosti zařízení, tedy o připravenosti přijímat zprávy na rozhraních správy zařízení a služeb a zasílání zpráv. Zpráva Register také obsahuje položku Lifetime, což je doba, po kterou je předpoklad aktivního spojení ze strany klienta. Klient může kdykoliv během tohoto intervalu odeslat zprávu Update s novou informací o této době. Po vypršení této doby je spojení automaticky považováno za ukončené. Pro indikaci odpojení zařízení je využito zprávy De-register [27].

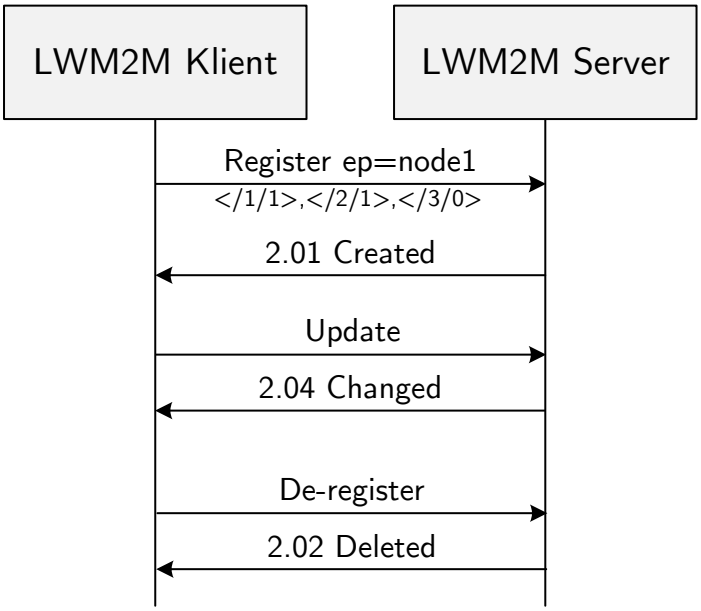
Mapování metod protokolu CoAP pro registrační rozhraní

Registrační rozhraní využívá pro signalizaci dvou metod protokolu CoAP.

- **Metoda POST -** Reprezentuje zprávy Register a Update.
  - **POST /rd?ep=Název klienta -** reprezentuje zprávu Register
  - **POST -** reprezentuje zprávu Update
- **Metoda DELETE -** Reprezentuje zprávu De-register [27].

4.3.3 Rozhraní správy zařízení a služeb

Rozhraní správy zařízení a služeb je hlavním rozhraním pro komunikaci a přenos dat protokolu LWM2M. Po úspěšné registraci klienta k serveru na registračním rozhraní je možné z pozice serveru číst, zapisovat, spouštět, vytvářet, mazat a modifikovat objekty a jejich zdroje na klientském zařízení. Rozhraní správy zařízení a služeb definuje metody přístupové metody k objektům a pro jejich správu.



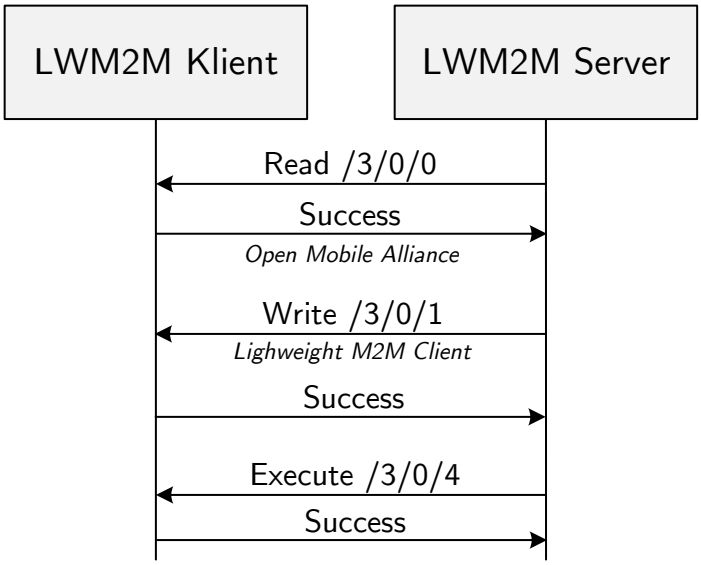
Obr. 4.5: Komunikace LWM2M rozhraní registrace

- **Discover -** Metoda pro získání atributů náležící danému objektu, jeho instanci nebo zdroji. Je možné ji také využít pro zjištění, které zdroje jsou v daném objektu instanciovány.
- **Create -** Metoda pro vytváření objektů na straně klienta. Metoda Create musí být cílena na objekt.
- **Read -** Metoda využitá pro přístup a čtení jednotlivých zdrojů objektu. S metodou Rad souvisí také metoda Read-Composite sloužící pro vyčtení několika zdrojů v jedné zprávě.
- **Write -** Metoda pro změnu dat v daném zdroji. Stejně jako u metody read i metoda write podporuje metodu pro zápis do více zdrojů najednou Write-Composite.
- **Delete -** Metoda využitá pro smazání celé instance objektu nebo jeho zdrojů.
- **Execute -** Metoda pro aktivování spustitelných zdrojů v daných objektech.
- **Write-attributes -** Metoda pro přiřazení jednotlivých atributů jednotlivým zdrojům [27].

Mapování metod protokolu CoAP pro rozhraní správy zařízení a služeb

Rozhraní správy zařízení a služeb, stejně jako ostatní rozhraní, využívá zprávy protokolu CoAP pro reprezentaci přístupových metod.

- **Metoda GET -** Metoda GET reprezentuje v závislosti na předaných parametruch metody Discover a Read.



Obr. 4.6: Komunikace LWM2M rozhraní správy zařízení a služeb

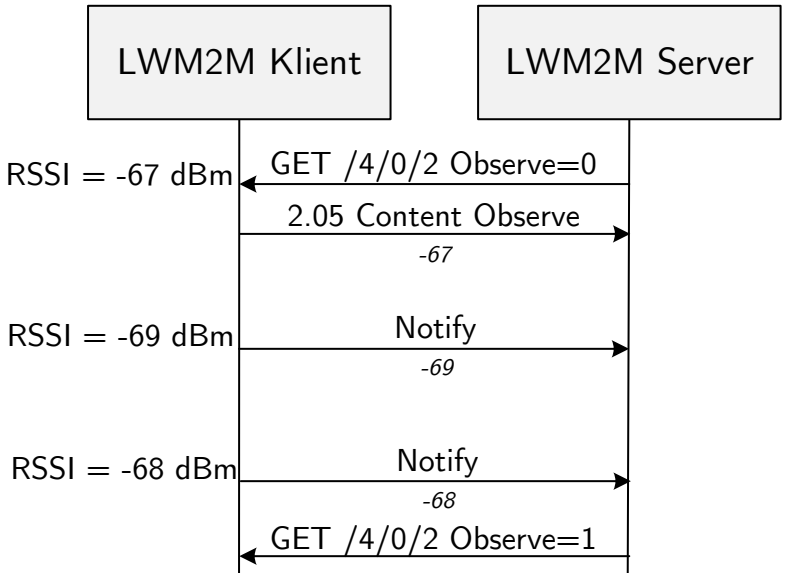
  - **GET Accept:application/link-format -** reprezentuje metodu Discover pro získání atributů objektu, instancí a zdrojů
  - **GET -** reprezentuje metodu read.
- **Metoda PUT -** reprezentuje metody Write a Create.
- **Metoda POST -** reprezentuje metodu Execute.
- **Metoda DELETE -** reprezentuje metodu Delete [29].

4.3.4 Rozhraní pro zasílání zpráv

Jedná se o rozšíření rozhraní správy zařízení a služeb. Umožňuje serveru periodické získávání aktualizací o stavu a změně jednotlivých objektů. Server odešle klientovi CoAP zprávu GET s URI požadovaného zdroje/objektu s nastavenou volitelnou možností Observe=0. Klient při změně hodnoty sledovaného parametru odesílá serveru zprávu Notify s informací o změně a aktuální hodnotou sledovaného parametru. Zrušení metody Observe je provedeno zasláním zprávy Read s nastavenou volitelnou možností Observe=1, případně zasláním CoAP zprávy RST (Reset) jako odpověď na zaslanou notifikační zprávu (Notify).

4.4 Komunikační scénář protokolu LWM2M

V první fázi komunikace musí dojít ke stažení potřebných informací k registraci (Bootstrapping). Klient tuto operaci může provést načtením konfigurace z paměti flash, paměťového uložiště nebo může kontaktovat Bootstrap Server s žádostí o předání



Obr. 4.7: Komunikace LWM2M rozhraní pro zasílání zpráv

těchto informací metodou Boostrap-Request. Po ukončení Bootstrappingu metodou Bootstrap-Finished, může klient kontaktovat samotný LWM2M server s žádostí o registraci (metoda Register), kde dojde k předání informací o podporovaných objektech a aktuálně vytvořených objektech. Součástí registrační zprávy je i informace o podporovaných formátech zpráv (Plain text, JSON, TLV apod.) a také doba života (lifetime) udávající dobu v sekundách, po kterou je zařízení považováno za registrované. Během této doby, typicky při probuzení zařízení z režimu spánku, či těsně před vypršením této doby, je nutné zaslat serveru zprávu Update, jinak dojde k deregistraci zařízení.Po dokončení registrace je umožněna komunikace mezi klientem a serverem, tedy vytváření, modifikace či výčet jednotlivých zdrojů a objektů.

4.5 Zabezpečení protokolu LWM2M

Pro verzi protokolu LWM2M 1.0 je využito šifrovacího protokolu DTLS pracujícím nad protokolem UDP na 6. vrstvě modelu OSI/ISO. Ve verzi protokolu 1.1 byly definovány objekty a metodiky pro implementaci zabezpečovacího protokolu OSCORE (Objekt 21 – LWM2M OSCORE).

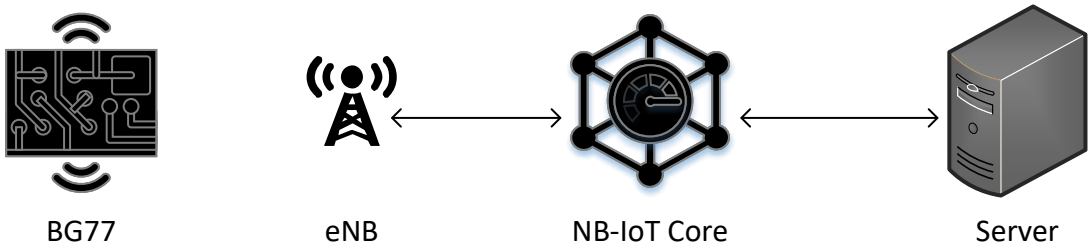
Pro přístup k Bootstrap serveru a k registraci na LWM2M serveru je zapotřebí znalost přístupových údajů a je nutná autentizace certifikátem vydaným příslušným správcem serveru.

5 Srovnání protokolů pro přenos dat LPWA

Tato část práce pojednává o srovnání výše popsaných protokolů z pohledu objemu přenesených uživatelských dat. Pro testovací účely bylo využito NB-IoT modulu BC68 čínského výrobce Quectel, který implementuje hardwarovou realizaci protokolů UDP, TCP, MQTT a CoAP, což ho činí ideálním pro daný test.

5.1 Metodika měření

Pro každý protokol bylo celkově zasláno 20 zpráv různých velikostí (12, 64, 128, 256 B) s rozestupem 30 sekund mezi každou zaslanou zprávou. Nastavení klientské strany pro jednotlivé protokoly bylo realizováno modulem BG77 za pomoci příslušných AT příkazů specifikovaných v produktových dokumentech modulu BG77. Nastavení serveru bylo realizováno příslušným skriptem v jazyce python (UDP, TCP), popřípadě open-source aplikací (MQTT – Mosquitto, CoAP – Leshan LWM2M). Měření bylo provedeno v prostředí s dobrými rádiovými podmínkami -72 dBm. Po provedení každého měření byly získány statistiky ze síťového provozu příslušného protokolu na straně serveru aplikací Wireshark.



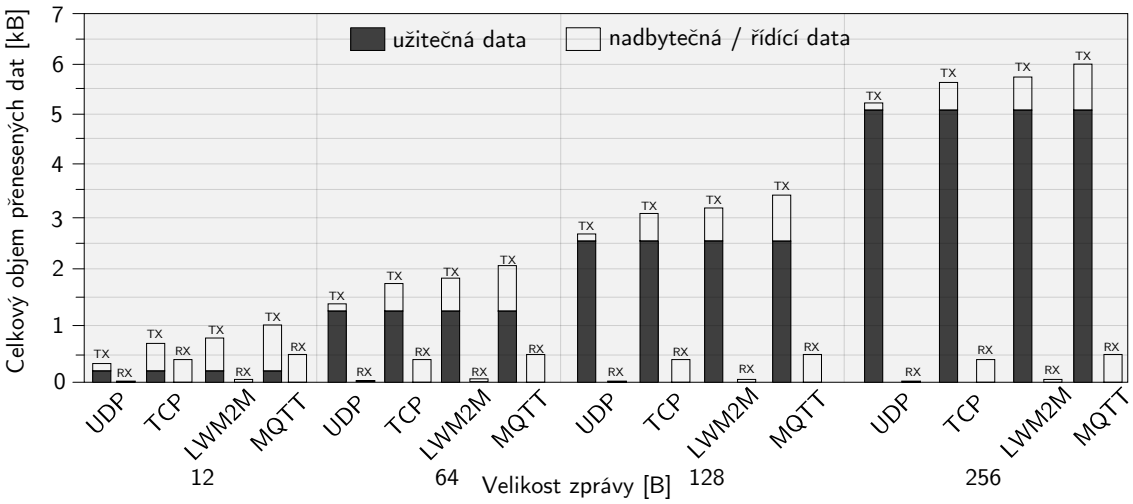
Obr. 5.1: Architektura realizace měření

5.2 Výsledky měření

Po provedení měření byly výsledky zpracovány do grafické podoby a následně evaluovány.

Obrázek 5.2 ilustruje objem dat přenesených ve směrech uplink (první sloupec) a downlink (druhý sloupec) u jednotlivých protokolů. Sloupec je rozdělen do dvou částí, kde vrchní část (přerušovaná čára bez výplně) zobrazuje nadbytečná řídící data protokolu (overhead) a spodní část uživatelská data, tedy samotnou zprávu.

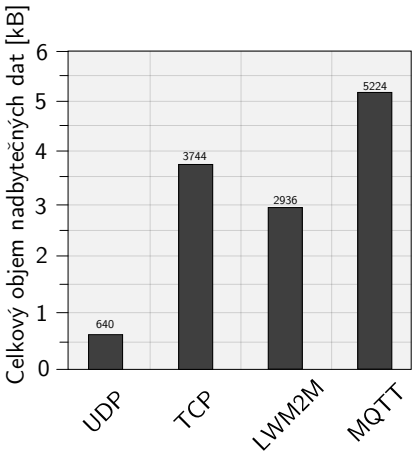
Z grafu lze vidět, že při využití protokolu UDP je objem přenesených dat nejnižší, což je způsobeno nulovou přenosovou režií (není zapotřebí žádná registrace,



Obr. 5.2: Srovnání přenosových protokolů na základě objemu přenesených dat

ani ustanovení spojení) a nenáročným záhlavím protokolu UDP. Protokol CoAP, vystavěný nad protokolem UDP, přidává režii do přenosů dat, což zvyšuje celkový objem přenesených dat, ale zaručuje spolehlivost doručení dat.

U protokolů TCP a MQTT lze pozorovat vysoký objem přenesených dat i v případě zaslání zprávy o malé velikosti, což je způsobeno nutností ustanovení spojení, registrace (u protokolu MQTT) a potvrzování zpráv. Lze vidět, že objem registračních a řídících dat může být v případech zaslání malých zpráv až několikanásobný oproti užitečným datům, které chce uživatel přenést. To je nevýhodné především u technologií licenčního spektra, kde je daným operátorem přenos těchto dat zpoplatněn [21].



Obr. 5.3: Srovnání na základě objemu přenesených nadbytečných dat.

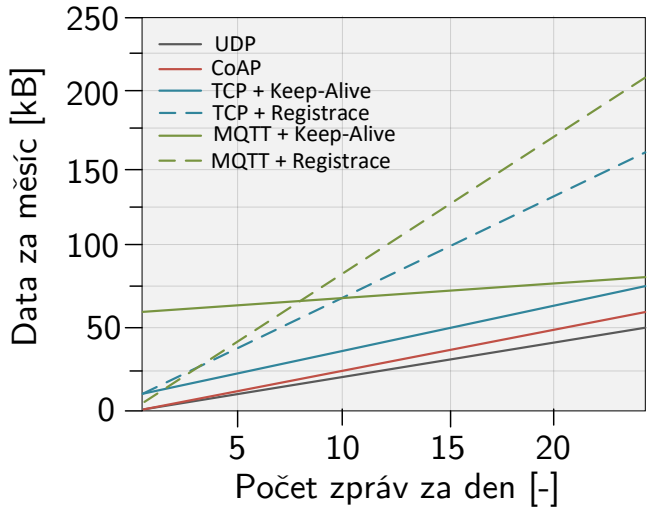
Obrázek 5.3 ilustruje součet celkových objemů přenesených nadbytečných dat pro jednotlivé protokoly. V porovnání s grafem 5.2 lze vidět, že přestože protokol CoAP/LWM2M generuje více nadbytečných dat ve směru uplink než protokol TCP, celkový objem nadbytečných dat je ve výsledku nižší, přičemž spolehlivost přenosu je zachována.

Po získání dat ze serveru byla sestavena tabulka 5.1, porovnávající minimální objem nadbytečných dat v bajtech pro jednotlivé protokoly ve všech fázích komunikace. Tabulka také zobrazuje objem přenesených dat pro udržení stávajícího spojení (keep-alive). Z tabulky lze opět pozorovat nižší hodnoty objemu nadbytečných dat způsobený nižší režií protokolů založených na UDP, a to ve všech fázích komunikačního scénáře.

Protokol	**Setup**		**Přenos dat**		**Keep-Alive**		**Terminace**	
-	UL	DL	UL	DL	UL	DL	UL	DL
UDP	-	-	8	0	-	-	-	-
TCP	44	24	20	20	20	20	20	40
CoAP	83	48	20	0	0	0	21	16
MQTT	118	68	28	20	42	42	20	40

Tab. 5.1: Minimální objem nadbytečných dat.

Pro porovnání dlouhodobého přenosu dat byl sestavený graf zobrazující objem přenesených dat za měsíc v závislosti na denním počtu přenesených zpráv. Graf zobrazuje porovnání metody udržení spojení keep-alive oproti registraci při každém přenosu dat.



Obr. 5.4: Srovnání přenosových protokolů z pohledu dlouhodobého přenosu dat

Z grafu lze vidět, že u protokolu MQTT je pro počet zaslaných zpráv za den větší než 8 výhodnější využít metodu keep-alive, jelikož je výhodnější z hlediska objemu přenesených dat oproti opakované registraci.

6 Implementace protokolu LWM2M

Tato část práce se zabývá detailním popisem vytvořené knihovny pro protokol LWM2M, jak z hlediska vnitřního fungování, tak z hlediska její uživání. Knihovna je napsána v programovacím jazyce *Embedded C++* (EC++) a následuje standard protokolu LWM2M 1.0 definovaný skupinou OMA v prvním vydání technické specifikace ze dne 8. února 2017 [30].

6.1 Srovnání C++ a Embedded C++

Embedded C++ je verzí (dialekt) programovacího jazyka C++, která je navržená a optimalizovaná pro programování embedded zařízení s ohledem na limitující parametry těchto zařízení (typicky nízká velikost paměti RAM, nižší výpočetní výkon / efektivita instukční sady). Cílem vzniku jazyka Embedded C++ bylo umožnit vývojářům vývoj aplikací pro embedded systémy v objektově orientovaném programovacím jazyce C++ při zachování pravidel pro programování mikroprocesorové techniky [31, 32].

Základní požadavky na jazyk pro embedded systémy:

- Minimalizace okupace paměti.
- Nevyvolávat nepredikovatelné stavy.
- Umožnění nahrávání kódu do pamětí ROM.

Oproti jazyku C++, jazyk EC++ klade důraz na minimalizaci výsledné velikosti kódu, při zachování objektové struktury jazyka a maximalizaci efektivity využití instrukční sady mikroprocesoru/mikrokontroléru. Z těchto důvodů EC++ omituje některé rozšíření jazyka C++, které jsou dostupné ve standartní verzi [31, 32].

Rozšíření	**C++**	**Embedded C++**
Šablony (Templates)	Ano	Ne
Dědičnost (Inheritance)	Ano	Ne
Vyjímky (Exceptions)	Ano	Ne
Informace o typu při běhu programu	Ano	Ne
Přetypování proměnných ve stylu C++ (static_cast...)	Ano	Ne
Prostředí (Namespaces)	Ano	Ne

Tab. 6.1: Rozdíly C++ a EC++.

Embedded C++ zachovává zpětnou kompatibilitu s jazykem C++ (kód napsaný v EC++ lze spustit v plné verzi standardu C++) [31].

6.2 Modularita knihovny

Knihovna je navržena k jednoduchému použití nezávisle na konkrétním výrobci, či typu systému. Tento systém s sebou však nese několik nevýhod, z nichž největší je nutnost použití vlastních metod a funkcí pro interakci s přenosovým systémem, což znesnadňuje prvotní integraci knihovny.

Minimálním požadavkem na uživatele knihovny je vytvoření funkcí pro zasílání a pro příjem zpráv. Také je nutné vytvořit funkci pro restartování zařízení, kterou následně předává do konstruktoru při inicializaci třídy klienta. Tyto funkce jsou následně využívány funkcemi uvnitř knihovny a je proto nutné je definovat před vytvořením objektu klienta.

V závislosti na nastavení knihovny jsou zasílané zprávy automaticky odesílány ihned nebo jsou předány do TX bufferu, což je indikováno příslušným TX flagem, jehož hodnotu může uživatel vyčíst a data odeslat "manuálně".

6.3 Základní architektura knihovny

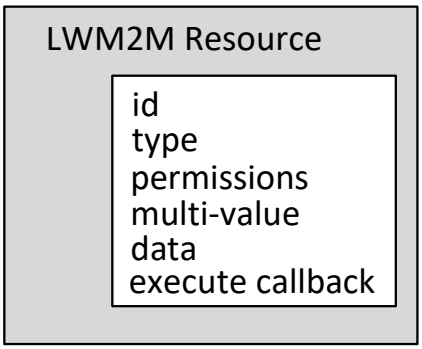
Základní architektura knihovny je postavena na třech hlavních třídách, LWM2M Client, LWM2M Object a LWM2M Resource a jednoho definičního souboru LWM2M Defines, kde jsou definovány některé z důležitých konstant knihovny a kde také lze pomocí preprocesorových direktiv upravovat výsledné chování knihovny (příkladem může být definování datových formátů pro čtení jednotlivých zdrojů, či celých bloků instancí/objektů). Mimo tyto hlavní třídy jsou v knihovně obsaženy i doplňující třídy, zejména pro formátování výstupních a parsování vstupních dat (knihovna pro CoAP protokol, formát JSON ...).

6.3.1 Třída LWM2M Resource

Třída LWM2M Resource uchovává informace o daném zdroji a implementuje základní metody pro čtení či zápis těchto informací. Informace obsažené ve třídě LWM2M Resource jsou:

- **ID zdroje** - Informace o ID zdroje.
- **Typ obsahu** - Informace o typu zdroje. Zda se jedná o číselnou hodnotu (Integer, Float), logickou hodnotu (Boolean), textový řetězeč (String) atp.

- **Přístupová práva** - Zda se jedná o spustitelný zdroj (Executable), pouze pro čtení (Read only), pouze pro zápis (Write only), či pro čtení a zápis (Read-Write).
- **Rozměry zdroje** - Logická hodnota vyjadřující, zda se jedná o multi-value zdroj, tedy o zdroj, který vystupuje jako pole hodnot.
- **Data** - Samotná data, která jsou ve zdroji uložena.
- **Callback funkce** - V případě, že je daný zdroj spustitelný, je v této proměnné uložena adresa funkce, která má být provedena po přijetí zprávy Execute.



Obr. 6.1: Architektura třídy LWM2M Resource

Vytváření jednotlivých zdrojů je realizováno konstruktorem, kde je nutné uvést všechny výše zmíněné parametry.

```
LWM2M_Resource(id, type, permissions, multi_level, value);
```

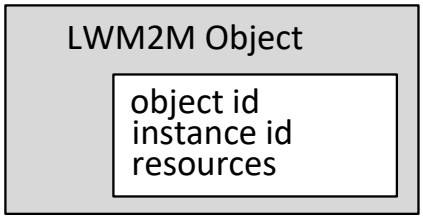
6.3.2 Třída LWM2M Object

Třída LWM2M Object reprezentuje v architektuře protokolu LWM2M jednu instanci objektu. Třída uchovává informace o ID daného objektu a také ID instance a jsou v ní sdruženy jednotlivé instance třídy LWM2M Resource, ke kterým je možné jednotlivě přistupovat přes ID zdroje, pomocí definované funkce. Informace obsažené ve třídě LWM2M Object jsou:

- **ID objektu** - Informace o ID objektu.
- **ID instance** - Informace o ID instance.
- **Instance třídy zdrojů**.

Třída je inicializována konstruktorem, kterému je nutné předat ID objektu a nepovinně i ID instance. Pokud není ID instance uvedeno, je automaticky nastaveno na hodnotu 0.

```
LWM2M_Object(object_id, instance_id = 0);
```



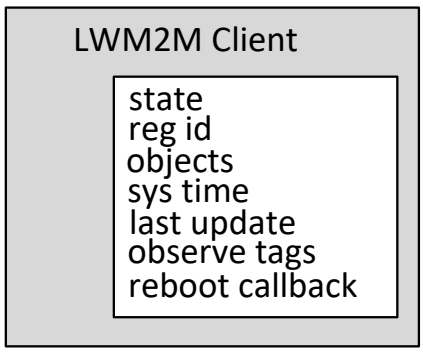
Obr. 6.2: Architektura třídy LWM2M Object

6.3.3 Třída LWM2M Client

LWM2M Client je hlavní třídou celé knihovny. Objekt vytvořený z této knihovny je hlavním prostředníkem pro interakci developera s knihovnou. Třída sdružuje funkce, metody a proměnné důležité pro zpracování požadavků, generování odpovědí a zpráv, správu objektů a všechny procedury týkající se registrace na LWM2M Server. Třída je také zodpovědná za časování odesílání zpráv update a observe.

Ve objektu třídy jsou uloženy důležité informace o aktuálním stavu, o registraci a o posledním updatu, informace o aktuálním čase od doby vytvoření obejktu a tagy pro metodu observe.

Při inicializaci objektu třídy je nutné předat v konstruktoru název koncového uzlu (endpoint name) a také odkaz na adresu funkce, která zapříčíní restart (reboot) zařízení. Důvodem je automatické vytvoření několika povinných objektů při vytváření objektu z této třídy, kde jeden z povinných zdrojů daného objektu je spustitelný zdroj, který restartuje zařízení. Vzhledem k modularitě knihovny musí být daná funkce vytvořena uživatelem knihovny před vytvořením samotného objektu.



Obr. 6.3: Architektura třídy LWM2M Client

6.4 Funkcionalita knihovny

Funkcionalita knihovny je řešena pomocí stavového automatu. Na základě jeho stavu je rozhodováno o dalším kroku programu.

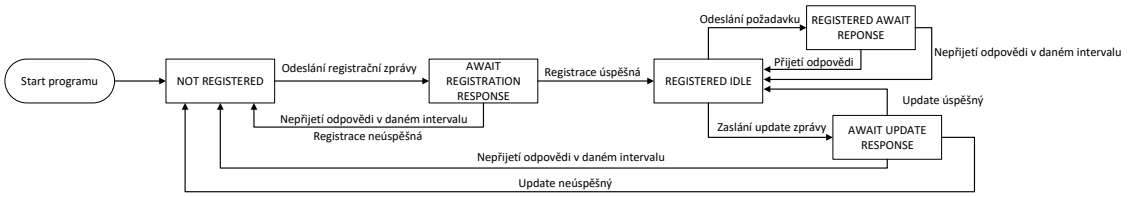
6.4.1 Stavový automat

Stavový automat je složen ze šesti jednotlivých stavů.

- **Not Registered** - Zařízení není registrováno k žádnému serveru.
- **Await Registration Response** - Zařízení očekává odpověď na zaslanou registrační zprávu.
- **Registered Idle** - Zařízení je úspěšně registrované a očekává příchozí zprávu.
- **Registered Await Response** - Zařízení je registrované a očekává odpověď na dříve zaslanou zprávu.
- **Await Update Response** - Zařízení je registrované a očekává odpověď na dříve zaslanou zprávu Update.

Po inicializaci klienta je počátečním stavem stav *Not Registered*. Po zaslání registrační zprávy je stav změněn na stav *Await Registration Response*. Při další přijaté zprávě v tomto stavu, je ověřeno, zda došlo k úšpěšné registraci, či nikoliv. V případě úspěchu je stav změněn na stav *Registered Idle*, v opačném případě je stav opět nastaven na stav *Not Registered*. V tomto stavu je možné přijímat či odesílat zprávy v závislosti na požadavcích serveru.

Pokud dojde k zaslání zprávy update, je stav změněn na stav *Await Update Response*. V tomto stavu může zůstává funkcionalita interakce se serverem zachována. Nedojde-li k přijetí potrvzovací zprávy update, je stav opět změněn na stav *Not Registered*, v případě přijetí je stav změněn na stav *Registered Idle*.



Obr. 6.4: Stavový diagram

6.4.2 Příjem zprávy

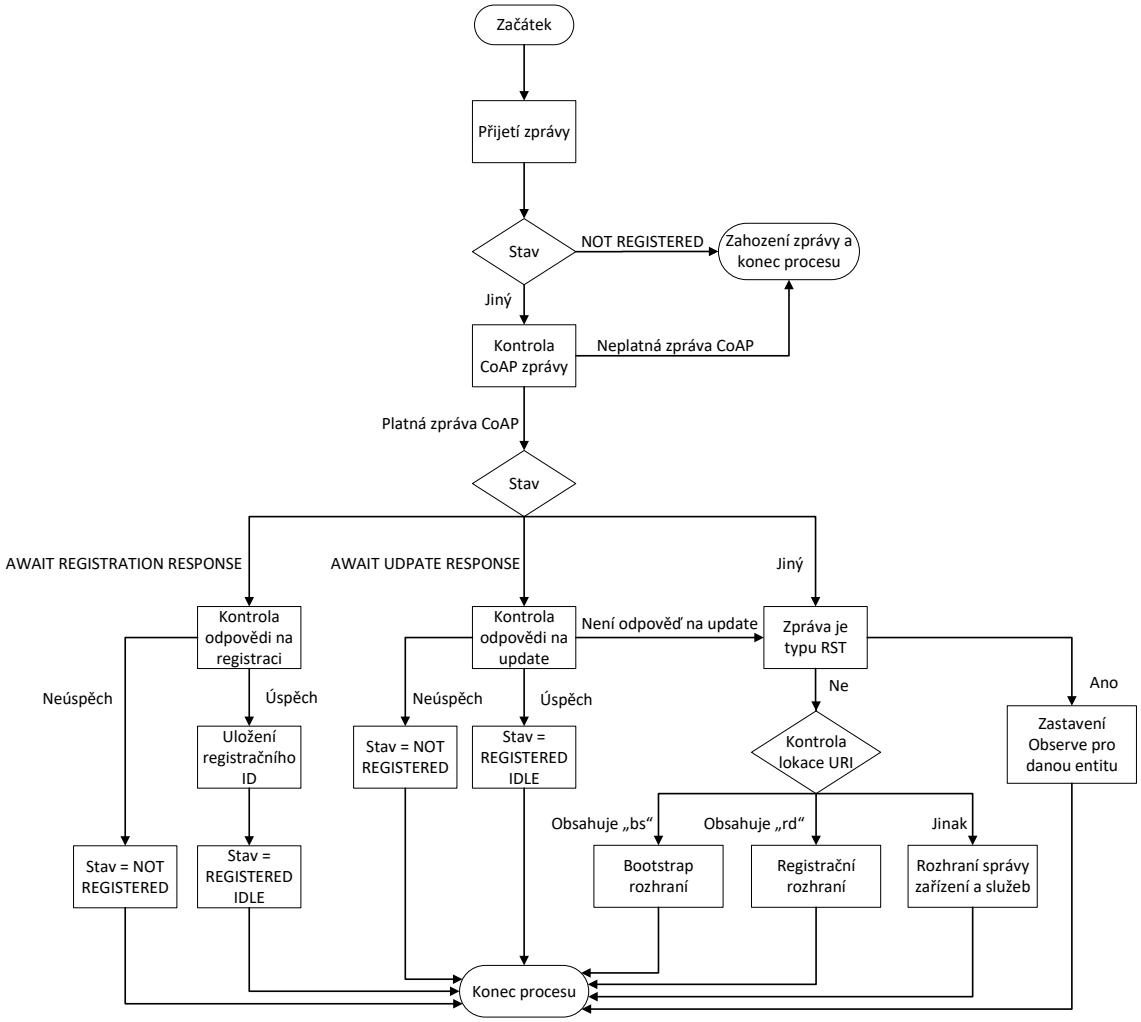
Po přijetí zprávy je nejprve kontrolováno, zda se zařízení nachází v jiném stavu než ve stavu *Not Registered*. Pokud je zařízení v tomto stavu, je zpráva zahozena. V

opačném případě je zpráva evaluována z hlediska kompability s protokolem CoAP. Není-li zpráva zprávou protokolu CoAP, je opět zahozena. Po evaluaci je na základě aktuálního stavu rozhodováno o následujícím procesu.

V případě, že je očekávána odpoveď na registrační zprávu, je přijatá zpráva podstoupena evaluaci z hlediska registrace. Výsledkem je změna stavu v závislosti na obsahu dané zprávy.

V případě, že je očekávána odpoveď na zprávu update, dochází k evaluaci odpovědi. Pokud se nejedná o odpověď na zprávu update, je předána k další evaluaci. Je-li zpráva odpovědí na zprávu update, je obsah zprávy evaluován a na základě výsledku je změněn stav.

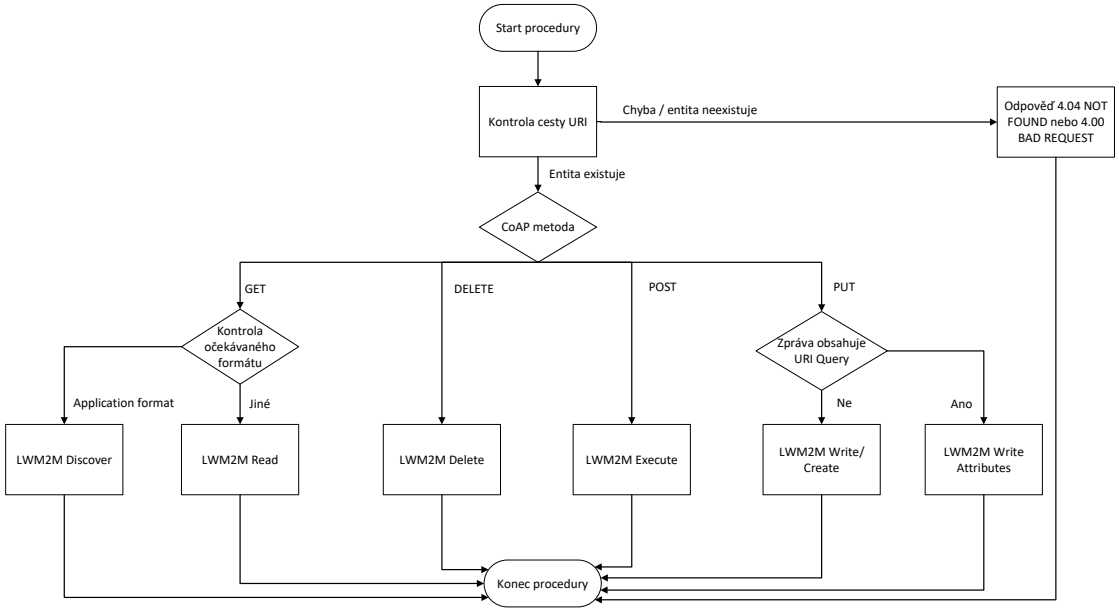
Pro ostatní případy je kontrolován typ zprávy. Pokud se jedná o zprávu typu RST je zpráva předána funkci pro řízení metody observe. V opačném případě je zpráva předána danému rozhraní protokolu LWM2M, dle obsahu.



Obr. 6.5: Vývojový diagram pro zpracování příjmu zprávy

6.4.3 Rozhraní správy zařízení a služeb

Po předání zprávy tomuto rozhraní je v první řadě kontrolována cesta URI, která je obsahem předané zprávy. V případě, že je cesta neplatná nebo daný objekt/zdroj, na který se URI odkazuje neexistuje, je odesílateli odeslána zpráva s chybovým kódem a procedura ukončena. V případě, že cesta URI odkazuje na existující entitu, je na základě uvedené přístupové metody protokolu CoAP, obsažené ve zprávě, rozhodnuto o korespondující metodě protokolu LWM2M. V případě zpráv GET a PUT je navíc ještě rozhodováno o typu metody. Zpráva a platné URI je dále předáváno do daných dílčích metod knohovny protokolu LWM2M, z nichž nejvýznamnější jsou popsány níže.



Obr. 6.6: Vývojový diagram pro rozhraní správy zařízení a služeb

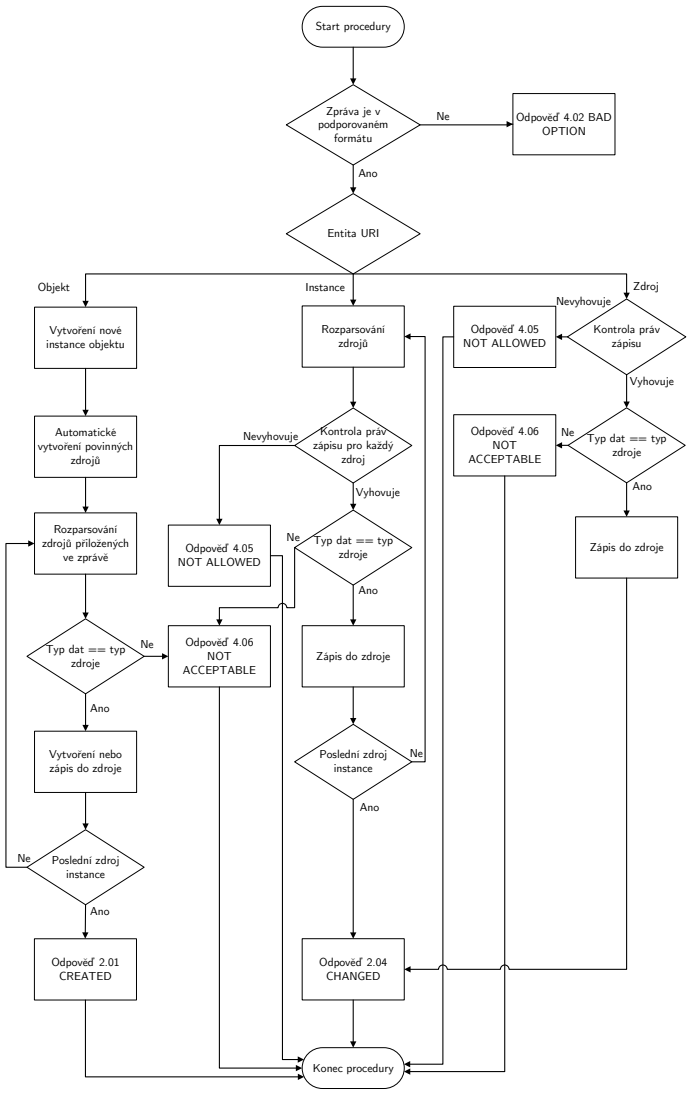
Metoda Write/Create

Prvním krokem metody Write/Create je kontrola formátu dané zprávy. Ta je obsažena ve volitelných možnostech zprávy CoAP. Daný formát je porovnáván s formáty zvolenými uživatelem. V případě, že je formát přijaté zprávy jiný, než podporovaný, je odpovězeno chybovou odpovědí Bad Option. Pokud je příchozí formát v souladu s nastavením knihovny je na základě požadované entity k zápisu (dostupné z URI) vyhodnoceno, zda se jedná o zprávu Write, či Create. V případě metody Create MUSÍ být cílovou entitou objekt, v jiném případě se jedná o metodu Write.

Pokud se jedná o metodu Create je v první řadě vytvořena nová instance objektu a také povinné zdroje. Následně jsou ze zprávy parsovány jednotlivé zdroje u

jejíchž hodnot dochází k porovnání typu zdroje s daty (číslo, řetězec znaků, logická hodnota). Pokud je porovnání úspěšné, jsou data zapsány do již existujícího zdroje (pokud byl vytvořen v rámci povinného zdroje), případně je vytvořen nový zdroj, který je do instance přidán. Proces se opakuje až do poslední položky a následně je navrácena odpověď CREATED.

V případě metody Write je postup závislý na entitě, na kterou odkazuje URI. Pokud se jedná o zdroj, jsou nejprve kontrolována přístupová práva (musí být povolen zápis do zdroje - Read/Write, Write only) a následně datový typ stejnou metodou, jako u metody Create. Pokud jsou podmínky splněny, jsou data zapsány do zdroje. V případě instance je tento proces opakován pro každý zdroj obsažený ve zprávě.



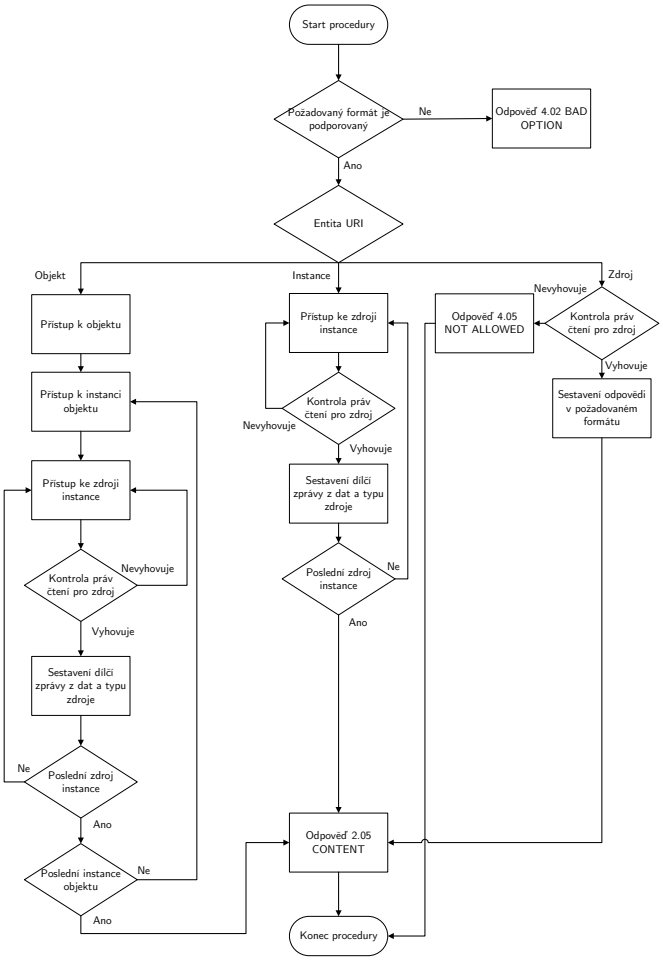
Obr. 6.7: Vývojový diagram pro metodu Write a Create

Metoda Read

Metoda Read slouží k vyčtení dat z dané entity. Stejně jako u ostatních metod, je na základě dostupného URI rozhodováno o následujícím procesu. Nejdříve je však kontrolován požadovaný formát odpovědi. Pokud je formát knihovnou a konfigurací podporován, je přisoupeno k dalšímu kroku, pokud nikoliv, je o tom žadatel informován chybovou odpovědí. Je-li požadovanou entitou zdroj jsou kontrolovány práva ke čtení daného zdroje. Pokud je zdroj typu Read/Write nebo Read only, je na základě požadovaného formátu předaného ve zprávě CoAP sestavena zpráva, která je následně odeslána žadateli o vyčtení entity.

U požadavku na entitu instance je tato procedura provedena pro každý zdroj. Pokud nelze daný zdroj instance číst (není typu Read/Write, Read only) je z datového toku vynechán.

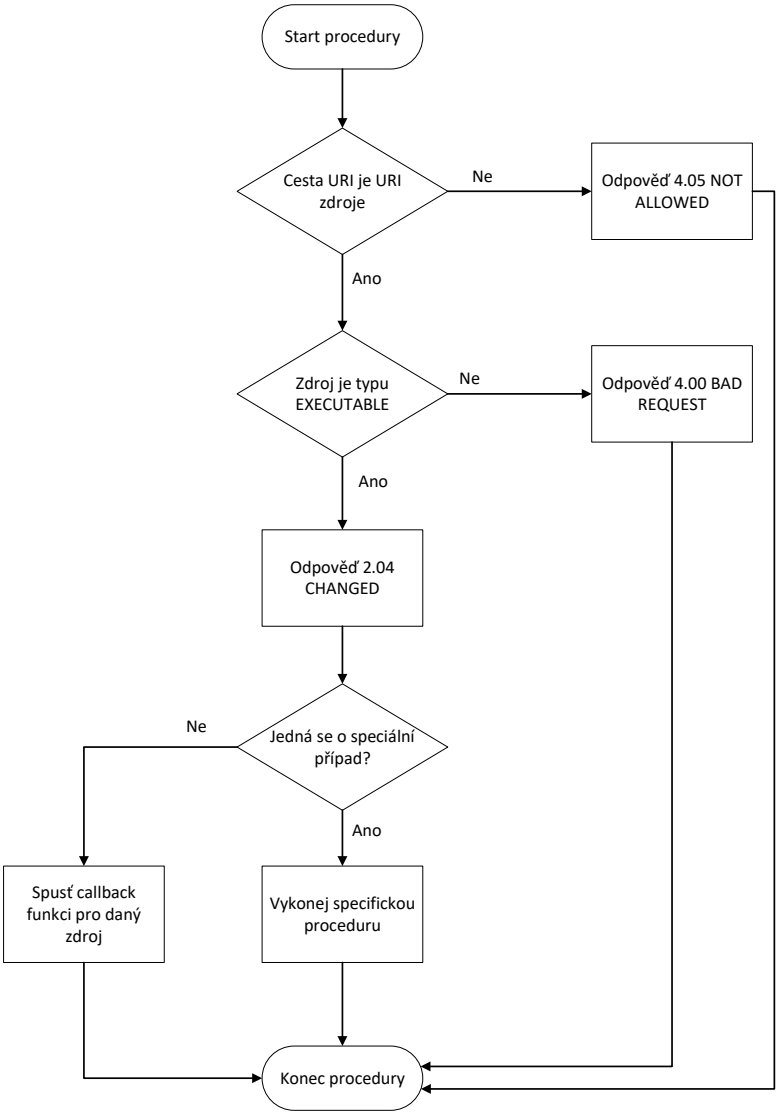
Ukazuje-li cesta URI na objekt, jsou data sestavena dle výše uvedených pravidel z každé instance a každého zdroje v této instanci.



Obr. 6.8: Vývojový diagram pro metodu Read

Metoda Execute

Na základě předaného URI z funkce vyšší vrstvy je nejprve ověřeno, zda daná cesta URI poukazuje na entitu zdroje a následně je ověřováno, zda se jedná o zdroj spustitelný (Executable). Při nevyhovění jedné z těchto podmínek je odeslána příslušná chybová odpověď a procedura je ukončena. Při splnění obou podmínek je kontrolováno, zda se nejedná o jeden z několika speciálních případů, případů u kterých je zapotřebí interně měnit hodnoty či spouštět uživateli nedostupné funkce. Příkladem může být požadavek na restart zařízení, či příkaz k zaslání zprávy update. Následně je spuštěna daná funkce, jejíž adresa je uložena v daném zdroji.



Obr. 6.9: Vývojový diagram pro metodu Execute

6.4.4 Řešení zasílání zpráv Update

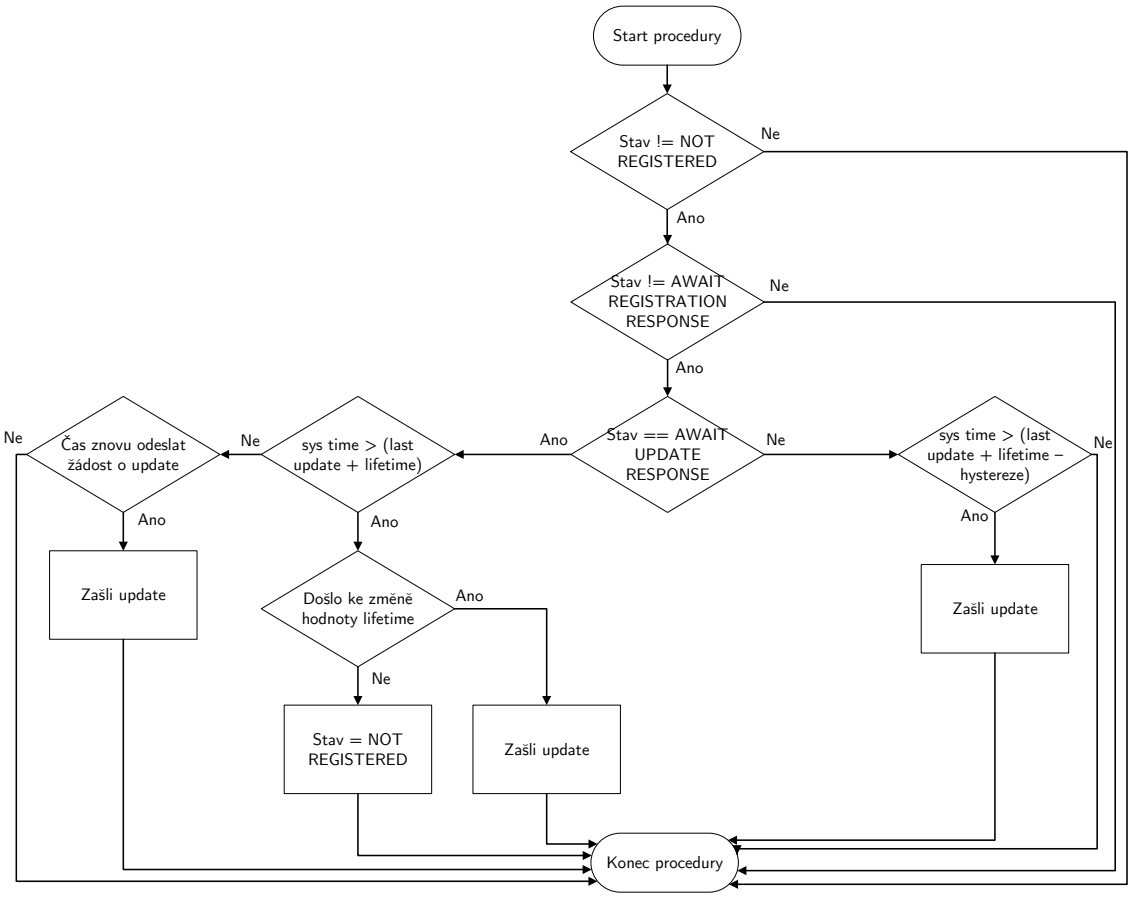
Pro udržení registrace na serveru, je zapotřebí periodicky zasílat zprávy update. K tomuto účelu je knihovna vybavena funkcí rutiny update.

Při každé změně hodnoty systémového času je spuštěna procedura vyhodnocující aktuální stav registrace. Výsledkem této procedury je zaslání zprávy update v době před vypršením registrace.

Jelikož se zařízení může nacházet v proměnlivých rádiových podmínkách, může docházet k výraznému zpoždění zasílání zpráv. Z tohoto důvodu implementuje knihovna základní hysterezi, která odesílá update zprávu s předstihem, před vypršením registrace. Velikost hystereze v sekundách, je možné nastavovat v souboru LWM2M Defines. Také je implementován mechanizmus opětovného zasílání update zprávy v pravidelných časových intervalech v případě nedoručení update zprávy na server.

V první fázi jsou kontrolovány stavy zařízení. Pokud zařízení není registrováno k serveru, je zbytečné vyhodnocovat, zda odeslat zprávu update. Pokud je zařízení registrováno a není ve stavu, kdy očekává odpověď na zprávu update, je na základě hodnoty lifetime, aktuálního systémového času a doby posledního odeslání zprávy evaluováno, zda odeslat zprávu update. V případě, že je podmínka splněna je zpráva update ihned zaslána. Pokud je zařízení ve stavu očekávání odpovědi na zprávu update, jsou kontrolovány dva stavy.

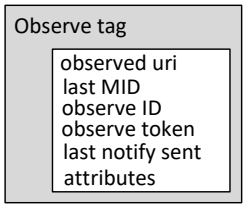
Prvním stavem je kontrola zda nedošlo k přetečení hodnoty lifetime. V takovém případě je také kontrolováno, zda nedošlo ke změně hodnoty lifetime, například lokálním přepisem nebo přepisem ze strany serveru. Nedošlo-li ke změně doby života, je registrace považována za expirovanou a zařízení je nastaveno do stavu *Not registered*. Druhým stavem je kontrola opětovného zaslání zprávy update. Pokud je doba od zaslaní poslední zprávy update rovná době pro odeslání nové žádosti o update, je zpráva update opětovně odeslána.



Obr. 6.10: Vývojový diagram pro udržení registrace (update)

6.4.5 Řešení zasílání zpráv Notify

Pro realizaci metody observe, obsahuje třída LWM2M Client pole observačních tagů. Pro každou observovanou entitu je vytvořený tag, který obsahuje informace o nastavených atributech pro danou observovanou entitu. Po přijetí žádosti o metodu Read, je vyhodnoceno, zda volitelné možnosti zprávy CoAP obsahují nastavení Observe. Pokud ano, je vytvořen nový tag, který je uložen v poli tagů.



Obr. 6.11: Struktura observační struktury (tagu)

S každou změnou hodnoty systémového času je pole tagů procházeno a na základě nastavených atributů pro danou entitu rozhodováno, zda je žádoucí odeslání zprávy Notify.

6.5 Použití knihovny

Před samotným použitím knihovny pro komunikace je nutná implementace několika funkcí ze strany vývojáře.

6.5.1 Prerekvizity

Reboot

Již zmíněnou prerekvizitní funkcí je funkce *reboot*, sloužící k restartu zařízení. Návratová hodnota funkce musí být 8 bitové celé číslo, kde hodnota 0 značí úspěch, jiná hodnota značí neúspěch. Příklad implementace funkce reboot, kde nedojde-li k úspěšnémů restartu je navrácena hodnota 1. Při úspěšném restartu by se program neměl nikdy dostat do bodu návratu z funkce.

```
uint8_t reboot()
{
    NVIC_SystemReset();
    return 1;
}
```

Výpis 6.1: Příklad implementace funkce reboot.

Funkce pro zasílání a příjem dat

Implementace těchto funkcí je nutná pro komunikaci se serverem. Z důvodu modularity nejsou tyto funkce implementovány v balíčku knihovny a je tedy nutné tyto funkce vytvořit pro konkrétní přenosové rozhraní.

Funkce pro odeslání dat musí vracet celočíselnou 8 bitovou hodnotu a přebírat argumenty ukazatele na pole znaků, společně s 16bitovou celočíselnou hodnotou reprezentující délku zasílaných dat. Návratová hodnota značí úspěch (0) a neúspěch (jiné). Příklad implementace funkce pro zasílání dat:

```
uint8_t send_data(char* data, uint16_t data_len)
{
     return BG77.send(SOCK1, data, data_len, SVR_IP, 5683);
}
```

Výpis 6.2: Příklad implementace funkce pro odeslání zprávy.

Funkce pro příjem dat není volána přímo z přiložené knihovny, tudíž rozsah návratových hodnot a implementace jsou čistě na vývojáři. Vývojář však musí zajistit předání znalosti celkové délky přijaté zprávy. Příklad funkce pro příjem dat:

```
uint16_t read_data(char* output)
{
    if (BG77.pollData(SOCK1) == BG77_DATA_READY)
    {
        return BG77.pullData(SOCK1, output);
    }
    return 0;
}
```

Výpis 6.3: Příklad definice funkce pro příjem zprávy.

6.5.2 Implementace knihovny

Nastavení v souboru LWM2M Defines

Prvním krokem pro implementaci knihovny je nastavení chování a podporovaných formátů v souboru LWM2M Defines. Možnosti nastavení jsou:

- **Velikost bufferu pro příjem dat**.
- **Velikost bufferu pro odesílání** - v případě, že je povoleno automatické odesílání, lze tuto položku ponechat na hodnotě 1.
- **Velikost hystereze pro zprávy update** - doba v sekundách, o kterou je odeslána zpráva update dříve.
- **Perioda opakovaného odeslání update zprávy** - doba v sekundách, po které je vyslání zprávy update opakováno.
- **Automatické odesílání** - v případě povolení tohoto nastavení jsou zprávy generované knihovnou automaticky odesílání pomocí poskytnuté callback funkce. Pokud tato položka není povolena, jsou dané zprávy ukládány do bufferu a nastavován příznakový bit.
- **Automatická registrace** - pokud je zařízení ve stavu *Not registered*, dojde k automatické generaci (v případě automatického odesílání i k odeslání) registrační zprávy.
- **Formát pro jednosložkové entity** - Jednotlivé zdroje.
- **Formát pro vícesložkové entity** - Instance, objekty, vícesložkové zdroje.

Příklad nastavení v souboru LWM2M Defines:

```
#define TX_BUFFER_MAX_SIZE 0x01
#define RX_BUFFER_MAX_SIZE 0x04
```

```
#define AUTO_SEND
#define AUTO_REGISTER

#define UPDATE_HYSTERESIS 10
#define UPDATE_RETRY_TIMEOUT 4

#define SINGLE_VALUE_FORMAT FORMAT_PLAIN_TEXT
#define MULTI_VALUE_FORMAT  FORMAT_JSON
```

Výpis 6.4: Příklad nastavení souboru LWM2M Defines.

Vytvoření objektu klienta a manuální bootstrapping

Druhým krokem je samotné vytvoření objektu ze třídy LWM2M Client. Do konstruktoru je nutné předat název klienta a také odkaz na předdefinovanou funkci pro restart zařízení. Délka názvu klienta je momentálně omezena na délku 16 znaků. Příklad inicializace objektu klienta :

```
LWM2M_Client client("RD_DP", reboot);
```

Výpis 6.5: Ukázka inicializace klienta.

Po inicializaci dojde k automatickému vytvoření povinných objektů standardu LWM2M. Pro editaci je využito funkce updateResource. V argumentech této funkce je nutné uvést ID objektu, ID instance, ID zdroje a nová data. Je-li zdroj vícerozměrový, je dalším argumentem index do pole hodnot.

```
void updateResource(obj_id, inst_id, rsrc_id, data, depth = 0);
```

Výpis 6.6: Deklarace funkce pro změnu hodnoty zdroje.

Příklad změny výrobce zařízení:

```
client.updateResource(3,0,0, "RD_Diploma_Thesis");
```

Výpis 6.7: Příklad aktualizace hodnoty zdroje.

Pro přidání zcela nového objektu slouží funkce *createObject*. U přidávání nového objektu je dodržení předpisů pro jednotlivé objekty standardizační autority OMNA zcela v rukou developera.

```
void createObject(obj_id, inst_id);
```

Výpis 6.8: Deklarace funkce vytvoření nového objektu.

Následně je možné do instance objektu přidávat další zdroje funkcí *addResource*. Argumenty funkce *addResource* jsou id objektu a jeho instance, id zdroje, typ zdroje (celé číslo - *TYPE_INT*, desetinné číslo - *TYPE_FLOAT*, logická hodnota -

TYPE_BOOL, řetězec - *TYPE_STRING*, spustitelný zdroj - *TYPE_EXECUTABLE*), přístupová práva ke zdroji (pouze ke čtení - *READ_ONLY*, pouze k zápisu - *WRITE_ONLY*, ke čtení i k zápisu - *READ_WRITE*, spustitelný zdroj - *EXECUTABLE*), logická hodnota vyjadřující, zda se jedná o vícerozměrový zdroj a inicializační hodnota, popřípadě odkaz na funkci v případě spustitelného zdroje.

```
void addResource(o_id, i_id, r_id, type, perm, mv, data);
```

Výpis 6.9: Deklarace funkce pro přidání nového zdroje.

Příklad vytvoření nového objektu a přidání několika zdrojů:

```
client.createObject(4,0);
client.addResource(4,0,0,TYPE_INT,READ_ONLY,false,41);
client.addResource(4,1,1,TYPE_INT,READ_ONLY,true,20);
client.updateResource(4,1,1,34,1);
client.addResource(4,0,2,TYPE_INT,READ_ONLY,false,-70);
```

Výpis 6.10: Příklad vytvoření nového objektu.

Tvorba časování pro knihovnu

Díky modularitě není knihovna schopná sama udržovat čas běhu programu. Pro správnou funkci a časování knihovny je proto nutné knihovnu periodicky informovat o časové změně vůči předchozímu stavu. Knihovna disponuje funkcí *advanceTime*, jejíž argument je celočíselná hodnota vyjadřující uplynulý čas od posledního volání této funkce v sekundách. Tato funkce by měla být volána v pravidelných intervalech, případně by měl být argument upravován tak, aby byl správně reflektován reálný časový posun.

```
void advanceTime(seconds);
```

Výpis 6.11: Deklarace funkce pro časový posun.

Příklad realizace:

```
//each second
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef* htim)
{
    client.advanceTime(1);
}
```

Výpis 6.12: Příklad funkce pro časový posun.

Příjem a odesílání zpráv

Příjem zpráv je řešený přes individuální funkci napsanou developerem pro konkrétní použité rozhraní. Po přijetí zprávy je zapotřebí předat zprávu knihovně k dalšímu zhodnocení. Pro tyto účely slouží funkce *receive*, jejíž argumenty je zpráva samotná, a také délka přijaté zprávy.

```
uint8_t receive(data, data_len);
```

Výpis 6.13: Deklarace funkce pro předání zprávy knihovně.

Příklad pro příjem zprávy:

```
char buffer[500];
uint16_t data_len = read_data(buffer);
if (data_len > 0)
{
    client.receive(buffer, data_len);
}
```

Výpis 6.14: Příklad příjmu zprávy.

Postup pro odesílání dat se může lišit v závislosti na nastavení položky *AUTO SEND* v hlavičkovém souboru LWM2M Defines. Je-li *AUTO SEND* povolen, je zapotřebí před samotným počátkem programu předat knihovně funkci pro odesílání dat. K tomu je využita funkce *register_send_callback*, jejíž argumentem odkaz na funkci pro zasílání.

```
void register_send_callback(send_function);
```

Výpis 6.15: Deklarace registrační funkce pro odesílání.

Příklad předání callback funkce pro automatické odesílání zpráv:

```
client.register_send_callback(send_data);
```

Výpis 6.16: Příklad pro registraci callback funkce pro odesílání.

Pokud položka *AUTO SEND* povolena není, je nutné se knihovny dotazovat, zda jsou v bufferu uložena data k odeslání. Funkce sloužící tomuto účelu je funkce *getTxData*, která vrací hodnotu délky dané zprávy a v jejíž argumentu je nutno předat odkaz na buffer, do kterého bude zpráva předána.

```
uint16_t getTxData(outputBuffer);
```

Výpis 6.17: Deklarace funkce getTxData.

V případě, že v interním bufferu není žádná zpráva k odeslání je navrácena hodnota 0 (také definovaná jako *NO_DATA_AVAILABLE*). Příklad odesílání zpráv při nenastavené položce *AUTO SEND*:

```
char buffer[500];
uint16_t msg_len = client.getTxData(buffer);
if (msg_len != NO_DATA_AVAILABLE)
{
    send_data(buffer,msg_len);
}
```

Výpis 6.18: Příklad odesílání zpráv při nenastavené položce AUTO SEND.

Funkce loop

Funkce loop je hlavní funkcí knihovny. Ve funkci loop jsou postupně spouštěny veškeré funkce potřebné k internímu provozu knihovny (zpracování zpráv, vyhodnocování rutiny update a observe apod.). Při automatické registrace je funkce loop zodpovědná za zaslání úvodní registrační zprávy. Funkci loop je tedy nutné volat ve smyčce, pokud možno, co nejčastěji.

```
void loop();
```

Výpis 6.19: Deklarace funkce loop.

Příklad použití funkce loop v kontextu běhu hlavní smyčky programu:

```
client.register_send_callback(send_data);
char buffer[500];
while(1)
{
    client.loop();
    uint16_t data_len = read_data(buffer);
    if (data_len > 0)
    {
        client.receive(buffer, data_len);
    }
}
```

Výpis 6.20: Příklad použití funkce loop

Zápis a čtení zdrojů

Je-li potřeba zapisovat do již existujícího zdroje, je možné využít dříve zmíněnou funkci *updateResource*.

Pro čtení zdrojů je k dispozici funkce *getResourceValue*. Jako většina funkcí pro manipulaci se zdroji i zde je zapotřebí předat ID objektu, instance a daného zdroje.

Návratová hodnota je řetězec znaků, který je v daném zdroji uložen. Jedná-li se o číselnou, či logickou hodnotu, je zapotřebí hodnotu do tohoto formátu převést.

```
string getResourceValue(o_id, i_id, r_id, depth = 0);
```

Výpis 6.21: Deklarace funkce getResourceValue.

Příklady pro vyčtení hodnoty zdoje:

```
int val = stoi(client.getResourceValue(4,0,0));
```

Výpis 6.22: Příklad vyčítání hodnoty zdroje.

Závěr

Cílem diplomové práce byl popis a porovnání aktuálně využívaných LPWA (Low-Power Wide-Area) technologií v licenčním i bezlicenčním frekvenčním pásmu a běžných přenosových protokolů v těchto systémech používaných. Důraz byl kladen na využití protokolu LWM2M (Lightweight Machine to Machine). Navazujícím výstupem bylo vytvoření programové knihovny pro protokol LWM2M a vytvoření laboratorní úlohy využívající tuto knihovnu pro účely předmětu „Komunikační systémy pro IoT“.

Srovnání přenosových technologií

Z pohledu porovnání komunikačních technologií nelze jednoznačně určit nejvhodnější technologii pro přenos uživatelských dat. Výběr vhodné technologie pro přenos dat je závislý na požadovaných komunikačních parametrech, podmínkách, ve kterých bude zařízení provozováno a požadované výdrži baterie. Obecně jsou technologie LPWA pracující v bezlicenčním pásmu vhodné pro přenos malého objemu dat bez vysoké náročnosti na spolehlivost přenosu. Technologie pracující v licenčním pásmu je naopak vhodné využít v případě nutnosti přenosu zpráv o vyšší velikosti s vyššími nároky na spolehlivost přenosu.

Technologie NB-IoT (Narrowband Internet of Things), pracující v licenčním pásmu, je vhodná pro přenos dat u aplikací s nízkými nároky na přenosovou rychlost a zpoždění přenosu. Díky úzké šířce frekvenčního pásma dosahuje výborného pokrytí a je tak vhodná pro nasazení v aplikacích s menším objemem dat, kde je očekávaná nízká úroveň rádiového signálu (sklepní prostory, odlehlé lokality) a je zapotřebí dlouhá životnost baterie. Technologie NB-IoT, ve vydání 3GPP č.13 (aktuální verze nasazení v České republice), nepodporuje možnost mobility a je tak vhodná pouze pro stacionární zařízení.

Technologii LTE Cat-M dosahuje z pohledu M2M (Machine to Machine) komunikace vysokých přenosových rychlostí a je ji tak vhodné použít u aplikací náročnějších na zpoždění přenosu a objem přenesených dat nebo u aplikací, u kterých je vyžadována mobilita. Díky vyšším přenosovým rychlostem je možné nasazení této technologie také v aplikacích přenášející hlasová data skrze VoLTE (Voice over LTE). Technologie je však současnosti v České republice nasazena pouze experimentálně.

Srovnání přenosových protokolů

Ze srovnání přenosových protokolů pro technologie LPWA v kapitole 5 je patrný vliv zavedené režie na celkový objem přenesených dat. Protokoly založené na spolehlivém přenosu (TCP (Transmission Control Protocol), MQTT (Message Queue Telemetry

Transport)) generují vysoký objem nadbytečných režijních dat, což se u licenčních technologií promítne do cenové kalkulace služeb u poskytovatele konektivity (operátor) a také do výdrže baterie zařízení, jelikož je nutné častěji vysílat/přijímat data.

Protokol UDP (User Datagram Protocol) negeneruje žádná režijní data, ale nepodporuje žádné mechanismy pro zajištění spolehlivosti přenosu. Protokol CoAP (Constrained Application Protocol), založený na protokolu UDP, s protokolem LWM2M (Lightweight M2M) zajišťují spolehlivost přenosu dat na aplikační vrstvě s minimálním množstvím nadbytečných dat.

Volba správného přenosového protokolu závisí na konkrétní implementaci. Protokol CoAP společně s protokolem LWM2M se jeví jako nejvíce univerzální řešení z pohledu poskytnuté přenosové režie, objemu nadbytečných dat (overhead) a možností vzdálené správy zařízení, které umožňují (vzdálený restart zařízení, vynucení určité akce).

Knihovna LWM2M

Knihovna pro protokol CoAP/LWM2M byla sepsána v programovacím jazyce C++. Po dobu vývoje byla knihovna testována vůči open source serveru Leshan a porovnávána s open source klientem Leshan. Výsledkem je, že vytvořená knihovna je plně kompatibilní se serverem Leshan. Knihovna, přiložená k diplomové práci, je již třetí revizí knihovny. Z tohoto důvodu nepodporuje některé nepovinné funkce. Příkladem aktuálně nepodporovaných funkcí je automatické nahrání konfigurace ze serveru (server bootstrap) a zabezpečení DTLS (Datagram Transport Layer Security).

Aktuálně dosahuje knihovna velikostí kolem 80 kB na procesorech s kompletní instrukční sadou CISC (Complex Instruction Set Computing). Podporovanými formáty jsou v momentálním stavu čistý text (Plain text) a JSON (JavaScript Object Notation). V porovnání s řešeními, aktuálně dostupnými na trhu, je současná verze knihovny méně náročná na paměťový prostor, avšak prozatím nedisponuje všemi funkcemi, viz předchozí odstavec. Co se týče přenosu a velikosti zpráv, je vytvořená knihovna srovnatelná s aktuálně dostupnými řešeními. Knihovna a její modularita byla vyzkoušena na třech přenosových technologiích, Ethernet, NB-IoT a LTE Cat-M1. Testování s technologií NB-IoT a LTE Cat-M1 bylo realizováno pomocí komunikačního modulu Quectel BG77, který integruje obě tyto komunikační technologie.

V budoucnu je plánováno doplnit chybějící funkcionalitu do knihovny, knihovnu dále vylepšovat a posouvat dále. Hlavním zaměřením optimalizace bude minimalizace náročnosti na paměťový prostor pro nasazení i na zařízeních s nižšími velikostmi pamětí flash. Dále bude věnována pozornost rozšíření možností konfigurace (ome-

zení některých typů zpráv, větší variabilita z hlediska aplikace), což by nadále mělo snížit výslednou velikost knihovny.

Laboratorní úloha

Vytvořená laboratorní úloha slouží k ilustraci fungování protokolu CoAP s návazností na protokol LWM2M. Laboratorní úloha má za cíl řešitele seznámit s protokolem CoAP a LWM2M na úrovni manipulace jednotlivých bitů a bajtů. Výstupem laboratorní úlohy je důkladná znalost fungování obou protokolů a také srovnání s ostatními protokoly pro technologie LPWA. Po absolvování laboratorní úlohy je student schopný popsat proč jsou protokoly LWM2M společně s protokolem CoAP výhodné řešení k implementaci na bezdrátová zařízení.

Jednotlivé body laboratorního cvičení jsou:

- Seznámení s protokoly využívaných v LPWA přenosech.
- Práce s modulem technologie NB-IoT a Cat-M, Quectel BG77.
- Analýza dat protokolu UDP.
- Analýza dat protokolu TCP.
- Analýza dat protokolu MQTT.
- Manuální sestavení CoAP zprávy.
- Demonstrace knihovny LWM2M vůči open source serveru Leshan.
- Srovnání protokolů.

Bibliografie

[1] SAUTER, Martin. *From GSM to LTE-Advanced Pro and 5G: An Introduction to Mobile Networks and Mobile Broadband*. John Wiley & Sons, 2017.

[2] TELE2IOT. *LTE-M: Licensed Spectrum vs Unlicensed Spectrum* [online]. 2017. Dostupné také z: `http://tele2iot.com/article/lte-m-licensed-spectrum-vs-unlicensed-spectrum/`.

[3] 3GPP. *About 3GPP* [online]. [B.r.]. Dostupné také z: `https://www.3gpp.org/about-3gpp`.

[4] KANJ, Matthieu; SAVAUX, Vincent; LE GUEN, Mathieu. A Tutorial on NB-IoT Physical Layer Design. *IEEE Communications Surveys & Tutorials*. 2020.

[5] SUDONULL. *NB-IoT, Narrow Band Internet of Things. Power Saving Modes and Control Commands* [online]. [B.r.]. Dostupné také z: `https://sudonull.com/post/3742-NB-IoT-Narrow-Band-Internet-of-Things-Power-saving-modes-and-control-commands`.

[6] MASEK, Pavel; STUSEK, Martin; FUJDIAK, Radek; MLYNEK, Petr; HOSEK, Jiri. Komunikační systémy pro IoT. 2019.

[7] CHANG, Xiangmao; CHEN, Kai; XING, Guoliang; HUANG, Jun. Optimizing NB-IoT Power Consumption via Adaptive Radio Access. *IEEE Internet of Things Journal*. 2021.

[8] MOZNY, Radek; MASEK, Pavel; STUSEK, Martin; ZEMAN, Krystof; OMETOV, Aleksandr; HOSEK, Jiri. On the Performance of Narrow-band Internet of Things (NB-IoT) for Delay-tolerant Services. In: *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*. 2019, s. 637–642.

[9] ASSOCIATION, GSM et al. NB-IoT Deployment Guide to Basic Feature set Requirements. *En ligne*. 2019. Dostupné také z: `https://www.gsma.com/iot/wp-content/uploads/2018/04/NB-IoT_Deployment_Guide_v2_5Apr2018.pdf`.

[10] MASEK, Pavel; STUSEK, Martin; ZEMAN, Krystof; DRAPELA, Roman; OMETOV, Aleksandr; HOSEK, Jiri. Implementation of 3GPP LTE Cat-M1 Technology in NS-3: System Simulation and Performance. In: *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. 2019, s. 1–7.

[11] QUALCOMM. *Leading the LTE IoT Evolution to Connect the Massive Internet of Things*. [B.r.].

[12] SEMICONDUCTORS, Altair. *Coverage Analysis of LTE-M Category-M1* [online]. 2017. Dostupné také z: https://www.altair-semi.com/wp-content/uploads/2017/02/Coverage-Analysis-of-LTE-CAT-M1-White-Paper.pdf.

[13] HSIEH, Ping-Chun; JIA, Yupeng; PARRA, Darwin; AITHAL, Prabha. An Experimental Study on Coverage Enhancement of LTE Cat-M1 for Machine-type Communication. In: *2018 IEEE International Conference on Communications (ICC)*. 2018, s. 1–5.

[14] ASSOCIATION, GSM et al. *LTE-M Deployment Guide to Basic Feature Set Requirements*. London, UK, 2018.

[15] SIGFOX. *What is Sigfox* [online]. [B.r.]. Dostupné také z: https://build.sigfox.com/sigfox.

[16] LORAALLIANCE. *About LoRaWAN* [online]. [B.r.]. Dostupné také z: https://lora-alliance.org/about-lorawan/.

[17] JEŘÁBEK, Jan. *Pokročilé komunikační techniky*. Vysoké učení technické v Brně, 2021.

[18] *MQTT V3.1 Protocol Specification* [online]. 2010. Dostupné také z: http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/MQTT_V3.1_Protocol_Specific.pdf.

[19] *Mosquitto Manual* [online]. [B.r.]. Dostupné také z: https://mosquitto.org/man/mqtt-7.html.

[20] HW-GROUP. *MQTT - univerzální protokol pro cloudové a IoT aplikace* [online]. [B.r.]. Dostupné také z: https://www.hw-group.com/cs/podpora/mqtt-univerzalni-protokol-pro-cloudove-a-iot-aplikace.

[21] *MQTT for Sensor Networks Version 2.0* [online]. 2021. Dostupné také z: https://www.oasis-open.org/committees/download.php/68448/mqtt-sn-v2.0-wd08.docx.

[22] SADIO, Ousmane; NGOM, Ibrahima; LISHOU, Claude. Lightweight Security Scheme for MQTT/MQTT-SN Protocol. In: *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. 2019, s. 119–123.

[23] *RFC7252 - The Constrained Application Protokol (CoAP)* [online]. 2014. Dostupné také z: https://datatracker.ietf.org/doc/html/rfc7252.

[24] *RFC7641 - Observing Resources in the Constrained Application Protocol (CoAP)* [online]. 2015. Dostupné také z: https://datatracker.ietf.org/doc/html/rfc7641.

[25] *RFC8613 - Object Security for Constrained RESTful Environments (OSCORE)* [online]. 2019. Dostupné také z: https://datatracker.ietf.org/doc/html/rfc8613.

[26] PALOMBINI, Francesca; SELANDER, Göran; MATTSSON, John Preuß. *OSCORE: bA Look at the New IoT Security Protocol* [online]. 2019. Dostupné také z: https://www.ericsson.com/en/blog/2019/11/oscore-iot-security-protocol.

[27] ALLIANCE, Open Mobile. *Lightweight Machine to Machine Technical Specification Version 1.2*. 2020. Č. 1.

[28] BASU, Subho Shankar; HAXHIBEQIRI, Jetmir; BAERT, Mathias; MOONS, Bart; KARAAGAC, Abdulkadir; CROMBEZ, Pieter; CAMERLYNCK, Pieterjan; HOEBEKE, Jeroen. An End-to-End LWM2M-based Communication Architecture for Multimodal NB-IoT/BLE Devices. *Sensors*. 2020, roč. 20, č. 8, s. 2239.

[29] AVSYSTEM. *OMA LwM2M - Brief Description* [online]. [B.r.]. Dostupné také z: https://avsystem.github.io/Anjay-doc/LwM2M.html.

[30] ALLIANCE, Open Mobile. *Lightweight Machine to Machine Technical Specification Version 1.0, Approved Version* [online]. 2017. Dostupné také z: http://www.openmobilealliance.org/release/lightweightm2m/V1_0-20170208-A/OMA-TS-LightweightM2M-V1_0-20170208-A.pdf.

[31] COMMITTEE, Embedded C++ Technical. Rationale for the Embedded C++ Specification [online]. 1998. Dostupné také z: http://www.caravan.net/ec2plus/rationale.html.

[32] PLAUGER, Philip J. Embedded C++: An Overview. *Embedded Systems Programming*. 1997, roč. 10, s. 40–53.

[33] TECHPLAYON. *Maximum Coupling Loss (MCL) and Maximum Path Loss (MPL)* [online]. 2018. Dostupné také z: https://www.techplayon.com/maximum-coupling-loss-mcl-and-maximum-path-loss-mpl/.

[34] CHALAPATI, S. Comparison Of LPWA Technologies And Realizable Use Cases. 2018.

[35] STUSEK, Martin; ZEMAN, Krystof; MASEK, Pavel; SEDOVA, Jindriska; HOSEK, Jiri. IoT Protocols for Low-power Massive IoT: A Communication Perspective. In: *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. 2019, s. 1–7.

[36] HIVEMQ. *Payload Encryption - MQTT (Message Queue Telemetry Transpor) Security Fundamentals* [online]. 2015. Dostupné také z: https://www.hivemq.com/blog/mqtt-security-fundamentals-payload-encryption/.

[37] BISNE, Lochan; PARMAR, Manish. Composite Secure MQTT for Internet of Things Using ABE and Dynamic S-box AES. In: *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*. 2017, s. 1–5.

[38] AVILA, Risto. C++ for Embedded: Advantages, Disadvantaged, and Myths. *Embedded Development Talk*. 2022. Dostupné také z: https://www.qt.io/embedded-development-talk/c-for-embedded-advantages-disadvantages-and-myths.

Seznam symbolů a zkratek

16QAM	16 Quadrature Amplitude Modulation
3GPP	Third Generation Partnership Project
AEAD	Authenticated Encryption and Associated Data
AES	Advanced Encryption Standard
ACK	Acknowledge
API	Application Programming Interface
BPSK	Binary Phase Shift Keying
CE	Coverage Enhancement
CIoT	Cellular Internet of Things
CISC	Complete Instruction Set Computer/Computing
CoAP	Constraint Application Protocol
CON	Confirmable
CSS	Chirp Spread Spectrum
DBPSK	Differential BPSK
DCCP	Datagram Congestion Control Protocol
DTLS	Datagram Transport Layer Security
E-CID	Enhanced Cell Identity
eDRX	extended idle-mode Discontinuous Reception
EC++	Embedded C++
ECL	Extended Coverage Level
eNodeB / eNB	evolved Node B
FD-FDD	Full Duplex-FDD
FDD	Frequency Division Duplex
GFSK	Gaussian Frequency Shift Keying

GPRS	General Packet Radio Service
GSM	Groupe Spécial Mobile
GW	Gateway
HD-FDD	Half Duplex - Frequency Division Duplex
HTTP	HyperText Transfer Protocol
IETF	Internet Engineering Task Force
IoT	Internet of Things
ISM	Industrial, Scientific and Medical
ITU	International Telecommunication Union
JSON	JavaScript Object Notation
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
LPP	Location and Positioning Protocol
LPWA	Low-Power Wide Area
LTE	Long Term Evolution
LTE-M	LTE-Machine
LWM2M	Lightweight Machine to Machine
M2M	Machine to Machine
MCL	Maximum Coupling Loss
MIMO	Multiple Input Multiple Output
mMTC	massive Machine Type Communication
MPDCCH	MTC Physical Downlink Control Channel
MQTT	Message Queuing Telemetry Transport
MQTT-SN	MQTT for Sensor Networks
NB-IoT	Narrowband Internet of Things

NON	Non-confirmable
NPDCCH	Narrowband Physical Downlink Control Channel
OFDM	Orthogonal Frequency Division Multiplex
OFDMA	Orthogonal Frequency Division Multiple Access
OMA	Open Mobile Alliance
OMNA	Open Mobile Naming Authority
OSCORE	Object Security for Constrained RESTful Environments
OTDOA	Observed Time Difference of Arrival
PDSCH	Physical Downlink Shared Channel
PRB	Physical Resource Block
PSM	Power Saving Mode
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RAI	Release Assistance Indication
RAM	Random Access Memory
RAU	Routing Area Update
REST	Representational State Transfer
RFTDMA	Random Frequency and Time Division Multiple Access
ROM	Read Only Memory
RRC	Radio Resource Control
RSVP	Resource Reservation Protocol
RX	Receive
SC-FDMA	Single Carrier - Frequency Division Multiple Access
SCTP	Stream Control Transmission Protocol
SF	Spreading Factor

SINR	Signal to Interference plus Noise Ratio
SISO	Single Input Single Output
SNR	Signal to Noise Ratio
TAU	Tracking Area Update
TCP	Transmission Control Protocol
TDD	Time Division Duplex
TLV	Type Length Value
ToA	Time on Air
TLS	Transport Layer Security
TX	Transmit
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
UTF-8	Unicode Transformation Format
VoLTE	Voice over LTE
WuS	Wake up Signal

A Obsah elektronické přílohy

- /
  - LWM2M_Lib.zip ..... knihovna protokolu LWM2M
    - inc
      - CoAP.hpp
      - json.h
      - Logger_xdvora2g.h
      - LWM2M_Client.h
      - LWM2M_Defines.h
      - LWM2M_Object.h
      - LWM2M_Resource.h
      - Utils.h
    - src
      - CoAP.cpp
      - json.cpp
      - Logger_xdvora2g.cpp
      - LWM2M_Client.cpp
      - LWM2M_Defines.cpp
      - LWM2M_Object.cpp
      - LWM2M_Resource.cpp
      - Utils.cpp
  - lab.zip ..... laboratorní úloha
    - lab_LWM2M.pdf
    - lib
      - BG77.h
      - BG77.cpp
      - SerialComm.h
      - SerialComm.cpp
    - TCP-UDP-Server.py