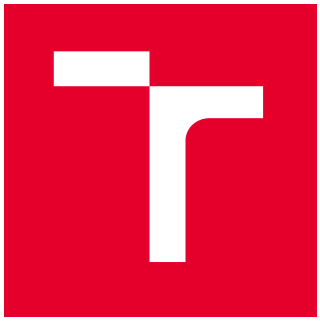


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

MOBILNÍ APLIKACE PRO SYSTÉM MONITORUJÍCÍ OPTICKÁ VLÁKNA

SMARTPHONE APPLICATION FOR OPTICAL FIBER MONITORING SYSTEM

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jan Vaverka**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Dejdar**

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. Jan Vaverka ***ID:*** 205963

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

NÁZEV TÉMATU:

Mobilní aplikace pro systém monitorující optická vlákna

POKYNY PRO VYPRACOVÁNÍ:

Cílem diplomové práce je teoretický rozbor zabezpečení fyzické vrstvy optické vláknové infrastruktury. V rámci rozboru bude analyzován systém pro kontinuální sledování vlastností optických vláken a kabelů. Bude navržen způsob komunikace mezi mobilní aplikací a serverem ovládajícím celý systém. Na základě analýzy bude následně naprogramována mobilní aplikace pro OS Android a připravena mobilní aplikace pro iOS. Součástí práce bude také podrobná analýza současného stavu v oblasti zabezpečení optických vláken. V rámci diplomové práce bude také provedeno testování naprogramované aplikace a odzkoušeny všechny funkcionality.

DOPORUČENÁ LITERATURA:

[1] FILKA, Miloslav. Optoelektronika: Pro telekomunikace a informatiku. Vyd. 1. Brno :Centa, 2009. 369 s. ISBN 978-80-86785-14-1.

[2] STROBEL, Otto. Optical and Microwave Technologies for Telecommunication Networks. John Wiley & Sons, 2016.

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

Vedoucí práce: Ing. Petr Dejdar

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

ABSTRAKT

Hlavním cílem této diplomové práce je vývoj multiplatformní mobilní aplikace ve frameworku Flutter pro platformy iOS a Adnroid. Aplikace bude vzdáleně komunikovat s analyzátorem optických vláken skrz server. V úvodu práce je popsána problematika optických vláken a možnosti zabezpečení. V další části jsou popsány možnosti komunikace mezi aplikací a serverem, rovněž je zde popsán framework Flutter. V praktické častí je navrhnuta komunikace mezi aplikací a serverem. Velká část diplomové práce se věnuje návrhu a vývoji multiplatformní aplikace. Poslední část se věnuje testování komunikace mezi aplikací a serverem.

KLÍČOVÁ SLOVA

Flutter, Mobilní aplikace, monitorující systém, multiplatformní vývoj, optické vlákno

ABSTRACT

The main goal of this master thesis is to develop a multi-platform mobile application in the Flutter framework for iOS and Adnroid platforms. The application will remotely communicate with a fiber optic monitoring system through a server. In the introduction of the thesis, the issues of fiber optics and fiber optic security options are described. The next section describes the communication options between the application and the server, also the Flutter framework is described. In the practical part, the communication between the application and the server is proposed. A large part of the practical section is focused on the design and programming of the multiplatform application. The last part is focused on testing the communication between the application and the server.

KEYWORDS

Flutter, Mobile app, monitoring system, multiplatform development, optical fiber

VAVERKA, Jan. *Mobilní aplikace pro systém monitorující optická vlákna*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 99 s. Diplomová práce. Vedoucí práce: Ing. Petr Dejdar

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Jan Vaverka
VUT ID autora:	205963
Typ práce:	Diplomová práce
Akademický rok:	2021/22
Téma závěrečné práce:	Mobilní aplikace pro systém monitorující optická vlákna

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

Děkuji vedoucímu diplomové práce panu Ing. Petru Dejdarovi, za účinnou metodickou, pedagogickou a odbornou pomoc při zpracování mé diplomové práce.

Obsah

Seznam obrázků

Úvod

Tato diplomová práce má za cíl vytvořit a otestovat multiplatformní aplikaci pro platformy Android a iOS. Tato aplikace bude zobrazovat naměřená data a aktuální stav analyzátoru monitorující optická vlákna. Rovněž je nutné navrhnout komunikaci mezi zmíněnou aplikaci a serverem, kde budou data z analyzátoru uložena.

V teoretické části diplomové práce je podrobný rozbor optických vláken, jejich rozdělení, principů přenosů a hlavních parametrů. V teorii je také rozebrána problematika zabezpečení optických vláken a to jak na lokálních spojích, tak i na těch dálkových.

První kapitola se věnuje popisu a rozboru optických vláken. Jsou zde popsány jednotlivé druhy optických vláken a jejich využití. Dále je zde popsán princip šíření signálu v optických vláknech. Velká část této kapitoly se věnuje nejdůležitějším vlastnostem a parametrům, které se v optických vláknech vyskytují. V první kapitole je rovněž popsán princip měření útlumu pomocí OTDR.

Druhá kapitola se věnuje zabezpečení optických vláken. Jsou zde popsány principy odposlouchávání optických vláken. Velká část se zabývá automatizovanými systémy správy infrastruktury, kde je popsán jejich princip a jednotlivé techniky monitorování. Dále je zde popsán speciální hardware pro monitorování optických vláken od firmy Viavi, který využívá kontinuální měření pomocí OTDR.

Třetí kapitola je věnovaná teoretickému rozboru komunikačních technik mezi klientem a serverem. Je zde popsán HTTP protokol a jeho varianty, ale i méně známý protokol WebSocket, který nám přináší oproti HTTP určité výhody. Rovněž jsou zde vysvětleny různé druhy formátu dat, které lze pro přenos dat mezi klientem a serverem využívat.

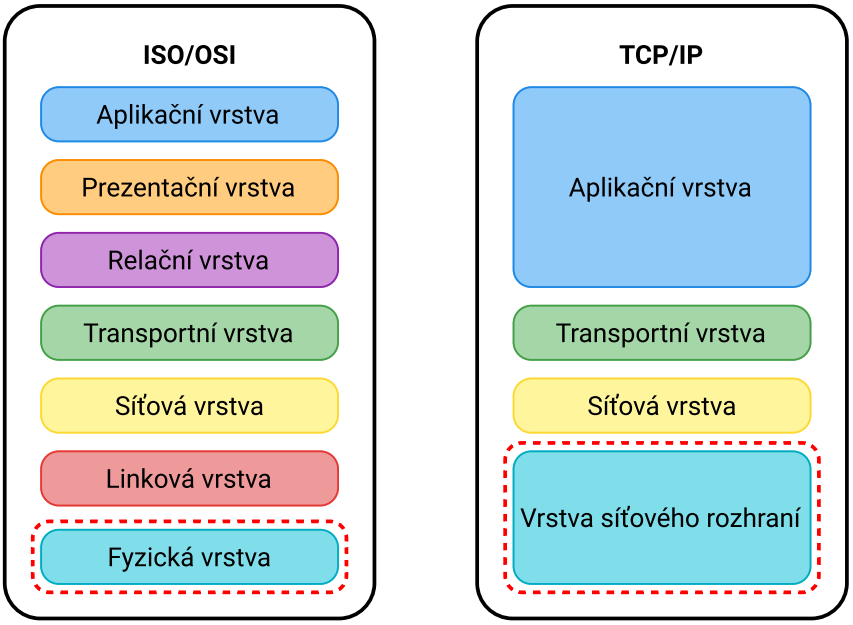
Na teoretický rozbor komunikačních protokolů a technik navazuje čtvrtá kapitola. Zde jsou tyto poznatky implementovány do návrhu komunikace mezi mobilní aplikaci a serverem. Pátá a šestá kapitola se snaží uvést čtenáře do problematiky multiplatformního vývoje, pomocí nástrojů Flutter a Firebase. Jsou zde popsány a ukázány základní struktury a problematiky vývoje mobilní aplikace.

V sedmé kapitole je popsána finální multiplatformní aplikace. V první části je popsaná celková architektura aplikace a struktura projektu. Dále je zde popis jednotlivých funkcionalit i jejich implementace v kódu. Poslední osmá kapitola diplomové práce se věnuje testování mobilní aplikace na obou platformách a testování komunikace mezi aplikací a serverem.

1 Optická vlákna

V dnešní době jsou optické vlákna čím dále více využívaná a se zrychlujícím tempem nahrazují klasické metalické média. Největší využití mají optické vlákna v takzvaných páteřních spojích. Přes tyhle spoje proudí největší množství dat a musí být k tomu tak uzpůsobena. Právě na těchto spojích byly jako první nahrazeny metalické spoje optickými. Jelikož v dnešní době je čím dál větší poptávka pro rychlejších datových přenosech, tak se optické vlákna stále častěji dostávají až ke koncovým uživatelům [1].

Přenosová média spadají pod první vrstvu ISO/OSI a TCP/IP modelu, jak je znázorněno na obrázku 1.1. Právě tato vrstva předepisuje jaké vlastnosti mají přenosová media (optická vlákna) mít. Určuje např. maximální délku kabelu, rychlosti přenosu, tvar konektoru a v neposlední řadě i charakteristiku signálu [1].



Obr. 1.1: Referenční model ISO/OSI a TCP/IP [2].

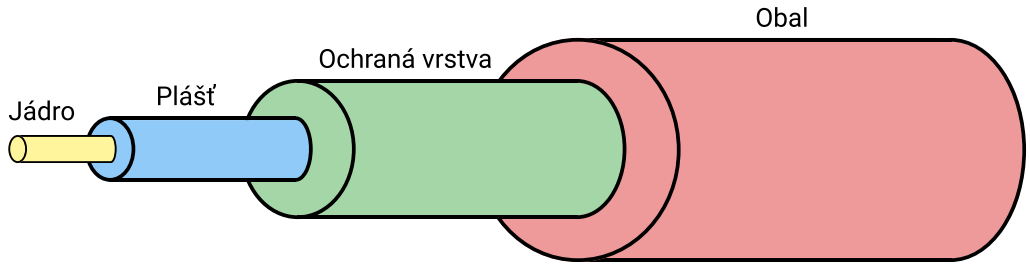
1.1 Struktura optického vlákna

Optické vlákno je v podstatě válcové dielektrikum zhotovené z vhodného materiálu. Na výrobu optických vláken se používá velká paleta materiálů nejpoužívanější jsou, ale sklo, plasty a syntetický křemen. Do těchto základních materiálu se velmi často přidávají příměsi jako je oxid boritý (B_2O_3), oxid germaničitý (GeO_2) nebo oxid

fosforečný (P_2O_5). Tyto příměsi mají hlavně vliv na index lomu jádra a pláště vlákna, čímž se se ovlivňují jeho celkové přenosové vlastnosti [3].

Optické vlákno se skládá z několika válcových struktur, jak je vidět na obrázku 1.2. Úplně ve středu je jádro, ve kterém se šíří světelný paprsek. Nad jádrem se nachází plášť, zde je důležité, aby jádro a plášť měl každý jinou hodnotu indexu lomu. Kdyby tomu tak nebylo, paprsek by se v jádru nešířil, tento problém je dále vysvětlený v sekci 1.2. Nad pláštěm se nachází ještě další dvě vrstvy. Tyto vrstvy nemají žádný vliv na šíření paprsku ve vláknu, slouží pouze ke zpevnění a ochraně vlákna před poškozením.

Z jednotlivých vláken se následně smotává optický kabel. V jednom kabelu mohou být desítky až stovky vláken. V kabelu se dále mohou nacházet tahové prvky, které zlepšují odolnost kabelu v tahu v příčném směru a další ochranné, nebo stínící prvky.



Obr. 1.2: Struktura optického vlákna [1].

1.2 Princip šíření paprsku v optickém vláknu

Hlavním úkolem optického vlákna je přenášet světelný paprsek od zdroje až po přijímač na druhé straně vlákna a to s co nejmenšími ztrátami. Paprsek se při dopadu na rozhraní dvou materiálu v našem případe jádro–plášť (n_1, n_2) částečně láme do druhého prostředí (n_2) a částečně se odráží zpět do původního prostředí (n_1).

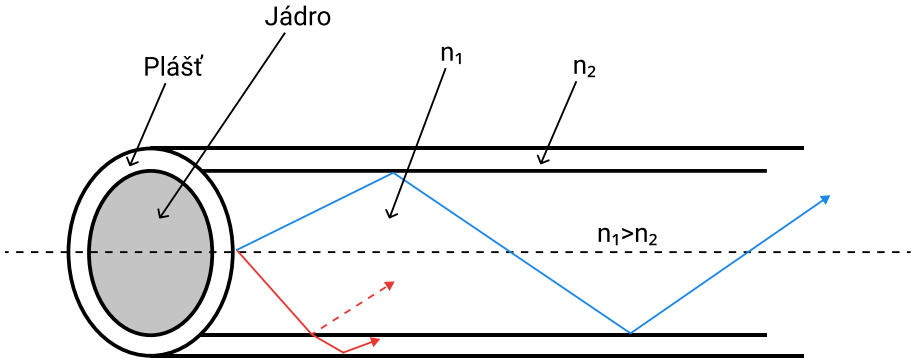
Index lomu můžeme lehce vypočítat pomocí vzorce 1.1 kde v_1 a v_2 jsou rychlosti šíření světla v určitých prostředích. Tato bezrozměrná veličina (vždy větší než jedna) vyjadřuje, kolikrát je rychlost světla v daném prostředí menší než rychlost světla ve vakuu [4].

$$n = \frac{v_1}{v_2} \tag{1.1}$$

1.2.1 Totální odraz

V optickém vláknu nejvíce využíváme totálního odrazu, který nastává při šíření paprsku z prostředí hustšího do opticky řidšího prostředí. To znamená že pokud je splněná podmínka ($n_1 > n_2$) a úhel lomu je větší než 90° (mezní úhel) tak dochází k totálnímu odrazu paprsku. To má za důsledek to, že se paprsek vůbec nedostane do druhého prostředí (plášť) a zůstává pouze v původního prostředí (jádro) [4].

Celá situace je znázorněna na obrázku 1.3, kde modře je znázorněn paprsek který se láme pod větším uhlem než je mezní uhel, tudíž dochází k totálnímu odrazu. Naopak červeně je znázorněna situace, kde k totálnímu odrazu nedochází. To má za následek, že část paprsku přejde do pláště a tím postupně červený paprsek zanikne.



Obr. 1.3: Princip totálního odrazu v optickém vláknu [5].

1.3 Základní dělení optických vláken

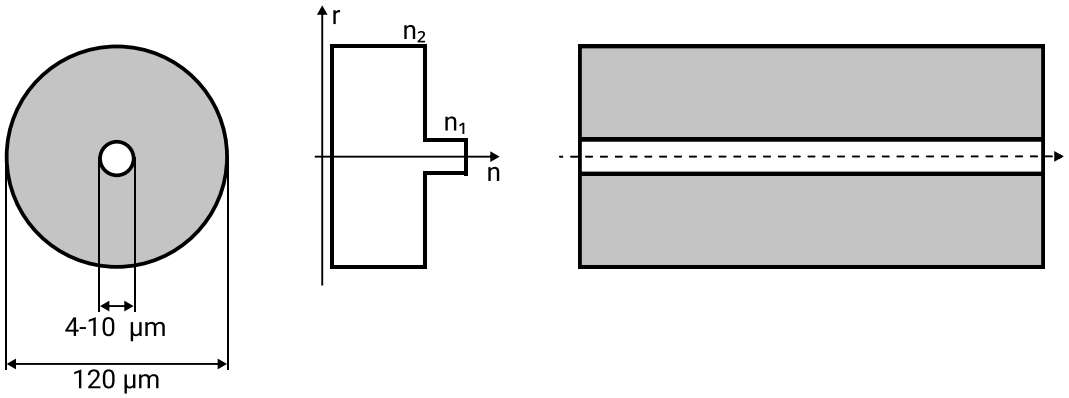
Optická vlákna se dělí na několik typů. Nejdůležitější dělení je na jednovidová a mnohovidová vlákna. Každý z těchto typů má určité parametry které určují jeho rychlost a maximální vzdálenost na kterou se může vlákno použit, aniž by bylo potřeba použít opakovač. Rovněž každý typ optického vlákna potřebuje specifický vysílač a přijímač.

1.3.1 Jednovidová vlákna

U jednovidového vlákna (SMF) se průměr jádra u těchto vláken je velmi malý pohybuje se v rozmezí 4 až 10 μm a umožňuji přenos pouze jediného takzvaného základního vidu elektromagnetické vlny. Vid se ve vlákně šíří přímo bez jakéhokoliv odrazu. Musí být zde splněna podmínka V $<$ 2,405 kde V je normalizovaná frekvence [1]. NA těchto vláken se pohybuje v rozmezí 0,08 až 0,15. Nevýhodou těchto vláken je,

že pro jejich buzení je zapotřebí úzká spektrální čára, kterou dokáže vytvořit pouze laser, nebo laserové diody [6]. Struktura jednovidového vlákna je na obrázku 1.4.

Tyto vlákna jsou nejčastěji používané na delší přenosy v páteřní sítí jelikož, dosahuji na rozdíl od mnohovidových vláken nižších hodnot disperze a útlumu, zároveň mají největší šířku pásma.



Obr. 1.4: Jednovidové optické vlákno [1].

1.3.2 Mnohovidová vlákna

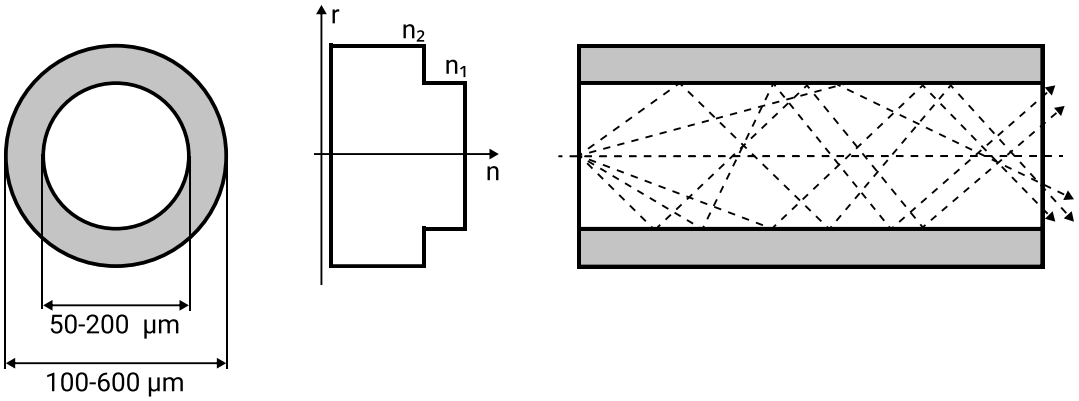
Mnohovidová vlákna můžeme ještě dále rozdělit na dva typy a to podle typu změny indexu lomu ve vlákně.

Mnohovidové vlákno se skokovou změnou indexu lomu

Mnohovidová vlákna se skokovou změnou indexu lomu (MMF SI) jsou na výrobu nejjednodušší, průměr jádra se pohybuje od 50 do 200 μm. Na rozdíl od jednovidového vlákna se zde šíří skupina až tisíce vidů které do vlákna vstupují pod různými úhly a využívá se zde totálního odrazu na rozhraní jádro–plášť ($n_1 > n_2$) jak je vidět na obrázku 1.5. Problém těchto vláken je, že u nich vzniká vidová disperze a to omezuje celkovou šířku pásma. Je zde splněná podmínka V > 2,405. NA se pohybuje v rozmezí 0,3 až 0,6. Tyto vlákna se používají na kratší vzdálenosti než jednovidová vlákna a využívá se vlnová délka 850 nm. Pro buzení optického vlákna se využívají levnější luminiscenční diody [5] [6].

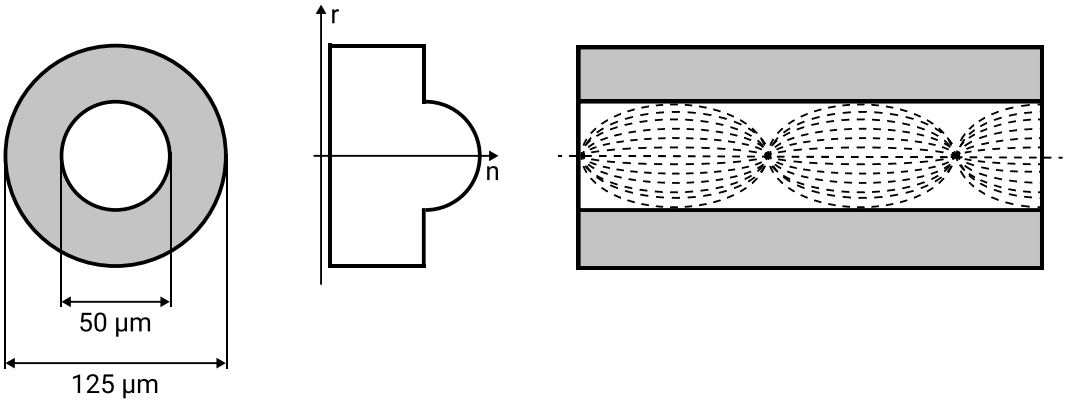
Mnohovidové vlákno s gradientní změnou indexu lomu

Gradientní optické vlákno (MMF GI) je speciálně upravené vlákno které vidy ohýbá do tvaru kvadratické paraboly jak je vidět na obrázku 1.6. Tyto vlákna mají průměr jádra 50 μm a jsou složitější a dražší na výrobu než vlákna se skokovou změnou



Obr. 1.5: Mnohovidové optické vlákno se skokovou změnou indexu lomu [1].

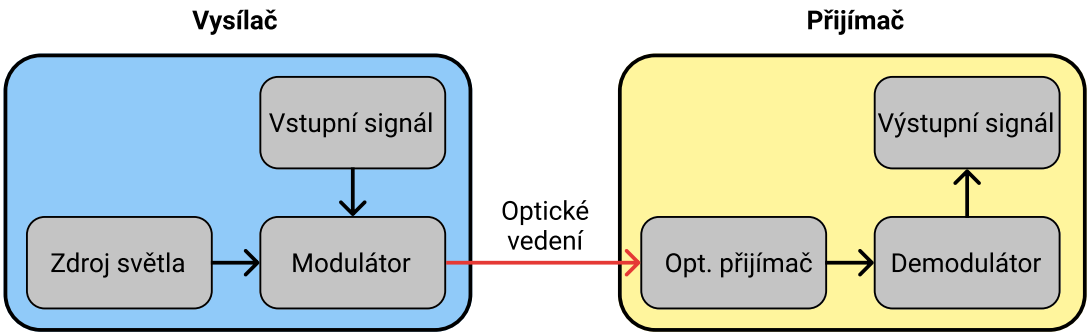
indexu lomu. NA se pohybuje kolem 0,2. Na druhou stranu se vyznačují lepšími parametry které umožňuji podstatně snížit vidovou disperzi a rovněž útlum. Tyto vlákna využívají vlnové délky 850 a 1300 nm a využívají se na středně dlouhé přenosy [6].



Obr. 1.6: Mnohovidové optické vlákno s gradientní změnou indexu lomu [1].

1.4 Vysílaní a přijímaní dat

Samotné optické vlákno by nám samozřejmě pro přenos dat nestačilo, optické vlákno zde zastupuje jenom cestu kudy se data, nebo zpráva šíří. Nejdříve je tedy nutné data změnit na světelné paprsky, přenést je pomocí optického vlákna a následně ze světla opět poskládat původní data. Celou optickou trasu si můžeme rozdělit tedy na tři části, které si níže popíšeme. Zjednodušené schéma optické trasy můžeme vidět na obrázku 1.7.



Obr. 1.7: Zjednodušené schéma optické trasy [3].

1.4.1 Optický vysílač

První části je optický vysílač, jedná se o zařízení, které převádí elektrický signál na signál optický, který následně vstupuje do optického vlákna. Jako zdroj světla se využívají laserové nebo luminiscenční diody. Velkou výhodou laserových diod je úzká spektrální čára, vysoká přenosová rychlost a vysoký emitovaný výkon, který se pohybuje až v desítkách mW. Zdroje světla využívají jednu ze tří vlnových délek, které se v optických vláknech nejlépe šíří (850 nm, 1310 nm a 1550 nm) [7].

Další důležitou práci optického vysílače je modulace signálu. Modulace je proces, kdy se pomocí modulujícího signálu mění charakter nosného signálu. Dalo by se to říct také tak, že se nosnému signálu vtiskne informační obsah, který chceme přenést. Existují dva základní druhy modulace a to analogová modulace, kde se signál mění spojitě v čase. A druhá modulace je digitální, kde je signál vyjádřen diskrétně ve formě bitové posloupnosti [8].

Typy analogových modulací

- Fázová modulace (PM)
- Amplitudová modulace (AM)
- Frekvenční modulace (FM)

Typy digitálních modulací

- Amplitudové klíčování (ASK)
- Fázové klíčování (PSK)
- Frekvenční klíčování (FSK)
- Kvadraturní amplitudová modulace (QAM)

1.4.2 Optická trasa

Optickou trasou propojíme dvě místa mezi sebou. Nejdůležitějšími parametry optické trasy jsou nízký útlum a zkreslení. Tyto parametry ovlivňuje spousta proměnných. Od typu modulace přes typ optického vlákna až k tomu, jak je celkově optický kabel ohnutý a uložený v zemi. Dalším důležitým parametrem, který ovlivňuje útlum je kolik se na trase nachází spojek a konektorů [7].

Na optické trase se vyskytuji i další prvky a to jak pasivní tak i aktivní. Mezi typické pasivní prvky patří spojky a konektory. Optické spojky používáme na propojení dvou optických kabelu mezi sebou a konektory se nejčastěji používají na propojení optického kabelu a aktivních zařízení. Dalším pasivním prvkem, který můžeme na trase využít je optický rozbočovač. Rozbočovač funguje tak, že na vstupu rozdělí signál a na výstupu jej vyšle např. do dvou optických kabelů. Nevýhodou je, že se přidáním každého rozbočovače na trase zvětší celkový útlum [7].

Jelikož ani optický kabel nemůže být neomezeně dlouhý, protože v něm působí parazitní jevy jako útlum a disperze viz kapitola 1.5, je potřeba signál po určité vzdálenosti vyčistit a zesílit. Pro tento účel slouží opakovač, řadíme jej do aktivních prvků, které se na trase nacházejí. Opakovač funguje tak, že přijme zkreslený a zašuměný signál, následně jej převede na elektrický signál, který je zesílen a tento elektricky signál je opět převedený na optický signál, který je dále vyslán do optického vlákna [7, 9].

Využití optických tras má spoustu výhod. Celkově je optický trasa velice imunní proti přeslechům a interferencím. To znamená, že není optické vlákno ovlivňováno elektromagnetickým rušením z různých zdrojů. Jelikož se světelný signál nachází neustále v jádru, tak nedochází k přeslechům mezi jednotlivými vlákny. Jelikož je optické vlákno vyrobeno pouze z dielektrik, to znamená že jsou to izolanty, tak je lze využít v elektrických nebezpečných prostředích.

1.4.3 Optický přijímač

Optický přijímač má za úkol detekovat výkon světla, který se k němu dostane a následně jej převést na odpovídající elektrickou úroveň. Základní části každého optického přijímače je fotodetektor, nejčastěji se jedná o fotodiodu lavinového typu. Jakmile fotodioda zachytí signál, tak je potřeba signál ještě demodulovat a dostat z něho potřebné informace, které se v něm přenášely [7, 10].

1.5 Vlastnosti a parametry optických vláken

V následující sekci jsou popsány a vysvětleny nejdůležitější parametry optických vláken.

1.5.1 Útlum

Útlum je měřítkem ztrát optické energie ve vlákně. Stejně jako u metalických medií tak i u optických nám výkon se vzdáleností, kterou urazí postupně klesá. Útlum je závislý na vlnové délce záření. Jedná se o poměr výkonu vyslaného (P_1) a přijatého (P_2), nejčastěji vyjádřen v logaritmickém měřítku, jak je vidět ve vzorci 1.2, tento útlum se udává v (dB) [11].

$$\alpha = 10 \log(\frac{P_1}{P_2}) \tag{1.2}$$

Samozřejmě, že je útlum závislý na vzdálenosti vlákna a proto existuje hodnota vztažená k jednotce kilometrů (dB/km). Útlum je v optických vláknech způsobován hlavně rozptylem viz 1.5.5 a absorpcí prostředí viz 1.5.7.

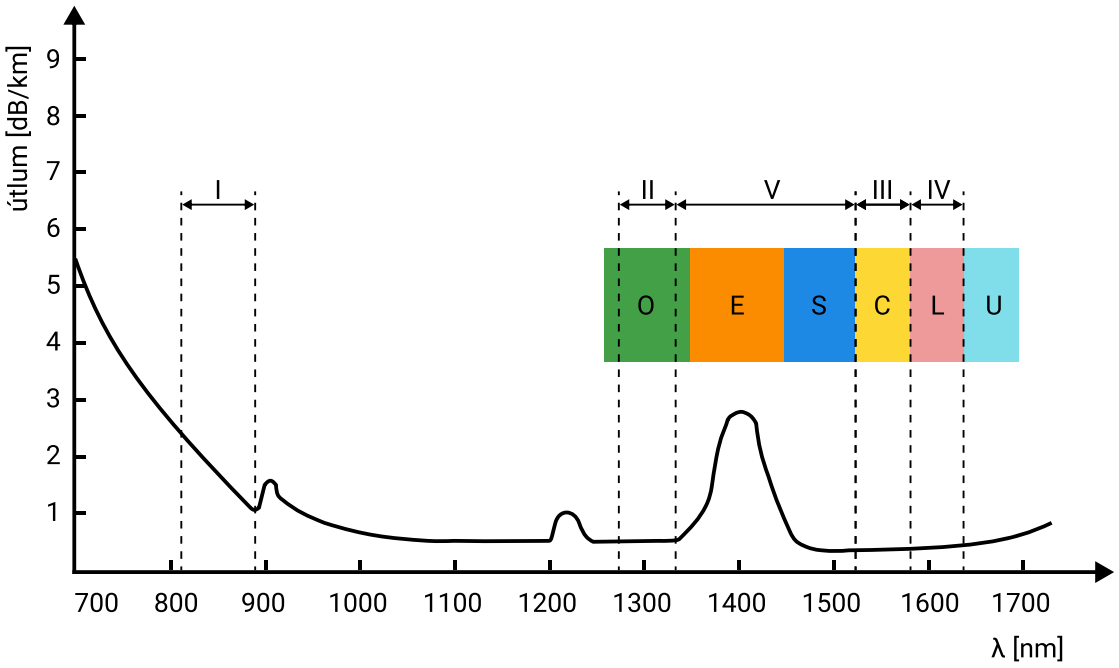
Důležité je rovněž brát v potaz útlum v závislosti na vlnové délce zdroje záření. Tato charakteristika je vykreslená na obrázku 1.8, na obrázku jsou zvýrazněné jednotlivá přenosová okna (I–IV), která se využívají pro přenos v optických sítích:

- I. Okno: 840–940 nm – I. okno se využívá výhradně pro mnohovidové vlákna na krátké vzdálenosti řádově stovky metrů. Nejčastěji se využívají v LAN sítích, jelikož se zde využívají levné zdroje záření.
- II. Okno: 1280–1335 nm – Toto okno se využívá pro dálkové přenosy pomocí jednovidových vláken. Hodnota měrného útlumu je zde 0,35 dB/km.
- III. Okno: 1530–1565 nm – Měrný útlum zde dosahuje minima 0,19–0,22 dB/km, pro jednovidová křemíkové vlákna, využívá se v transportních a globálních sítích.
- IV. Okno: 1565–1625 nm – Parametry IV. okna se moc neliší od III. okna, jelikož je v téhle části útlumová křivka velice plochá. Když spojíme III. a IV okno dohromady tak lze zde využít techniku WDM viz 1.6.1, pro vícenásobné využití optického vlákna.
- V. Okno: 1335–1530 nm – Toto okno bylo přidáno jako poslední jelikož se nachází přímo v lokálním maximu OH-. S pokrokem v technologii výroby optických vláken bylo lokální maximum OH- eliminováno. Opět zde dochází ke spojování jednotlivých oken tentokrát II. a V. Při spojení těchto oken získáváme přenosový kanál o šířce pásma 50 THz [12].

Dále jsou na obrázku znázorněné OH- nečistoty, které zapříčiňuji největší ztráty v optických vláknech, tyto nečistoty vznikají při žíhání skla ve výrobě [11]. Barevně je na obrázku znázorněné nové značení pásem vlnových délek jednovidových optických vláken:

- O-pásmo: 1260–1360 nm (Original),
- E-pásmo: 1360–1460 nm (Extended),
- S-pásmo: 1460–1530 nm (Short),

- C-pásmo: 1530–1565 nm (Conventional),
- L-pásmo: 1565–1625 nm (Long)
- U-pásmo: 1625–1675 nm (Ultra long).



Obr. 1.8: Útlumová charakteristika optického vlákna [12].

1.5.2 Šířka pásma

Šířka pásma udává maximální kmitočet přenášeného signálu, který může být přenesen optickým vláknem na vzdálenost 1 km bez nadměrného zkreslení signálu. Tento parametr se uvádí v jednotkách MHz · km. Šířka pásma nejvíce závisí na vlnové délce vysílaného signálu a dále také na materiálu optického vlákna. Šířka pásma je přímo úměrná s rychlosti přenosu, to znamená že čím větší šířku pásma máme, tím větší rychlosti dokážeme komunikovat [11].

1.5.3 Numerická apertura

Numerická apertura nám udává maximální úhel pod kterým může světelný paprsek vstupovat do optického vlákna tak, aby byl vláknem přenášen. Vztah pro výpočet numerické apertury je vidět na výpisu 1.3. Číselně je NA rovna sinu maximálního úhlu, pod kterým se vstupující paprsky budou ještě šířit od začátku vlákna k jeho konci. Paprsek, který vstoupí do vlákna pod větším úhlem, se již šířit vláknem nebude [11].

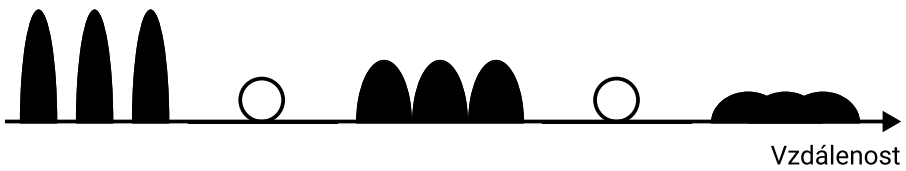
Tento parametr nám říká, jak velký optický výkon je schopné jádro vlákna navázat z okolního prostředí. Čím je tento parametr větší, tím větší celkový výkon dokážeme v optickém vláknu přenést.

$$NA = sin\Theta = \sqrt{n_j^2 - n_p^2} \tag{1.3}$$

1.5.4 Disperze

Disperzi řadíme k nejdůležitějším parametrům optických vláken. Je jednou z hlavních příčin zkreslení signálu na konci vedení. Disperze se definuje jako rozdíl šířky impulsu v polovině výšky na konci a na začátku optického vlákna, jak je zobrazeno na obrázku 1.9 [3].

Disperzi můžeme rozdělit na několik části a to na: vidovou, chromatickou, polarizační, materiálovou a vlnovodnou disperzi. V následujících sekcích si je postupně rozebereme.



Obr. 1.9: Vliv disperze na optický signál [10].

Materiálová disperze

Materiálová disperze vzniká tak, že světlo o různých vlnových délkách se šíří různou rychlostí. To má za následek to, že vstupní signál, který obsahuje více spektrálních složek, se bude časově natahovat díky tomu, že některé spektrální složky se budou zpožďovat oproti jiným složkám [3].

Vlnovodná disperze

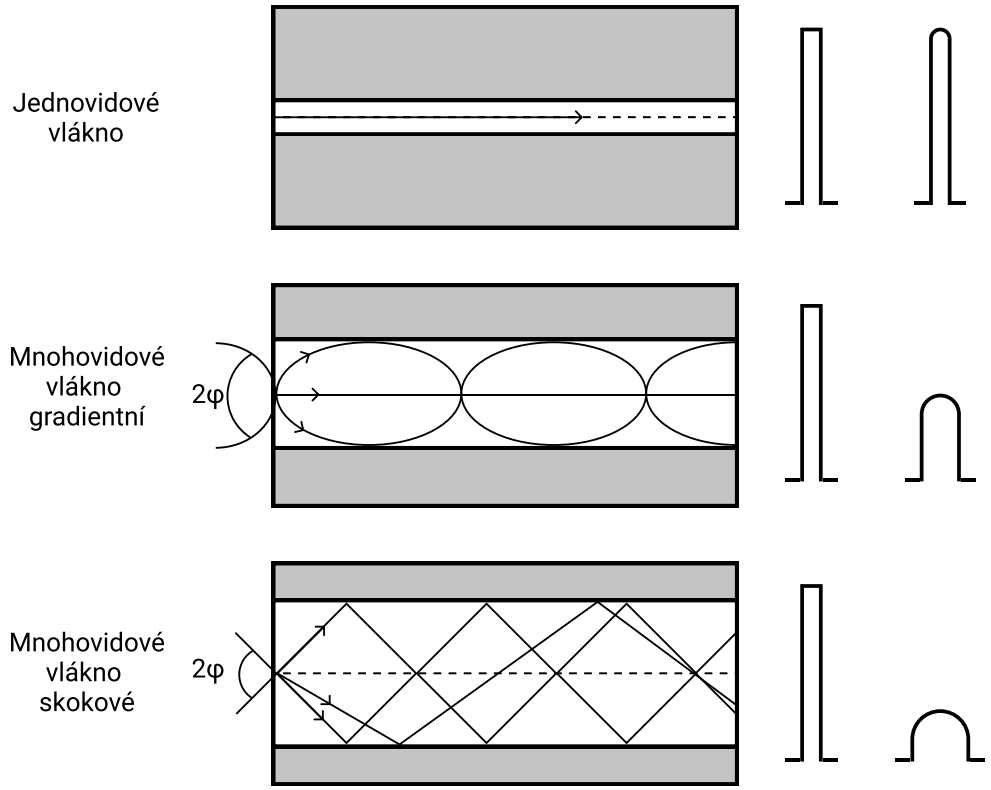
Tato disperze se u mnohovidových vláken neřeší, ale hraje velkou roli u jednovidových vláken, kde tvoří velkou část celkové disperze. Příčinou této disperze je, že konstanta šíření je pro každý vid jiná, tím se mění i jeho kmitočet, což má za následek změnu rychlosti šíření [3].

Chromatická disperze

Chromatická disperze je tvořena disperzí materiálovou a vlnovodnou. Tato disperze nejvíce omezuje právě jednovidové vlákna, kde omezuje hlavně přenosovou rychlost.

Vidová disperze

Vidová disperze se projevuje pouze u mnohovidových vláken. Je způsobena rozdílnou dráhou jednotlivých paprsku (vidů) v optickém vláknu. Můžeme říct, že čím větší bude úhel mezi osou a trajektorií paprsku, tím bude výsledná trajektorie delší mezi začátkem a koncem vedením. To je důvod proč se vyšší vidy, které jsou blíže svému meznímu kmitočtu šíří v optickém vlákně pomaleji než vidy nižší [3]. Vliv vidové disperze můžeme vidět na obr. 1.10.

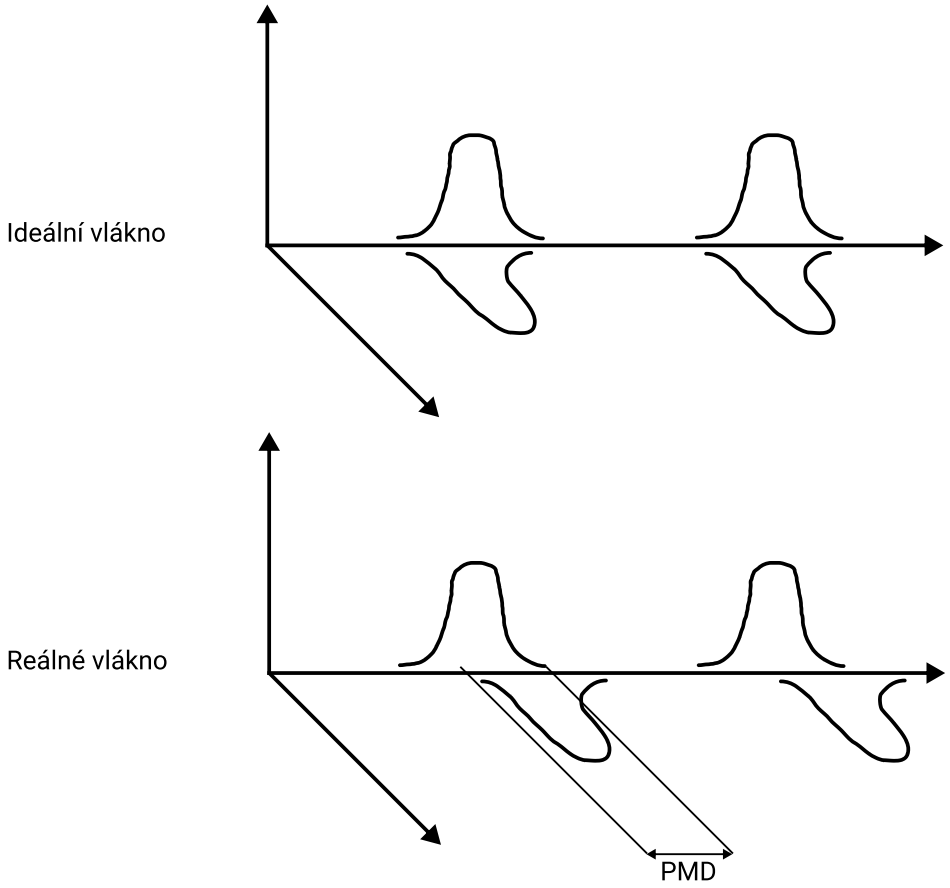


Obr. 1.10: Porovnaní disperze v různých vláknech [3].

Polarizační vidová disperze

Tato disperze se opět vyskytuje pouze u jednovidových vláken, kde se vid ve vlákně šíří ve dvou vzájemně kolmých polarizačních rovinách. Každá nepřesnost z výroby nebo i špatné uložení kabelu v zemi má za následek to, že se obě roviny budou šířit

různou rychlostí a to má za následek zkreslení výsledného signálu, jak je vidět na obr. 1.11. Tato disperze se nejméně podílí na celkové disperzi optického vlákna [3].



Obr. 1.11: Polarizační vidová disperze [11].

1.5.5 Rozptyl

Rozptyl světla vzniká odchýlením světelného záření od původní dráhy letu malými částicemi, které mohou být různého skupenství. Stejně jako u disperze i u rozptylů rozeznáváme několik druhů, které si níže popíšeme.

Rayleighův rozptyl

Tento rozptyl vzniká nepravidelnou strukturou materiálu v optickém vlákně. Důsledkem toho je, že se světelný paprsek odráží a tříští do všech směrů. Tyto odkloněné části paprsku následně vniknou do pláště kde zaniknou. Rayleighův rozptyl nastává přibližné 10milionkrát častěji, než ostatní rozptyly a proto je jedním z dominantních

jevů, které v optice pozorujeme. Rayleighův rozptyl je využíván v reflektometrické metodě (OTDR) měření a analýzy optických tras. Využívá se zde toho, že část Rayleighova rozptylu se vrátí zpátky ke zdroji kde je následně zachyceno a zpracováno [11, 13].

Ramanův rozptyl

Dochází zde k interakci mezí světlem šířícím se optickým vláknem a teplotně nabuzenými molekulovými vibracemi. Dojde-li ke srážce fotonu a molekuly dané látky, nastává rozptyl. Tyto rozptýlené fotony mají opět jinou frekvenci než původní fotony [13].

Brillouinův rozptyl

Tento typ rozptylu vzniká stejně jako všechny výše uvedené rozptyly interakcí světelného paprsku a materiálem optického vlákna. Hlavní příčinou je změna indexu lomu materiálu vlivem tahové a tlakové deformace. Vzniká tak akustická vlna, která interaguje se světelnou vlnou. Tento rozptyl je hlavně významný pro jednovidová vlákna, kde je nutná úzká spektrální čára. Fotony rozptýleného paprsku mají většinou opačný směr a jinou frekvenci než fotony původního paprsku. Pomoci Brillouinova rozptylu se dá měřit mechanická deformace a teplota optického vlákna [13].

1.5.6 Fresnelův odraz

Fresnelův odraz vzniká ve vlákně náhlými změnami indexu lomu. Tyto náhle změny jsou nejčastěji způsobeny nečistotami (vzduchovou mezerou) na koncích vlákna, na spojkách a při přerušení vlákna. Velikost odrazu závisí na velikosti úhlu pod kterým signál narazí na rozhraní dvou materiálů a na velikosti změny hustoty prostředí. Tohoto odrazu se využívá při měření pomocí OTDR, kde nám Fresnelův odraz dokáže lokalizovat spojky, konec vedení, nebo poruchu na trase [14].

1.5.7 Absorpce

Ztráty absorpcí v optickém vláknu můžeme rozdělit na dvě části a to vlastní a nevlastní.

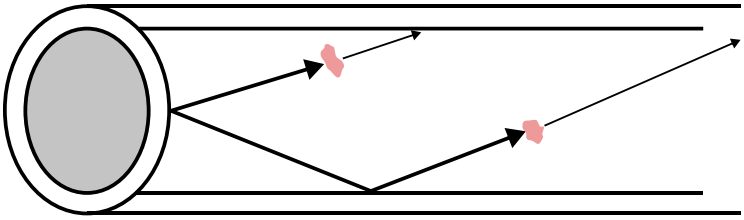
Vlastní absorpce

Vlastní absorpce je způsobená materiálem vlákna, který pohlcuje optické záření. Tato absorpce se velice liší u každého materiálu, např. skleněná vlákna vykazují

velmi malou vlastní absorpci, naopak křemenný materiál vykazuje absorpční maxima, jak v ultrafialové, tak i v infračervené oblasti spektra [11].

Nevlastní absorpce

Nevlastní absorpce zapříčiňují nečistoty v optickém vláknu, které pohltí část optického záření. Tyto nečistoty se do vlákna dostanou při výrobě. Nejčastěji tuto absorpci zapříčiňují ionty kovů jako je železo a měď. Dalším velkým viníkem nevlastní absorpce jsou ionty OH-. Znázornění nevlastní absorpce můžeme vidět na obrázku 1.12, kde jsou nečistoty znázorněné červenou barvou [11].



Obr. 1.12: Nevlastní absorpce v optickém vláknu [11].

1.6 Vícenásobné využití optického vlákna

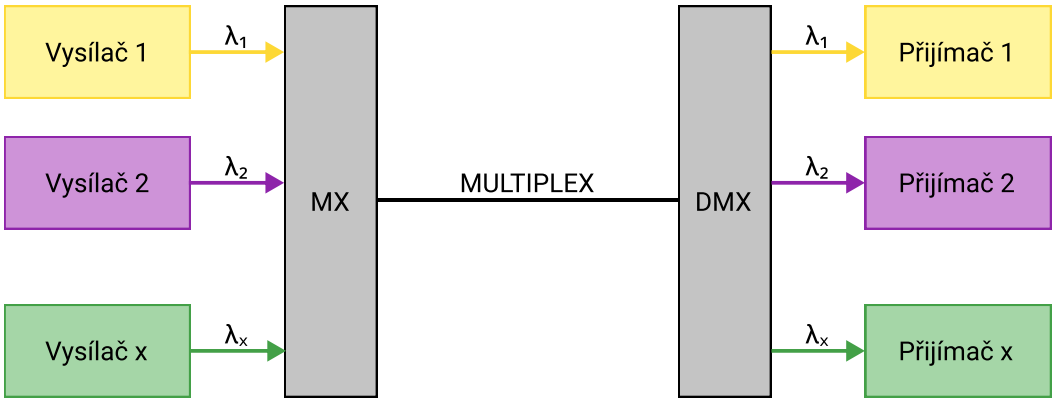
V dnešní informační a multimediální době se zvyšují nároky na počty přenosových kanálů, které by tuto potřebu obsloužily. Jedním z nejjednodušších řešení by bylo položit nový optický kabel, to je ale velice nákladné jak finančně, tak i časově. Proto se hledaly způsoby jak zvýšit přenosovou kapacitu již položených optických tras. Tím řešením je multiplexování optického signálu na vedení. V optických sítích můžeme využít následujících způsobů vícenásobného přenosu:

- Časový multiplex – Nejednoduší multiplex, využívá časových intervalů které jsou pro jednotlivé signály určeny. V těchto intervalech je na jedné straně připojený vysílač a na druhé přijímač pro konkrétní signál. Po uplynutí intervalu se vysílače a přijímače přepojí.
- Prostorový multiplex – Jedná se o využití více optických vláken pro přenos různých signálů.
- Frekvenční multiplex – Jednotlivé signály se přenáší do vyšších kmitočtových pásem, kde se vytvoří skupiny, tyto skupiny jsou následně namodulovány na optický signál. Tento multiplex je založený pouze na elektronickém obvodu ve vysílači a jeho funkčnost je omezená parametry zdrojů světelného záření.

- Vlnový multiplex – Každý přenášený signál využívá různou vlnovou délku světla. Tyto vlnové délky spadají do tzv. oken které vykazují nejmenší útlum ve vlákně.
- Elektronický multiplex – Po jednom optickém vlákně se přenášejí vícestavové signály na rozdíl od klasických binárních signálů.
- Hybridní multiplex – Tento multiplex slučuje dva předchozí multiplexery (vlnový a elektronický multiplex) do jednoho multiplexu.

1.6.1 Vlnový multiplex

Právě vlnový multiplex (WDM) je v optice nejčastěji využívám. Úkolem WDM multiplexoru je sdružit světelné svazky o různých vlnových délkách do jediného optického vlákna. Multiplexory často obsahují selektivní členy neboli optické filtry na potlačení interference. Demultiplexor má za úkol rozčlenit přijmutý světelný svazek na dílčí paprsky podle vlnové délky. Demultiplexor rovněž obsahuje optické filtry nebo přímo selektivní fotodetektory [15]. Znázorněny princip vlnového multiplexu můžeme vidět na obrázku 1.13.



Obr. 1.13: Princip vlnového multiplexu [3].

U vlnového multiplexu velmi závisí na typu zdroje světla a tedy i na šířce vyzařovaného spektra. Je zřejmé, že když budeme mít užší spektrum (laser), tak do celkové šířky pásma vměstnáme více vlnových délek tedy i více kanálu pro přenos. Samozřejmě musí být mezi jednotlivými kanály určité rozestupy, aby nedocházelo k interferencím. Tyto rozestupy lze jednoduše vypočítat pomocí vzorce 1.4. Hodnotu $\Delta\lambda$ velice ovlivňuje konstrukce multiplexerů, demultiplexerů a použitý zdroj záření [3].

$$\Delta\lambda = \lambda_{j+1} - \lambda_j \tag{1.4}$$

Jelikož je multiplexer a demultiplexer další z řady prvku na optické trase, tak se musí počítat s tím, že přispívá do celkového útlumu trasy. Zavádí se zde pojem vložný útlum, který definuje ztráty, které vzniknou průchodem optického záření multiplexorem a demultiplexorem, pro výpočet lze použít vzorec 1.2 [3].

Postupem času se objevovaly nové typy vlnových multiplexerů, které dokážou přenášet ještě více vlnových délek.

Široký vlnový multiplex

Široký vlnový multiplex (WWDM) se nejčastěji využívá pro přenos gigabitového, nebo 10gigabitového Ethernetu. Oproti WDM používá WWDM 4 vlnové délky a používá se jak pro jednovidové tak i mnohovidové optické vlákna. Typický odstup jednotlivých kanálu u WWDM je 25 nm [3].

Hustý vlnový multiplex

Hustý vlnový multiplex (DWDM) dokáže do jednoho optického vlákna vměstnat desítky kanálu. Jelikož využívá minimální odstup mezi kanály. Tento odstup může dosahovat pouhých 0,1 nm. Pro tak malý odstup je nutné využít ty nejlepší úzko spektrální a frekvenčně stabilní laserové zdroje. Ty musí být velmi dobře chlazeny, jelikož kolísání teploty zdroje rovněž ovlivňuje šířku spektrální čáry [16].

V reálném nasazení je nutné si uvědomit, že je velký rozdíl mezí prvním a posledním kanálem co se týče jeho přenosových vlastností, hlavně v jeho dosahu. Proto je vždy nutné pracovat s nejhoršími přenosovými vlastnostmi, co nám DWDM nabízí. DWDM se nejčastěji používá na delších optických trasách [3].

Hrubý vlnový multiplex

Hrubý vlnový multiplex (CWDM) je levnější varianta DWDM s tím rozdílem, že využívá větší rozestupy mezi kanály. U CWDM lze využít až 18 přenosových kanálů, ale musí se použit speciální optické vlákno typu Metro. Odstup kanálu byl přizpůsoben tomu, aby šlo použít jako zdroj laser který není nutné chladit. Odstupy kanálu se pohybují v rozmezí 50 až 8 nm pro vlnovou délku 1550 nm. CWDM se využívá v metropolitních optických sítích a řeší se pomocí něj poslední míle trasy. CWDM dokáže přenášet gigabitový Ethernet až do vzdálenosti 80 kilometrů [3, 17, 18].

1.7 Měření útlumu v optických vláknech

Útlum je v optických sítích nejdůležitější přenosový parametr a je celkovým měřítkem optických ztrát při šíření optického signálu ve vláknu. Měření útlumu se

nejčastěji provádí na diskrétních vlnových délkách 850 nm, 1300 nm, 1310 nm, nebo 1550 nm.

Měření útlumu lze provádět třemi základními metodami:

- Metoda dvou délek.
- Metoda vložného útlumu.
- Metoda zpětného rozptylu.

1.7.1 Metoda dvou délek

Metoda dvou délek se využívá jako referenční, jelikož dosahuje nevetší přesnosti v měření. Nejdříve se změří výkon P2 na konci optického vlákna a následně se za nezměněných podmínek vazby vlákno zlomí přibližně 2 m od počátku a změří se výkon P1. Velkou výhodou teto metody je její velká přesnost, která dosahuje až 0,01 dB/km. Nevýhodu je, že se jedná o destruktivní metodu, jelikož se musí vlákno vlákno v rámci měření zlomit [12, 19].

1.7.2 Metoda vložného útlumu

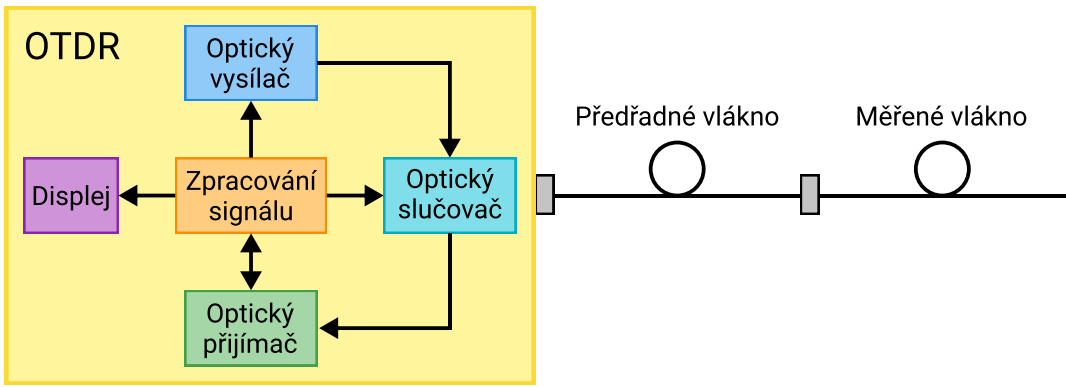
Měření útlumu pomocí metody vložného útlumu není destruktivní a provádí se ve dvou měřeních. Nejdříve se provede kalibrační měření pomocí referenčního vlákna, tím získáme hodnotu P1. Referenční vlákno bývá dlouhé zhruba 2 m. Ve druhém kroku se místo referenčního vlákna použije měřené vlákno a tím získáme hodnotu P2. Tato metoda je provozní a tudíž se dá provádět přímo v terénu na trase. Přesnost této metody je hodně závislá na použitých konektorech, dosahuje se zde přesnosti zhruba 0,2 dB/km [12, 19].

1.7.3 Metoda zpětného rozptylu

Metoda zpětného rozptylu využívá zcela rozdílného principu než předchozí dvě metody. U této metody se vyhodnocuje časová závislost zpětného rozptýleného optického výkonu při šíření impulsu ve vlákně. To nám dává větší množství informací nejenom o celkovém útlumu, ale rovněž i o kvalitě vlákna, útlumu na konektorech a svárech po celé jeho délce. Nevýhodou oproti předchozím metodám je to, že zařízení jemuž se říká optický reflektometr (OTDR) je proti měřiči výkonu podstatně sofistikovanější zařízení a tudíž je i o dost dražší. Přesto, že se jedná o poměrně drahé zařízení tak je OTDR nejpoužívanější metodou pro měření útlumu. OTDR se nevyužívá pouze pro měření útlumu, ale rovněž pro lokalizaci případných poruch na optické trase [12, 19].

OTDR metoda využívá dvou jevů, které v optických vláknech vznikají a to Rayleighova rozptylu viz 1.5.5 pro měření útlumu na trase a Fresnelova odrazu viz 1.5.6,

který je pro měření útlumu nežádoucí, ale využívá se pro lokalizaci poruch a měření délky vlákna. Do optického vlákna je vyslán signál, ale část jeho energie je odražená zpět právě na zmiňovaných jevech. Zpětně odražený signál je následně zpracován v OTDR přístroji. Schéma OTDR přistroje na je na obrázku 1.14.



Obr. 1.14: Schéma OTDR [12].

Vlnová délka

Důležitým parametrem pro OTDR je měřící vlnová délka, která se skládá z centrální vlnové délky a šířky čáry. Šířka čáry nám udává, jak široká bude celková vlnová délka. Tedy pokud máme vlnovou délku 1550 nm a šířka čáry je 20 nm tak rozsah testovací vlnové délky bude 1540–1560 nm [14].

Rayleighv zpětný rozptyl je závislý na vlnové délce signálu, to znamená, že pro vlnovou délku 1310 nm bude Rayleighův zpětný rozptyl větší než pro vlnovou délku 1550 nm. Tento jev se nazývá infračervené zředění. Pro testování je důležité, aby byla využita správná vlnová délka, která bude i následně využívaná pro ostrý provoz [19].

Dynamický rozsah

Dynamický rozsah se udává v dB a změnou dynamického rozsahu určujeme, jak dlouhé vlákno chceme měřit. Čím větší velikost dynamického rozsahu, tím delší vlákno lze měřit. Senzor v OTDR musí být dostatečně kvalitní, aby zachytil i ty nejslabší odražené signály, které se odrazí až na konci měřeného vlákna. Vysílaný testovací impulz musí mít dostatečnou sílu, aby se dokázal dostat až na konec měřeného vlákna. Citlivost senzoru a síla impulsu nám určuje dynamický rozsah [14].

Vzorkování

Vzorkování je často přehlížené, ale rovněž patří mezí důležité parametry OTDR. Vzorkování nám určuje vzdálenost mezi dvěma referenčními body, nebo také vzorky. Vzdálenost mezi vzorky můžeme vypočítat pomocí vzorce 1.5, kde l_{vz} je vzdálenost mezi vzorky, l_l je celková délka spoje a p_{vz} je počet vzorku [19].

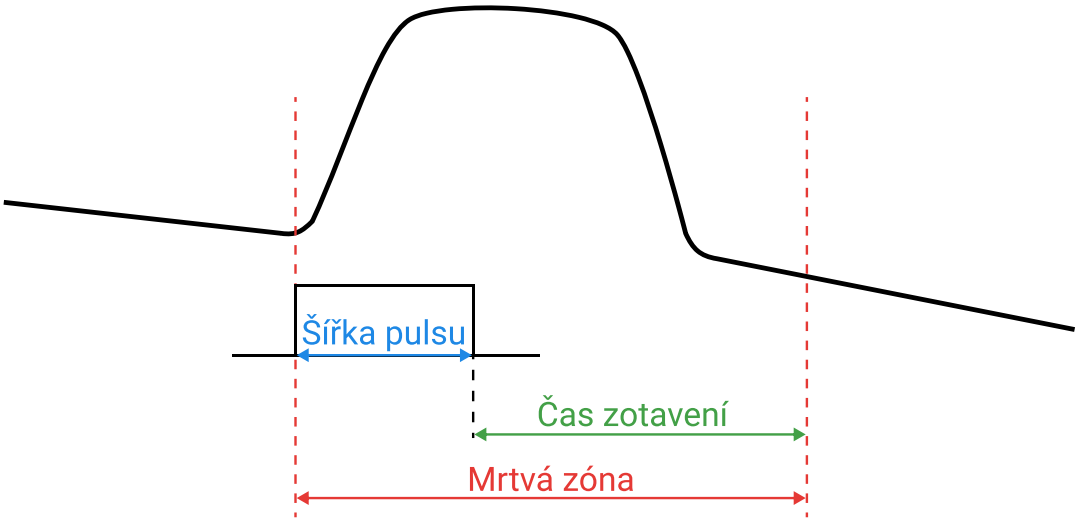
$$l_{vz} = \frac{l_l}{p_{vz}} \tag{1.5}$$

Pokud nebudeme mít dostatečné vzorkování a budeme měřit velkou vzdálenost, tak se nám může stát, že nebudeme moci závadu přesně lokalizovat. Rozmezí této závady může být i několik metrů. Při opravě na optické trase může tato vzdálenost hrát velkou roli, například pokud musíme kvůli opravě udělat výkop na dopravní komunikaci.

Mrtvá zóna

Mrtvá zóna je určitá vzdálenost za první konektorem OTDR, ve které se nedá měřit, jelikož světlo odražené zpět díky Fresnelovu odrazu způsobí saturaci fotodiody ve fotodetektoru v OTDR. Tento první Fresnelův odraz není možné úplně odstranit a proto se používá předřadné vlákno, jak je vidět na obrázku 1.15. Díky tomu se mrtvá zóna projeví pouze v předřadném vláknu a ne v měřeném vláknu. Velikost mrtvé zóny je závislá na šířce impulsů, které do vlákna vysíláme a na času zotavení fotodiody. Délka předřadného vlákna se pohybuje od stovek metrů až po jednotky kilometrů.

Kromě klasické mrtvé zóny rozeznáváme ještě další dva typy a to event dead zone (EDZ) a attenuation dead zone (ADZ). EDZ nám udává minimální vzdálenost od odrazné události, za kterou je možné lokalizovat další odraznou událost. ADZ udává rovněž minimální vzdálenost, ale tentokrát pro měření neodrazivé události například svár na trase [12].



Obr. 1.15: Mrtvá zóna [12].

2 Zabezpečení optických vláken

Celkově se považuje využití optických vláken za bezpečnější, nežli využití klasických metalických kabelů. Jelikož optické vlákno nevyzařuje žádné elektromagnetické záření, je velice těžké jej odposlouchávat. To ale neznamená, že by optická vlákna nebyla zranitelná proti útokům, a my nemuseli řešit jejich zabezpečení. Jelikož v dnešní době se rapidním tempem optické sítě rozšiřují i tam, kde by to bylo před několika lety nepředstavitelné (koncoví zákazníci internetu, továrny, přístroje, atd.), musí se zabezpečení optických vláken čím dále více řešit. Útočnici vymýšlejí stále více sofistikovanější druhy útoku a dohledová centra na to musí reagovat.

Pokud se zaměříme konkretně na fyzickou vrstvu optických sítí, tak útočník může provádět dva typy útoku:

- Útok na infrastrukturu sítě – Může se jednat o vandalismus, nebo cílený útok který má za cíl destrukci prvků na optické trase, nebo přerušení spojů. Rovněž se útočník nemusí pokoušet o destrukci, ale naopak se snaží vložit do sítě falešný prvek.
- Útok na přenášené informace – Zde se myslí odposlouchávaní informací a vkládaní podvodných informací.

Celkově útok na fyzickou vrstvu je spíše záležitost dálkových spojů a tras, jelikož lokální spoje jsou již chráněny tím, že se nachází v lokalitách a objektech, do kterých nemá většinou útočník přístup. Tyto objekty bývají chráněné pomocí klasických nástrojů, jako jsou kamerové systémy, přístupy pomocí RFID karty, nebo biometrických dat. U dálkových spojů je tohoto těžké dosáhnout, jelikož rozsah těchto sítí může být až stovky kilometrů. Proto zde využití výše zmíněných technologii nedává smysl.

2.1 Odposlech optických vláken

Odposlouchávání optických vláken je obtížné ne však nemožné, většinou se využívá fyzikálních zákonitostí jako využití odrazu lomu, nebo rozdělení paprsků. Odposlech můžeme rozdělit na dva druhy a to s přerušením vlákna, nebo bez přerušení vlákna [20].

2.1.1 Odposlech s přerušením

Odposlech s přerušením má velkou nevýhodu a to tu, že je nutné vlákno přerušit a znovu ho navázat. Toto přerušení vlákna je celkem lehce odhalitelné v síti, které provádějí kontinuální měření pomocí OTDR, jelikož jde na výsledném grafu vidět že se celková trasa zkrátila.

Optický rozdělovač

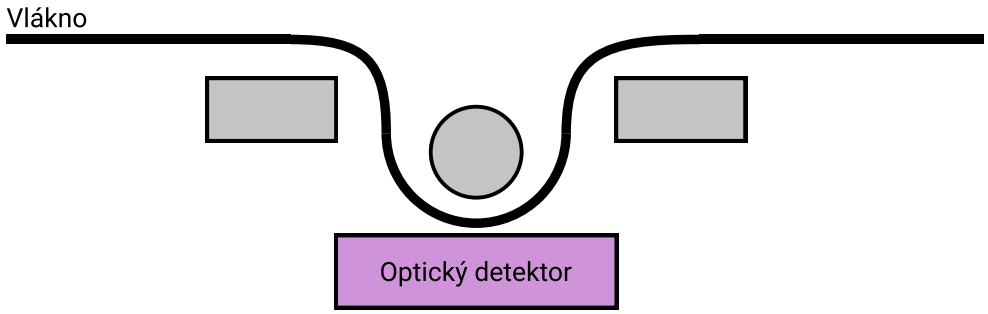
Jedná se o rozdělovač ve tvaru písmene Y, který se vloží do přerušené cesty. Pokud je rozdělovač správně napojen na trasu, tak se část optického signálu odkloní a útočník ho může následně dále zpracovávat. Zbytek signálu jde původní cestou, ale jde zde vypozorovat, že se celkový útlum trasy radikálně zvedl [20].

2.1.2 Odposlech bez přerušeni vlákna

Tyto metody jsou založeny na ohýbaní vlákna. Jedná se o nejednodušší metody odposlechu. Při správném ohybu dojde k částečnému odrazu, což zapříčiní, že část signálu pokračuje správnou cestou. část signál je nasměřována do vrstvy s vyšším indexem lomu, kde je připraven fotodetektor na zachycení signálů. Je zde obrovská výhoda a to, že se vlákno nemusí přerušit a tudíž jsou tyto metody velice těžké na zjištění. Obecnou ochranou je zde nepřetržitá detekce změn výkonové úrovně signálu [20].

Technika makro ohybu

Tato metoda je založena na tom, že při ohybu vlákna se změní úhel dopadu některých paprsků. Tyto paprsky jsou následně zachyceny pomocí fotodetektoru. Zařízení, které toto dovede se jmenuje clip-on coupler a dá se běžně koupit na trhu za částku okolo několika desítek tisíc korun [21]. Jak zařízení pracuje, je zobrazeno na obrázku 2.1.

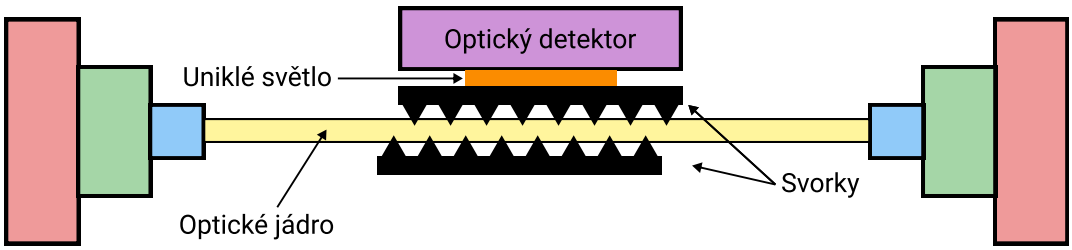


Obr. 2.1: Princip makro ohybů [21].

Technika mikro ohybu

Tato metoda využívá ještě menších ohybů než předchozí metoda. Optické vlákno se musí nejdříve svléknout z ochranných vrstev a následně se na jádro připevní odpo-

slouchávací zařízení, které vlákno lehce zohýbá a vychýlený signál zachytí pomocí optického detektoru.



Obr. 2.2: Princip mikro ohybů [21].

2.2 Ochrana infrastruktury sítě

K ochraně infrastruktury patří jak bylo zmíněno výše jejich fyzická ochrana, která zabraňuje, nebo ztěžuje útočníkovi přístup k zařízením. U ochrany spojů hodně napomáhá jejich vhodné umístění, tedy vybrat vhodnou trasu, kudy síť povede. Obecné pravidlo také je, že bychom měli využívat redundantní trasy a zálohování jednotlivých prvku sítě.

2.2.1 Ochrana konektorů a ukončení spoje

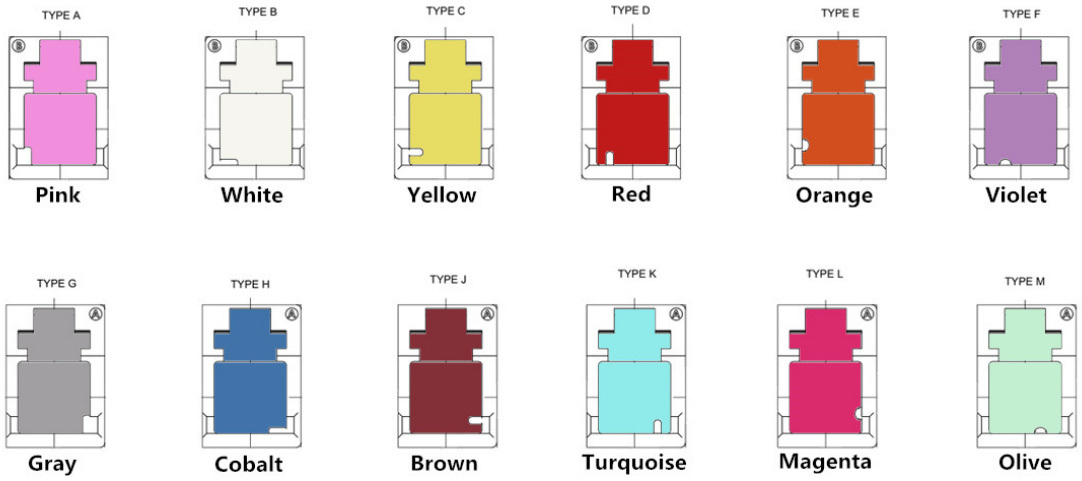
Ochrana ukončení spojů je jakousi poslední linií ochrany celé sítě. Když útočník překoná všechny ostatní ochrany, nezbývá nic jiného než se spolehnout na tuto poslední linii. Jedná se o velice jednoduché zařízení, které útočníkovi ztěžují, nebo zabraňují manipulaci s optickými vlákny např. v místnosti se servery. Tyto zařízení jsou vyráběný jak pro klasické metalické sítě, tak i pro optické sítě a jejich konektory [22].

Blokátory

Zde spadají prvky, které chrání, nebo blokují prvky optické kabeláže a konektivity. Prvním typem jsou jednoduché blokátory, ty slouží k zabezpečení nepoužitých portů proti neautorizovanému přístupu, nebo také proti poškození nevhodným konektorem. Tyto blokátory lze odstranit pouze speciálním nástrojem, který je dodáván spolu s blokátory. Rovněž existují i trvalé blokátory, které po implementaci již nelze vytáhnout. Dalším typem jsou blokátory se zámkem, ty mají za úkol znemožnit vytažení kabelu z portu bez správného klíče [22].

Klíčování

Tato metoda využívá klíčování konektoru, které zabraňují neúmyslného nebo neautorizovaného zapojení kabelu. Rovněž zabraňuje připojení jiných nekompatibilních konektorů. Konektory se vyrábějí ve 12 barevných variantách, které nelze mezi sebou kombinovat, jak je vidět na obrázku 2.3. Klíčování konektoru zvyšuje celkovou integritu a zabezpečení optických vláken [23].



Obr. 2.3: Typy klíčování LC konektorů [23].

2.3 Automatizované systémy správy infrastruktury

K zabezpečeni fyzické vrstvy sítě lze rovněž využít i automatizovaný systém správy infrastruktury (AIM), i když primární účel tohoto systému je správa propojení jednotlivých portů zařízení mezi sebou. Systém AIM vznikl hlavně pro datová centra, kde dochází k velké koncentraci prvků a spojů mezi nimi. V dnešní době se však systémy AIM dostávají i do menších firemních infrastruktur.

Problém AIM je v tom, že se jedná o proprietární řešení jednotlivých výrobců (Molex, RiT, Tyco Electronics, Commscope a další), tudíž nejsou mezi sebou kompatibilní. Pro AIM neexistuje žádný ucelený standart, kterého by se mohli výrobci držet a tudíž by jejich výrobky mohly být mezi sebou kompatibilní.

Hlavní výhoda systému AIM z bezpečnostního hlediska je ta, že správce sítě vidí v reálném čase stav jednotlivých portů a jejích vzájemné propojení. Tudíž když by útočník odpojil nějaký kabel, tak správce sítě může hned patřičně zareagovat [24] [25].

2.3.1 Hardware

Hardware AIM se skládá ze tří základních prvku:

- Analyzátor – Nebo také skener, jedná se o nejdůležitější prvek celého systému, je připojený k několika panelům a má za úkol analyzovat všechna propojení na těchto panelech, následně tyto data odesílá na server. Analyzátor je k panelům většinou připojeny pomocí speciálního kabelu. Analyzátor má přiřazenou vlastni IP adresu v síti.
- Server – Server shromažďuje všechna data a správce sítě si pomocí dodávaného softwaru k ním může přistupovat.
- Panel – Tyto panely slouží k využití speciálních monitorovacích technik viz 2.3.4. Pro každou techniku se využívá speciální typ panelu.

2.3.2 Software

Abychom mohli mít neustálý přehled o situaci v sítí, tak výrobci vetšinou ke svému řešení dodávají monitorovací program, který nám v reálném čase ukazuje aktuální situaci a udržuje databázi všech propojení. Programy umožňuji vytvářet jednotlivé místnosti v budově a možnosti je také nahrát celý plán budovy. Dále správce sítě vidí všechny zařízení od racků, aktivních prvku, zásuvek až po koncová zařízení.

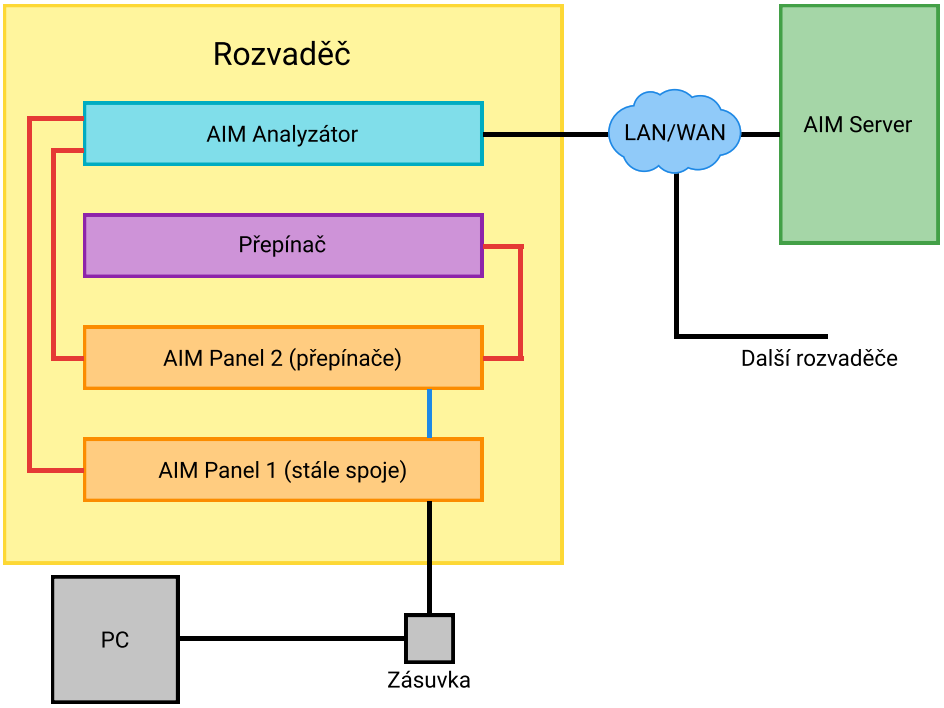
Díky všem těmto datům které se uchovávají, lze vytvářet velké množství statistik a reportů, což správcům sítě ukáže vytížení prvku a nevyužití portu atd. Všechny tyto data můžou napomoci k optimalizaci celé sítě [24].

2.3.3 Architektura propojení AIM

Celá architektura propojení je na obrázku 2.4. V každém rozvaděči jsou nějaké síťové prvky např. přepínač. Tyto zařízení vetšinou od výroby nepodporuji žádnou z technik monitorování. Proto jsou tyto porty vyvedené z přepínače do AIM panelu 2 (červený spoj). Na další AIM panel 2 (patch panel) jsou vyvedené stále spoje k datovým zásuvkám. Oba dva AIM panely jsou propojeny s AIM analyzátorem (červený spoj), který monitoruje propojení mezi AMI panely (modrý spoj). Zde se musí využít speciálních kabelů, které se liší v závislosti na monitorovací technice. Analyzátory všech rozvaděčů v sítí komunikují se AIM serverem. Tento server má informace o všech stálých spojích v sítích a o vzájemném propojení prvků mezi sebou [25].

2.3.4 Monitorování

Monitorováním se myslí sledování stavů jednotlivých portů a jejích vzájemné propojení pomocí spojů. Jak již bylo zmíněno, každý výrobce si své řešení navrhuje sám



Obr. 2.4: Architektura systému AIM [25].

a řešení nejsou mezi sebou kompatibilní. Někdy lze využít k monitorování fyzické vrstvy i vrstvy vyšší.

Technika mikrospínače

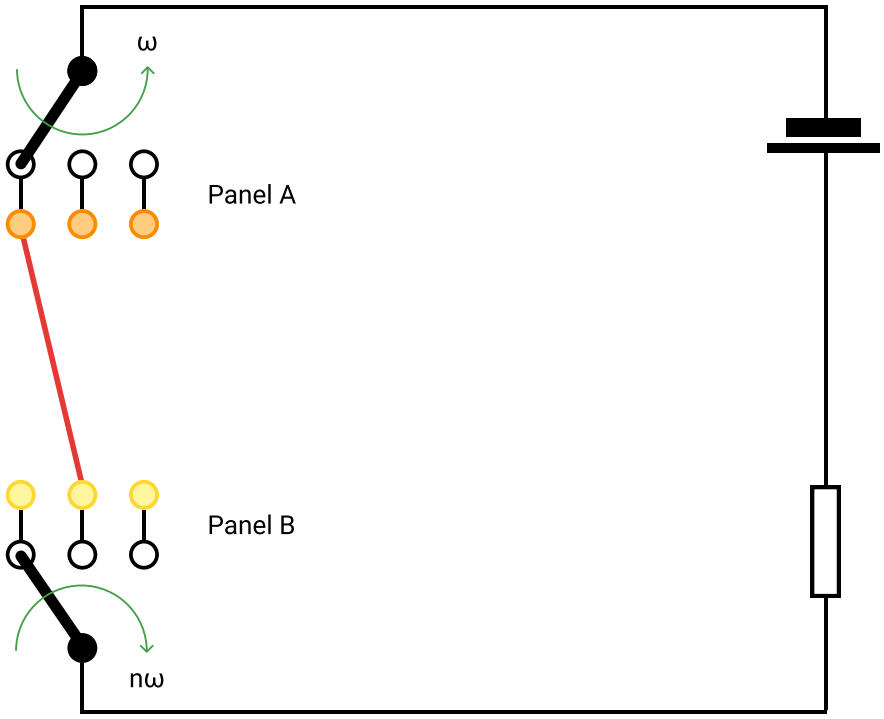
Jedná se o jednu z nejjednodušších metod monitorování. V dnešní době je už hodně zastaralá a nenabízí nám tolik možností jako novější techniky. Technika mikrospínače funguje tak, že je do portu přidaný mikrospínač, který se po připojení kabelu sepne. Systém je tak schopný poznat, které porty jsou obsazeny. Nevýhodou je, že systém nepozná, zda je kabel funkční. Může dojít k přerušení kabelu a systém to nedokáže rozpoznat, jelikož je mikrospínač v portu stále sepnutý. Další nevýhodou je, že nejde poznat jaké dva porty jsou kabelem propojeny. Naopak velkou výhodou je to, že lze použit standardní kabely [24].

Technika přídavného vodiče

Této technice se také říká metoda devátého vodiče. Devátého protože je ke klasickým osmi vodičům přidán devátý na možnost monitorování. Toto je případ klasických metalických sítí, ale tato technika jde implementovat i do sítí optických. Technika funguje tak, že je každý port vybaven kontaktní ploškou, nebo jinými kontakty. Když se do portu připojí kabel, dojde ke spojení plošky a monitorovacího vodiče.

Jednotlivé plošky jsou vyvedeny do analyzátoru propojení, který je cyklicky testuje jak je vidět na obrázku 2.5. Princip je takový, že analyzátor přepíná panel A s úhlovou rychlosti ω a panel B s úhlovou rychlosti ωn kde n je počet portu na panelu. Když dojde k propojení obvodu, analyzátor zapíše na server, že jsou tyto dva porty propojeny.

Výhodou tohoto řešení je, že lze poznat když je kabel přerušen, protože s velkou pravděpodobností se přeruší i monitorující vodič. Naopak nevýhodou je použití speciálních kabelů, které toto řešení prodražuje [24] [25].



Obr. 2.5: Princip přídavného vodiče [25].

Technika RFID čipu

Tato technika funguje tak, že port v panelu je speciálně upraven a obsahuje kontaktní rozhraní. Aby tato technika fungovala, musí se využívat speciální typ kabelu, který má na obou koncích speciální konektor. Tento konektor obsahuje RFID čip, který v sobě má uložené ID kabelu. Po zapojení kabelu do portu dojte k propojení rozhraní a čipu na daném konektoru. Následně si analyzátor muže vyčíst ID kabelu a spárovat ho s daným portem na panelu, k tomu stejnému dojde i na druhé straně kabelu. Systém má tedy uloženo – port A je propojen s portem B a to kabelem X. Máme tedy kompletní cestu odkud kam vede daný kabel.

Nevýhodou je, že je nutné využít speciálních kabelu, které obsahují RFID čip, což toto řešení opět prodražuje. Další nevýhodou je, že nelze zjistit, zda je kabel stále funkční a zda není přerušen [24] [25].

2.4 Monitorování optických vláken Viavi

Monitorováním vláken se rozumí průběžné hodnocení kvality vláken pomocí softwarových nástrojů a zařízení, které tvoří integrovaný systém monitorování a správy vláken. Tyto prvky společně usnadňují detekci poruch, zhoršení kvality nebo narušení bezpečnosti a v reálném čase upozorňují správce systému, když dojde k ohrožení integrity optické sítě [26].

Právě tímto problémem se zabývá spousta firem a jednou z nich je firma Viavi, která má na trhu několik zařízení, které slouží k odhalování závad a bezpečnostních rizik u optických síti.

2.4.1 Optická testovací jednotka OTU-8000

Optická testovací jednotka OTU-8000 kombinuje OTDR a technologii optických přepínačů a umožňuje nepřetržité OTDR monitorování více vláken kdekoliv v síti. Platforma OTU-8000 je plně modulární a umožňuje použití mnoha vlnových délek a dynamických rozsahů, aby bylo možné optimalizovat monitorování vláken pro měnící se síť. OTU-8000 rovněž dokáže měřit rozložení teploty a deformací ve vláknu. K zařízení se dá přistupovat pomocí webové aplikace a zařízení dokáže odesílat emaily a SMS při vzniku problému na optické trase [27].

2.4.2 ONMSi

Optical Network Management System (ONMSi) je software na který se dá napojit většina výrobku od Viavi. Tento systém dokáže přesně detekovat závadu a ukázat ji operátorovi sítě na mapě, tím se zkracuje celková doba výpadku sítě. Systém rovněž vyhodnocuje dlouhodobou výkonnost vláken a celkově optické sítě.

3 Komunikace mezi serverem a aplikací

Jedním z cílů této diplomové práce je prostudovat a následně navrhnout komunikaci mezi mobilní aplikaci, serverem a systémem pro monitorování optických vláken. Právě možnostem komunikace se tato kapitola věnuje.

3.1 Komunikační protokoly

Pro komunikaci přes internet se nejčastěji používá HTTP protokol, který je spjat s internetem od jeho počátku. HTTP protokol a jeho poddruhy si rozebereme v sekci 3.1.1. Druhým komunikačním protokolem, který se používá pro real time aplikace je WebSocket viz 3.1.2.

3.1.1 HTTP

Hypertext Transfer Protocol (HTTP) jedná se o protokol pro komunikaci s www servery. Nejčastěji se používá pro přenos hypertextových dokumentů s formátem HTML a XML, ale lze ho použit i pro přenos jiných typu souboru a dat. HTTP běží na portu 80 a jeho zabezpečená verze HTTPS na portu 443. Jedná se o bezstavový protokol, což znamená, že komunikace mezi dvěma uzly se skládá z nezávislých dvojic zpráv. Komunikace probíhá pomocí dotazování a to následovně, klient např. aplikace odešle dotaz na server a tento server mu odešle odpověď, kterou klient zpracuje. Princip jde vidět na obrázku 3.1.

HTTP je poměrně starý a jednoduchý protokol, jeho první verze byla nasazena v roce 1991. Z toho vyplývají některé nevýhody. První z nich je, že se jedná o poloduplexní spojení, což má za následek to, že každá strana musí čekat na dokončení akce protistrany. Další nevýhoda je velikost záhlaví, které zvětšuje velikost výsledné zprávy a prodlužuje režii. Jelikož se pro každou komunikaci sestavuje nové spojení, musí být záhlaví součástí každé zprávy.

Stavový kód HTTP je součásti odpovědi serveru na požadavek klienta a může nabývat 5 základních stavu. Stavový kód má vždy formát trojmístného čísla, písmena X ve výčtu označují další část kódu, která více specifikuje výsledný stav požadavku.

- 1XX – Informační,
- 2XX – úspěch,
- 3XX – přesměrování,
- 4XX – chyba klienta,
- 5XX – chyba serveru.

HTTP dlouhé dotazování

Dlouhé dotazování je velice podobné klasickému dotazování. Až na výjimku, že pokud server nemá nová data, které by klientovi poslal, čeká než jsou k dispozici data nová, nebo dokud nevyprší časový limit. Klient následně znovu může zaslat další dotaz na server. U dlouhého dotazování dochází ke zlepšení výkonu oproti klasickému dotazování, ale stále se využívá HTTP hlavička [28].

HTTP streaming

V případě HTTP Streamingu je server nakonfigurován tak, aby podržel konkrétní spojení mezi ním a klientem, aby přes něj mohly proudit data. Jakmile jsou na straně serveru k dispozici data nová, týkající se požadavku, server odešle odpověď prostřednictvím otevřeného kanálu a spojení uzavře pouze tehdy, když je to výslovně nařízeno. Tímto způsobem může klient naslouchat aktualizacím ze serveru a okamžitě je přijímat a to bez režie spjaté s otevíráním/zavíráním spojení. Na straně klienta také odpadá nutnost neustálého dotazování [29].

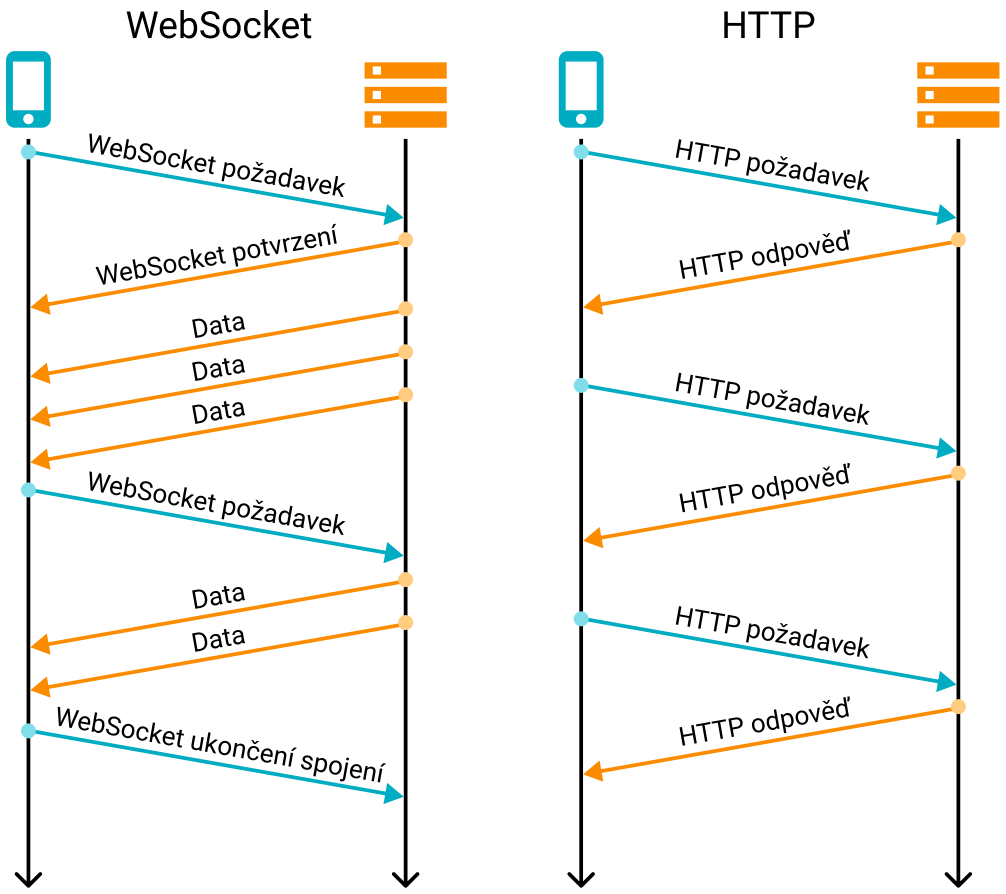
Aby HTTP streaming byl možný, musí server na požadavky klienta odpovídat tak, že nastaví příznak Transfer Encoding: chunked. Tím se nastaví trvalé spojení mezi serverem a klientem a server může odesílat data oddělené novými řádky. Tyto kousky dat pak může klient přijímat a zpracovávat za běhu, což klade na klienta vetší nároky, jelikož musí z nasekaných dat pochopit o co se jedná. Výhodou je, že po sestavení spojení už není potřeba HTTP hlavička pro přenos dat, jak tomu bylo u dotazování [29].

3.1.2 WebSocket

WebSocket byl vytvořen konsorciem World Wide Web Consortium (W3C) v roce 2008 a jsou součásti HTML5 standartu. WebSocket byl způsob jak nadále využívat výhod HTTP protokolu na aplikační vrstvě a zároveň využívat výhod Transmission Control Protocol (TCP) na transportní vrstvě. TCP má tu výhodu, že navazuje a udržuje spojení mezi dvěma body, dokud spojení jeden spoj neukončí. Právě této vlastnosti WebSocket využívá, není tedy nutno neustále navazovat nová spojení. Jednoduše se naváže jednou a následně si mohou jednotlivé body např. mobilní aplikace a server vyměňovat real time data, jedná se tedy o trvalé spojení neboli stream [30].

Pro vytvoření WebSocket spojení mezi aplikaci a serverem je stále využívaný protokol HTTP. Vytvoření spojení se taky někdy říká handshake. Po dokončení handshake, začne spojení využívat služeb TCP protokolu, který garantuje spolehlivé doručení všech segmentů. WebSocket spojení je plně duplexní, to znamená, že

obousměrná komunikace probíhá současně. Vytvoření spojení a komunikace mezi jednotlivými prvky je vidět na obrázku 3.1 pro porovnání je na obrázku znázorněna i HTTP komunikace. WebSocket využívá speciální URL adresu, kde předpona je ws, nebo wss pro zabezpečené připojení pomocí SSL. Komunikace pomocí WebSocket se nejčastěji využívána tam, kde je nutné, aby uživatel měl co možná nejčerstvější data [31].



Obr. 3.1: Porovnání WebSocket a HTTP komunikace [30].

3.1.3 WebRTC

WebRTC je nádstavba nad protokolem WebSocket a jedná se o komunikaci prohlížeče se serverem. Jeho hlavní využití je také pro real time komunikaci, ale je více soustředěn na hlasovou a video komunikaci.

3.1.4 WebTransport

WebTransport je rovněž nádstavba nad protokolem WebSocket, která vylepšuje spoustu věcí, jako je spolehlivost přenosu a zrychlení handshaku při navázaní komunikace. Tato technologie je ale stále v rané fázi vývoje a tudíž se nedoporučuje využívat pro produkční aplikace.

3.2 Formát dat

Poslední části kterou pro přenos dat potřebujeme je formát v jakém se data budou přenášet. Popíšeme si dva nejpoužívanější formáty a to JSON viz 3.2.1 a XML viz 3.2.2. Samozřejmě existují i další formáty jako například YAML, ale ty jsou velice podobné jako výše uvedené formáty a navíc nejsou tak rozšířené, takže se jimi nebudeme dále zabývat.

3.2.1 JSON

JavaScript Object Notation (JSON) je formát pro výměnu dat mezi aplikacemi. Pro lidi je jednoduchý pro zápis i čtení a pro stroje je jednoduchý na analýzu a generování. JSON je textový formát, který je zcela nezávislý na jazyku, ale používá konvence, které jsou známé programátorům jazyků rodiny C, Javy, JavaScriptu, Pythonu a mnoha dalších [32].

JSON struktura je postavená na dvojici jméno:hodnota. V různých jazycích se tato struktura realizuje jako objekt, záznam, struktura, slovník, hashovací tabulka, seznam s klíčem, asociativní pole, nebo jako mapa. Z tohoto důvodu se z JSON formátu stal nejoblíbenější formát pro výměnu dat, jelikož se velice jednoduše naparsuje na jakoukoliv výše uvedenou strukturu. Ukázku JSON struktury můžete vidět na obrázku 3.2.

3.2.2 XML

Extensible Markup Language (XML) je značkovací jazyk, který definuje soubor pravidel pro kódování dokumentů ve formátu čitelném pro člověka i pro stroj. Byl původně navržen pro potřeby rozsáhlého elektronického publikování, ale hraje také stále důležitější roli při výměně nejrůznějších dat na webu a mezi aplikacemi. Na obrázku 3.3 je vidět struktura XML souboru a jak je na první pohled patrné zápis XML se velice podobá HTML. XML rovněž jako HTML používá tagy, ale tady veškerá podobnost končí.

Tagy XML nejsou předdefinované jako tagy HTML, to znamená, že si můžeme vytvářet jakýkoliv tag uznáme za vhodné. XML bylo navrženo k přenosu dat se

Obr. 3.2: Struktura JSON formátu.

zaměřením na to, jaká data jsou. Na rozdíl od HTML, které bylo navrženo pro zobrazení dat se zaměřením na to, jak data vypadají [33].

Obr. 3.3: Struktura XML formátu.

4 Návrh komunikace

Tato kapitola se věnuje první praktické části diplomové práce. V praktické části bylo za úkol navrhnout způsob komunikace mezi serverem a mobilní aplikaci. V první části kapitoly je popsán formát dat a data, která bude systém odesílat do mobilní aplikace viz 4.1. V druhé části je popsána komunikace mezi mobilní aplikací a serverem viz 4.2.

4.1 Formát dat

Pro přenos dat se bude využívat JSON formát. JSON byl vybrán, jelikož celý server je napsaný v jazyce JavaScript. V JavaScriptu je velice jednoduché převádět objekty, se kterými se pracuje v kódu na JSON. JSON je rovněž dobře podporován i na straně mobilní aplikace, tedy v programovacím jazyce Dart viz 5.3. Server bude aplikaci vracet tyto parametry - potvrzení o ověření klíče, verzi firmwaru, list portů a jejich stav, informace o OTDR měření, list jednotlivých událostí, které na trase vznikly a jako poslední list naměřených hodnot pro OTDR graf. Příklad struktury dat je vidět na obrázku 4.1.

- isKeyVerified – Jedná se o potvrzení zda byl klíč pro komunikaci ověřen.
- fwVersion – Verze firmwaru, který v současné chvílí běží na monitorovacím systému.
- portList – Jedná se o list objektů, které reprezentují jednotlivé hardwarové porty na monitorovacím systému. Jednotlivé objekty následně obsahují dva atributy a to ID portu a jeho současný stav, ve kterém se port nachází. Port se může nacházet v jednom ze čtyř stavů:
  - OFF – Značí, že port je neobsazený a není využíván.
  - ON – Značí, že do portu je připojen kabel a je využíván.
  - ECHO – Je speciální stav ve kterém se port nachází, když zaznamená, že se optické vlákno pohnulo, nebo jinak se s ním manipulovalo.
  - ERROR – Tento stav značí, že port nefunguje správně.
- info – Jedná se o objekt který v sobě nese podrobné informace o OTDR měření. Je zde uvedeno celkem 22 atributů, které jsou uživateli v aplikaci k dispozici, je zde například uvedený čas a lokalita měření, vlnová délka a nebo typ optického vlákna.
- eventList – Jedná se o list objektů který reprezentuje události, které nastaly na trase během měření. Je zde uvedeno typ události, vzdálenost kde událost nastala a další podrobnosti.
- pointList – Opět se jedná o list objektů, které tentokrát reprezentují poslední měření pomocí OTDR. V objektu se nachází dva atributy, první je vzdálenost

```
{
    "isKeyVerified": true,
    "fwVersion": "1.0.0",
    "portList": [
        { "id": 0, "status": "OFF" },
        { "id": 1, "status": "ON" },
        { "id": 2, "status": "ECHO" },
        { "id": 3, "status": "ERROR" }
    ],
    "info": {
        "fiber type": "G.652 (standard SMF)",
        "wavelength": "1310 nm",
        "location A": "Drzovice",
        "location B": "Olomouc",
        "pulse width": "275 ns",
        "sample spacing": "0.0015625 usec",
        "index": "1.466000",
        "BC": "-79.40 dB",
        "loss thr": "0.020 dB",
        "refl thr": "-65.535 dB",
        "EOT thr": "5.000 dB",
        "datetime": 1380625552000,
        "maxX": 35.0,
        "maxY": 33.0,
        "minY": 0.0,
        "total loss": 10.575,
        "loss start": 0,
        "loss end": 29.042405,
        "ORL": 30.186,
        "ORL start": 0,
        "ORL finish": 29.042405,
        "range": 35.0488518055295
    },
    "eventList": [
        {
            "type": "1E9999LS {auto} reflection",
            "distance": "29.042",
            "slope": "0.328",
            "splice loss": "0.000",
            "refl loss": "-14.142"
        }
    ],
    "pointList": [
        { "distance": 0.509470, "power": 44.461000 },
        { "distance": 0.514564, "power": 44.458000 },
        { "distance": 0.519659, "power": 44.456000 }
    ]
}
```

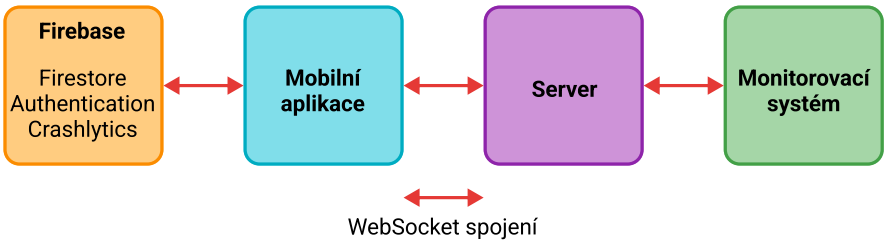

Obr. 4.1: Struktura dat.

v kilometrech. Druhý atribut je hodnota měření útlumu v dB. Tyto data jsou využívaná pro vytvoření grafu v mobilní aplikaci.

4.2 Komunikace

V celém komunikačním systému jsou čtyři prvky a to backend Firebase, mobilní aplikace, server a monitorovací systém. Celý tento systém je vidět na obrázku 4.2. Jelikož hlavním cílem diplomové práce je navrhnout a naprogramovat mobilní aplikaci, která bude komunikovat se serverem, je potřeba vybrat správný typ komunikace. Aplikace je určená pro platformy iOS a Android, proto se bude aplikace vyvíjet v multiplatformním frameworku Flutter. S ohledem na Flutter se musí vybrat komunikace, kterou Flutter podporuje.

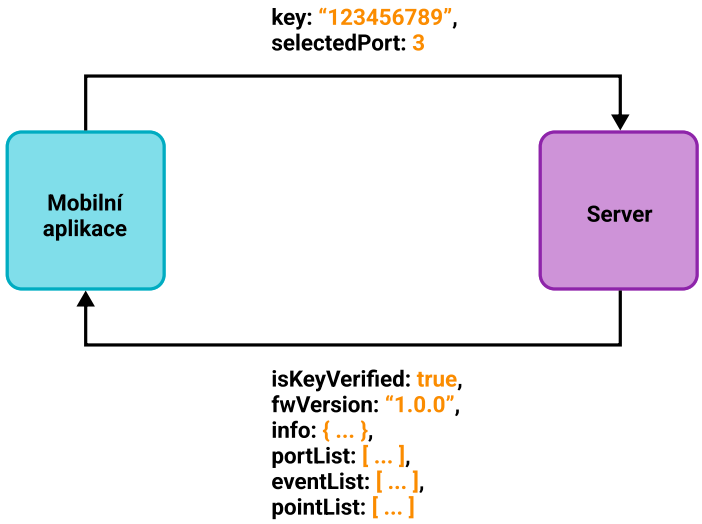
Flutter v základu podporuje jak HTTP tak i méně rozšířený protokol WebSocket. Proto lze v mobilní aplikaci využít oba dva zmíněné protokoly. Jelikož některé funkce jako třeba stav ECHO u portu je časově náchylný a chtěli bychom mít tento stav k dispozici v reálném čase, je vhodné zde využít protokol WebSocket. Dal by se využít i HTTP protokol, ale zde bychom se museli dotazovat několikrát za sekundu, což je dost neefektivní. Aplikace tedy bude komunikovat se serverem pomocí protokolu WebSocket. Druhá strana, s kterou bude aplikace komunikovat je backend aplikace, který poběží na platformě Firebase. Nativně je tato komunikace provozována přes protokol WebSocket. Poslední část komunikace je mezi monitorovacím systémem a serverem. Tato část v době psaní této diplomové práce nebyla dostupná, proto ji není dále v textu věnovaná pozornost. V budoucnu by zde měl rovněž být využit WebSocket protokol.



Obr. 4.2: Znázornění komunikačního systému.

Komunikace mezi mobilní aplikací a servem je vidět na obrázku 4.3. Tento obrázek nebere v potaz navazování samotné komunikace, ale již přenos dat. První zprávu odesílá mobilní aplikace a obsahuje dva parametry key a selectedPort. Klíč slouží k zabezpečení komunikace, aby nebylo možné číst data z jakéhokoliv serveru. V případě že server zjistí, že se klíč neshoduje s jeho klíčem, tak do aplikace zpět odesílá zprávu o tom že klíč nebyl ověřen. Zbylé atributy jako je portList, info atd. jsou prázdné. Následně server ukončí spojení s aplikací.

Aby server nemusel do aplikace zbytečně odesílat všechna naměřená data pro OTDR, je zde druhý parametr a to selectedPort. Tímto parametrem aplikace dává serveru najevo, že požaduje data pouze pro konkrétní port. Zde se připouští i možnost, že selectedPort bude rovný null. V tomto případě server aplikaci posílá prázdné atributy - info, eventList a pointList. Zbylé atributy objektu isKeyVerified, fwVersion a portList jsou vyplněné normálně aktuálními daty.



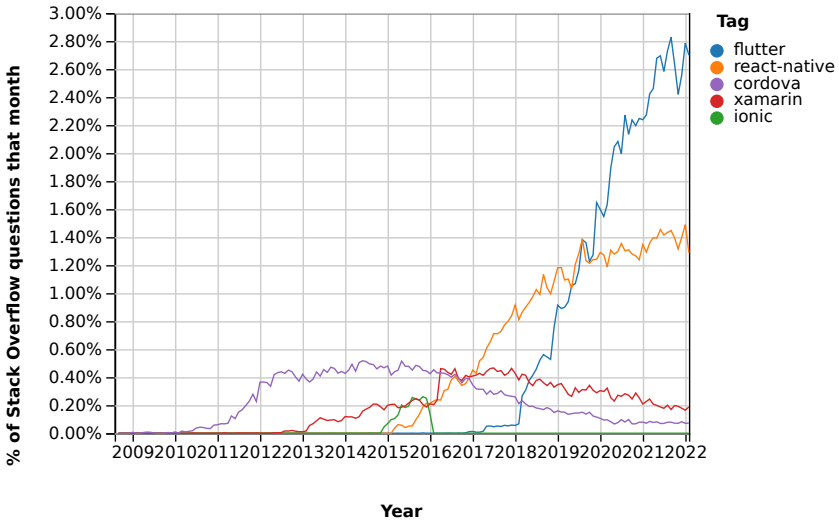
Obr. 4.3: Příklad komunikace mezi mobilní aplikací a serverem.

5 Flutter

V dnešní době, když chceme vyvíjet mobilní aplikaci, která má co největší zásah, tak potřebuje cílit na obě největší mobilní platformy Android a iOS. To ale přináší hodně problémů, jelikož každá z platforem má svá specifika a každá používá svůj vlastní programovací jazyk (Android – Kotlin, iOS – Swift). To znamená, že je potřeba napsat stejnou aplikaci pro každou platformu zvlášť, z toho vyplývá, že máme dva různé kódy, které dělají více méně to samé. Tudíž se musíme starat o dva kódy a každou úpravu musíme dělat rovněž dvakrát a to není efektivní.

Pravě tento problém se snaží vyřešit multiplatformní nástroje. Těchto nástrojů je celá řada, ale v praxi se využívají pouze dva a to Flutter a React Native. React Native vyvinula společnost Facebook a je odvozen od webového frameworku React. Jeho velká výhoda je ta, že pokud vývojář zná React a nebo má zkušenosti s webovým vývojem, bude se cítit jako doma. Primárním programovacím jazykem pro React Native je JavaScript.

Pro diplomovou práci byl vybrán framework Flutter. Jedním z důvodu proč byl vybrán je, že jeho popularita stále stoupá, jak můžeme vidět na obrázku 5.1. Podrobnější popis Flutter následuje v dalších sekcích této kapitoly.



Obr. 5.1: Počet vyhledaných otázek pro jednotlivé multiplatformní frameworky na stránce Stack Overflow.

5.1 Historie a směr Flutteru

Za frameworkem Flutter stojí společnost Google, jedná se o open-source framework pro tvorbu aplikaci. Jeho první verze byla vydaná v květnu 2017 a podporovala tvorbu aplikací pro mobilní zařízení Android a iOS. Cílem Flutteru je být opravdu multiplatformním nástrojem v pravém slova smyslu. V roce 2021 přidal podporu pro tvorbu webových aplikací a v roce 2022 podporu pro Windows aplikace. Do konce roku 2022 se počítá s podporou pro zbylé desktopové systémy macOS a Linux. Posledním krokem je podpora tzv. vestavěných zařízení jako jsou například řídící systémy v automobilech atd.

5.2 Architektura Flutteru

Architektura Flutteru je postavená na vrstvovém modelu. Na každé vrstvě se nacházejí jednotlivé moduly či knihovny, které pracují zcela nezávisle na sobě. Každá úroveň frameworku je navržena tak, aby byla volitelná a bylo ji možné nahradit [34].

První vrstvu tvoří tzv. Embedder, který je pojítkem mezi aplikaci a operační systémem, jelikož je Flutter multiplatformní tak je Embedder napsán v různých jazycích pro konkrétní platformu. Java a C++ pro Android, Objective-C/Objective-C++ pro iOS a macOS a C++ pro Windows a Linux. Finální aplikace napsaná ve Flutteru se pro operační systém tváří jako nativní aplikace a to právě díky Embedderu [34].

Druhá vrstva je engine Flutteru, ten je napsaný pomocí jazyka C++ a obsahuje všechna primitiva, které jsou nezbytná pro aplikace napsané pomocí Flutteru. Tato vrstva se stará o vykreslování jednotlivých snímku obrazu. Flutter má za cíl vykreslovat obraz při 60 snímcích za sekundu, používá se pro to výkony grafický engine Skia. Skia je open-source knihovna pro 2D grafiku, která poskytuje společné rozhraní API fungující na různých hardwarových a softwarových platformách [34].

Třetí vrstvu architektury tvoří samotný framework Flutter, který je napsaná pomocí programovacího jazyka Dart viz 5.3. S touto vrstvou se vývojáři potkávají nejčastěji. Obsahuje základní knihovny pro tvorbu UI (Material – Android, Cupertino – iOS). Dále jsou zde definované widgety, animace, gesta a kreslení různých tvarů [34].

5.3 Dart

Flutter používá programovací jazyk Dart a rovněž za ním stojí společnost Google. Dart je typově bezpečný, jelikož používá statickou typovou kontrolu. Systém typování v Dartu je ale rovněž flexibilní a umožňuje používat dynamické typy v kombinací s kontrolou za běhu.

Nedávno Dart přidal novou funkcionalitu null-safety. To znamená, že proměnné nemohou mít hodnotu null pokud to výslovně nedovolíme. Tato funkcionalita chrání vývojáře před pády programu, když se snaží pracovat s proměnnou, jejíž hodnota je null.

Dart má k dispozici dva typy kompilátorů které, se mění v závislosti na cílené platformě. Rovněž pro konkretní platformu se využívají rozdílné typy kompilaci, které se odvíjejí od toho, zda se jedná o produkční kód nebo kód určený pro vývoj [35].

- Native platform – Tento kompilátor je určen pro mobilní a deskopové aplikace. Využívá se zde JIT (just-in-time) kompilace pro potřeby vývoje. Tento typ kompilace umožňuje Flutteru využívat funkce hot reaload a hot restart. Obě tyto funkce zkracují prodlevy mezi sestavením nového kódu, který se následně vykreslí na obrazovku zařízení. Druhý typ kompilátoru je AOT (ahead-of-time), tento typ se využívá u finální kompilaci programu [35, 36].
- Web platform – Tento kompilátor se využívá pro webové aplikace a rovněž se zde využívají dva typy kompilací a to (dartdevc) pro vývoj a (dart2js) pro produkční kód [35, 36].

Dart nám dává tedy široké možnosti kompilace a je možné kompilovat jak do architektury x86 tak do ARM architektury.

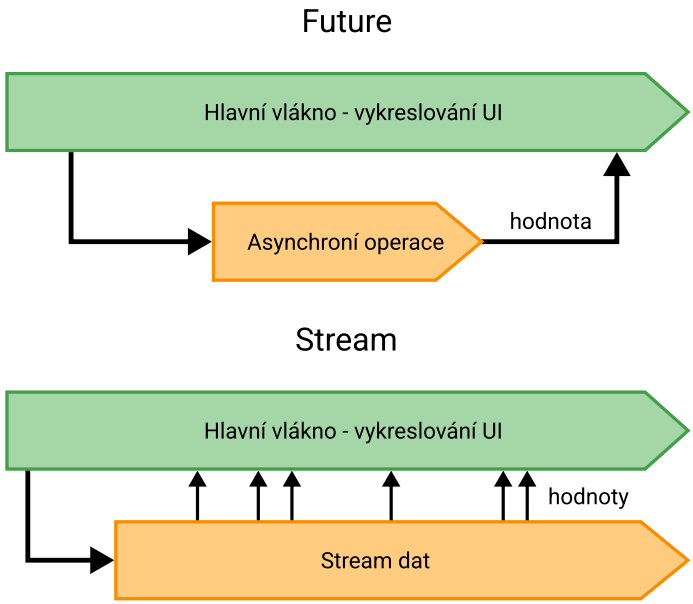
5.3.1 Reaktivní a asynchronní programování

Dart je velmi moderní programovací jazyk, který v základu podporuje jak asynchronní tak i reaktivní programování. Asynchronní programování nám umožňuje počkat na výsledek asynchronní operace, například stažení nových dat ze serveru, aniž bychom zablokovali hlavní vykreslovací vlákno. Takové zablokování hlavního jádra by se projevilo tím, že by aplikace zamrzla a uživatel by ztratil přehled o tom co se děje [35].

Dart nám pro tyto účely poskytuje datový typ Future, který reprezentuje výsledek asynchronní operace. Takto označené operace jsou přesunuty do druhého vlákna, kde se čeká na výsledek operace, který se předá do hlavního vlákna. Future je generická třída Future<T> kde T je datový typ výsledku operace. Future se nejčastěji využívá s klíčovými slovy async a await. Async se označují funkce které jsou asynchronní a slovo await nám říká, že než přejdeme a vykonáme další řádek kódu, musíme počkat než dostaneme výsledek operace [36].

Stream si můžeme definovat hodně způsoby, nejčastěji se uvádí příklad s potrubím, kdy na jednom konci do něj vložíme data a na druhé straně data opět získáme. V podstatě se jedna o FIFO (First In first Out) frontu. Víme, že pořadí dat se nemůže změnit, ale přesně nevíme kdy data dorazí. Pomocí Streamu můžeme

přenášet jednoduché datové typy i složité struktury. Streamem můžeme dokonce přenášet i další Stream. Porovnání Future a Streamu můžeme vidět na obrázku 5.2 [37].



Obr. 5.2: Porovnání fungování future a stream.

Flutter obsahuje widgety, které dokážou pracovat jak s asynchronními operacemi, tak i se streamy. Jedná se o widgety FutureBuilder a StreamBuilder. V závislosti na stavu dat, můžeme vykreslovat různé widgety. Například pokud čekáme na první data, můžeme uživateli vykreslit načítací obrazovku, nebo pokud při přenosu nastane chyba, tak chybovou obrazovku.

5.4 Platform channel

V případě kdy potřebujeme implementovat nějakou nativní funkcionalitu kterou Flutter, nepodporuje je možné využít platformní kanály. Platformní kanály nejsou nic jiného než výměna zpráv mezi kódem napsaným v Dartu a nativním kódem jako je Kotlin nebo Swift. Obsah zpráv je ve formátu JSON, který lze jednoduše v Dartu přetypovat na Mapu a následně i na objekt. Na straně nativního kódu jsou data přetypována na ekvivalent typu Map v Kotlin (HashMap) a ve Swiftu (Dictionary) [34].

Jak je vidět na výpisu 5.3 na prvním řádku si definujeme kanál, který se jmenuje "myChannel". Na druhém řádku vyvoláme metodu, která má dva parametry, první je jméno metody "flutterToNative"a druhý parametr jsou již samotná data. V našem

případě jde o Mapu dat. Jelikož se jedná o asynchronní operaci, tak klíčovým slovem await čekáme než nám dojde odpověď od nativního kódu.

```
1  const channel = MethodChannel("myChannel");
2  await channel.invokeMethod("flutterToNative", {"data": "Hello from Flutter"});
```

Obr. 5.3: Vytvoření platformního kanálu na straně Flutteru.

Tento systém výměny zpráv není nijak neobvyklý. Velmi podobný princip využívá např. framework Electron, který slouží pro tvorbu desktop aplikací pomocí webových technologii. Zde se tento systém výměny zpráv nazývá IPC (Inter-Process Communication).

5.5 Widgety

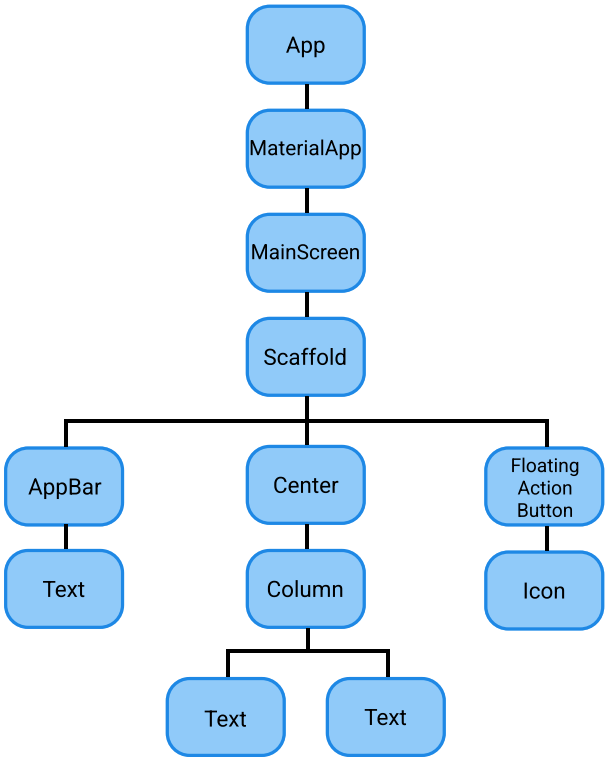
Základní stavební jednotkou pro UI aplikace ve Flutteru je widget. Widget může být jednoduchý text, tlačítko nebo velmi komplexní část UI jako je například widget Scaffold, který nám poskytuje základní strukturu celé obrazovky. Jednotlivé widgety se skládají do sebe a tím vzniká strom widgetů. Takový strom lze vidět na obrázku 5.4. Tento strom popisují i následující kódy pro stateful widget 5.5 a pro stateless widget 5.6. Jak můžeme vidět, některé widgety dokážou přijmout vice jak jeden další widget např. widget Column. Tento widget má parametr children, který není nic jiného než list widgetů. Tedy widget Column vykreslí jednotlivé widgety podle jejich pořadí pod sebe. Jedná se o velmi jednoduchý mechanizmus, kterým lze vytvářet komplexní uživatelské rozhraní [37, 34].

Když se pozorněji podíváme na výpis kódu z obrázku 5.6 tak si můžeme všimnout, že samotná třída MainScreen, kterou jsme vytvořili, dědí ze třídy StatelessWidget, tudíž i naše samotná třída MainScreen je widget.

Flutter nám umožňuje vytvářet dva druhy widgetů a to stateful widget a Stateless widget. Oba dva si níže podrobněji popíšeme.

5.5.1 Stateful widget

Kdykoliv widget bude obsahovat data, která se během jeho životního cyklu budou měnit, je vhodné ne však nutné použít stateful widget. Samotný widget je bez stavu, ale tento widget má přidružený objekt, který se nazývá state. Celý výpis stateful widgetu můžete vidět na obrázku 5.5 [37].



Obr. 5.4: Grafické znázornění stromu widgetů.

Stateful widget neobsahuje pouze metodu build(), ale rovněž může obsahovat pomocné metody, kterými lze řídit životní cyklus widgetu. Tři nejpoužívanější metody jsou:

- initState() – Voláno, když je widget vložen do stromu widgetu.
- dispose() – Tato metoda se volá, když je widget trvale odstraněn ze stromu widgetu.
- setState() – Oznámení, že se změnil vnitřní stav widgetu.

Stateful widget je ideální pro animace, jelikož animace není nic jiného než lokální stav, který se v průběhu času mění a je spjatý pouze se samotný widgetem. Rovněž zde jdou dobře využít metody initState() pro vytvoření AnimationController(), který řídí celou animaci a následně i dispose() pro zničení objektu když ho již nebudeme potřebovat.

5.5.2 Stateless widget

Stateless wiget je widget, který nespravuje svůj vlastní stav. Jakmile je tento widget sestavený pomocí metody build(), nelze jej měnit. Na první pohled se může zdát, že je tento widget úplné zbytečný, jelikož většina aplikací se v průběhu svého životního cyklu mění. To ale není pravda, jelikož data se ve widgetu měnit mohou, ale je

```dart
class MainScreen extends StatefulWidget {
  const MainScreen({Key? key}) : super(key: key);

  @override
  State<MainScreen> createState() => _MainScreenState();
}

class _MainScreenState extends State<MainScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("App Bar"),
      ),
      body: Center(
        child: Column(
          children: const [
            Text("Item 1"),
            Text("Item 2"),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {},
        child: const Icon(Icons.add),
      ),
    );
  }
}
```

Obr. 5.5: Jednoduchý stateful widget ve Flutteru.

to potřeba udělat jiným způsobem, než je tomu u stateful widgetu. Tento widget podporuje programátora, v tom aby správně od sebe rozdělil UI a logiku programu, která se například dotáže serveru o nová data. To má tu výhodu, že výsledný kód je přehlednější a lépe se čte. Příklad stateless widgetu je na obrázku 5.6 [37].

5.6 State

Stav není nic jiného než data, která chceme v daný okamžik zobrazit uživateli. Stav se dá rozdělit na dva druhy a to lokální a globální. UI se dá popsat pomocí vzorce na obrázku 5.7, kde UI se rovná funkci, která má jako parametr aktuální stav a vrací nové UI.

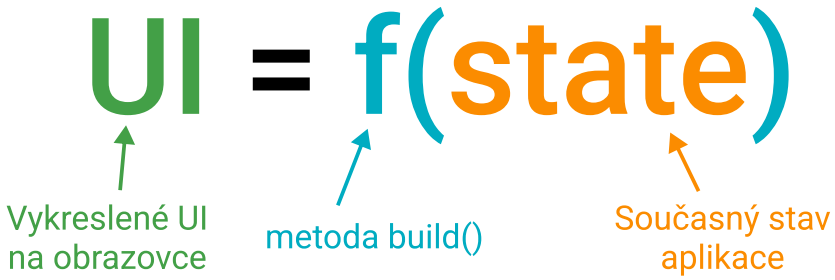
5.6.1 Local state

Lokální stav je stav, který je vázaný pouze k jednomu widgetu. Jedná se například o widget Switch, který může být ve stavu zapnuto, nebo vypnuto. Právě zde je

```dart
class MainScreen extends StatelessWidget {
  const MainScreen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("App Bar"),
      ),
      body: Center(
        child: Column(
          children: const [
            Text("Item 1"),
            Text("Item 2"),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {},
        child: const Icon(Icons.add),
      ),
    );
  }
}
```

Obr. 5.6: Jednoduchý stateless widget ve Flutteru.



Obr. 5.7: Vzorec uživatelského rozhraní [37].

doporučené používat stateful widget s metodou setState(), jelikož se jedná o lokální stav, který je vázaný pouze k tomuto widgetu [37].

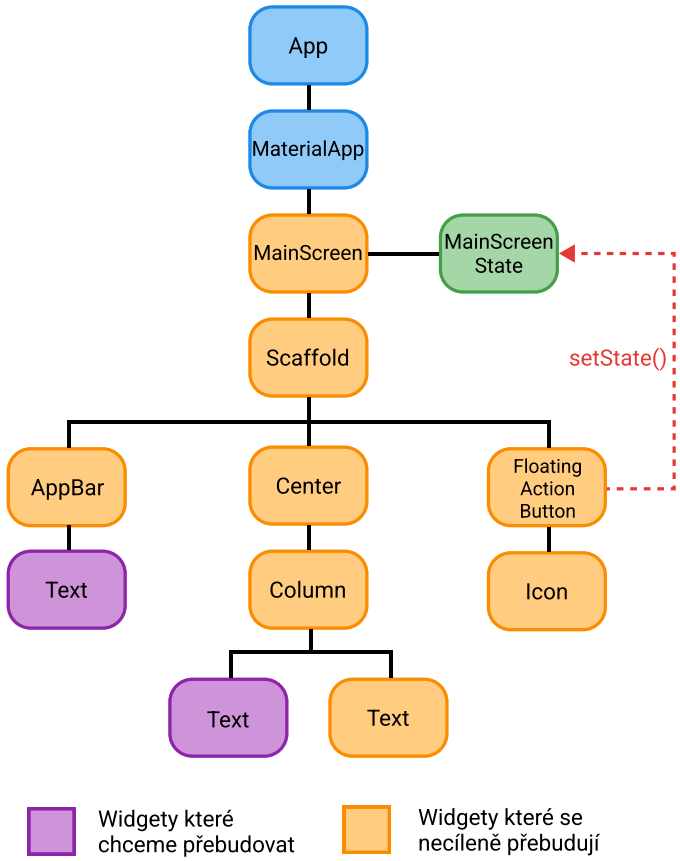
5.6.2 Global state

U globálního stavu začíná být situace složitější, jelikož je stav sdílený několika widgety. Příkladem může být situace, kdy si uživatel může vybírat mezi světlým a tmavým režimem zobrazení. O této změně se musí dozvědět všechny widgety, aby se

dokázaly překreslit. Tato situace je poměrně jednoduchá, jelikož tento stav hlídá widget MaterialApp, který je ve stromu widgetu kořenem, jak je vidět na obrázku 5.4 [37].

Jednoduchá poučka nám říká, že kdykoliv potřebujeme sdílet stav mezi více widgety, máme stav zvednout směrem nahoru. Stav by se měl nacházet ve widgetu, který je rodičem pro všechny widgety, které potřebují přístup ke stavu. Tento přístup jednoznačně funguje, ale má spoustu nedostatků.

Hlavním nedostatkem je, že widgety které sdílí společný stav, můžou být ve stromě widgetu různě hluboko. To má za následek, že Flutter nemůže efektivně cílit na konkretní widgety a dochází k přestavbě widgetů, které by se měnit vůbec neměly, jak je vidět na obrázku 5.8.



Obr. 5.8: Očekávaná a reálná přestavba stromu widgetu [37].

Dalším problémem je, že se kód stává velmi špatně čitelný a jednotlivé widgety jsou velmi těsně provazány, i když by měly být na sobě zcela nezávislé. Jeden z posledních problému je, že u takto provázaných widgetů se těžko testuje jejich vnitřní

logika. K výše uvedených problémům existuje celá řada řešení, celkově se této problematice říká state managment.

5.7 State managment

State managment je jednou z nejsložitější oblasti nejenom ve Flutteru, ale i celkově ve vývoji aplikací s uživatelským rozhraním. Flutter má výhodu obrovské komunity, která vymyslela spoustu originálních řešení a je pouze na programátorovy, které si vybere. Nejpoužívanější balíčky pro state managment jsou Provider, BLOC, Inherited Widget, Riverpod, GetX, Redux a mnoho dalších. Dále v textu je popsán Provider, jelikož je využit i v praktické části.

5.7.1 Provider

Provider je state managment řešení, které vychází z inherited widgetu, ale přidává nad ním další vrstvu abstrakce a zjednodušuje implementaci kódu. Celkově se jedná o jeden z nepopulárnějších balíčků pro Flutter. Dokonce sám Flutter doporučuje používat Provider jako hlavní state managment řešení. Abychom mohli používat ChangeNotifierProvider, je nutné udělat několik kroků. Jednotlivé kroky si ukážeme na jednoduché aplikaci, která inkrementuje počítadlo po stisku tlačítka. Na tomto příkladě je krásně vidět, jak se nám oddělí kód pro uživatelské rozhraní od kódu, který řídí vnitřní logiku.

Jako první je nutné vytvořit třídu, která bude spravovat vnitřní logiku. V našem případě inkrementuje hodnotu proměnné o jedničku. Jedná se o normální třídu, která dědí ze třídy ChangeNotifier. Jak se třída implementuje, je pouze na vývojáři. V našem případě třída obsahuje privátní proměnnou typu int s názvem _count. Podtržítko na začátku názvu značí, že se jedná o privátní proměnou. Následuje getter, který nám zpřístupňuje proměnou a metoda increment(), zde se provádí inkrementace proměnné. Jako poslední věc je nutné zavolat funkci notifyListeners(), pokaždé když chceme překreslit UI. Celou třídu můžeme vidět na výpisu 5.9.

Jako druhý krok je potřeba vytvořit samotného Providera. Bude se jednat o ChangeNotifierProvider, který překreslí UI kdykoliv se zavolá funkce notifyListeners(), kterou voláme když inkrementujeme naši proměnnou. ChangeNotifierProvider je widget, který má několik parametrů. Nejdůležitější parametr je create, zde říkáme, jaké data má Provider sledovat. V našem případě to bude třída, kterou jsme definovali v předchozím kroku. Pokud víme, že budeme data potřebovat na vice místech naši aplikace, tak je vhodné Providera definovat někde v blízkosti kořenu stromu widgetu. Balíček provider nám rovněž dovoluje definovat více providerů a to i různých

```
class CounterProvider extends ChangeNotifier {
  int _count = 0;

  int get count => _count;

  void increment() {
    _count++;
    notifyListeners();
  }
}
```

Obr. 5.9: Třída CounterProvider pro uchování stavu.

druhů jako třeba StreamProvider, který dokáže pracovat ze Streamy. Celý kód jde vidět na výpisu 5.10.

```
class App extends StatelessWidget {
  const App({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (_) => CounterProvider(),
      child: const MaterialApp(
        title: "Counter App",
        home: CounterScreen(),
      ),
    );
  }
}
```

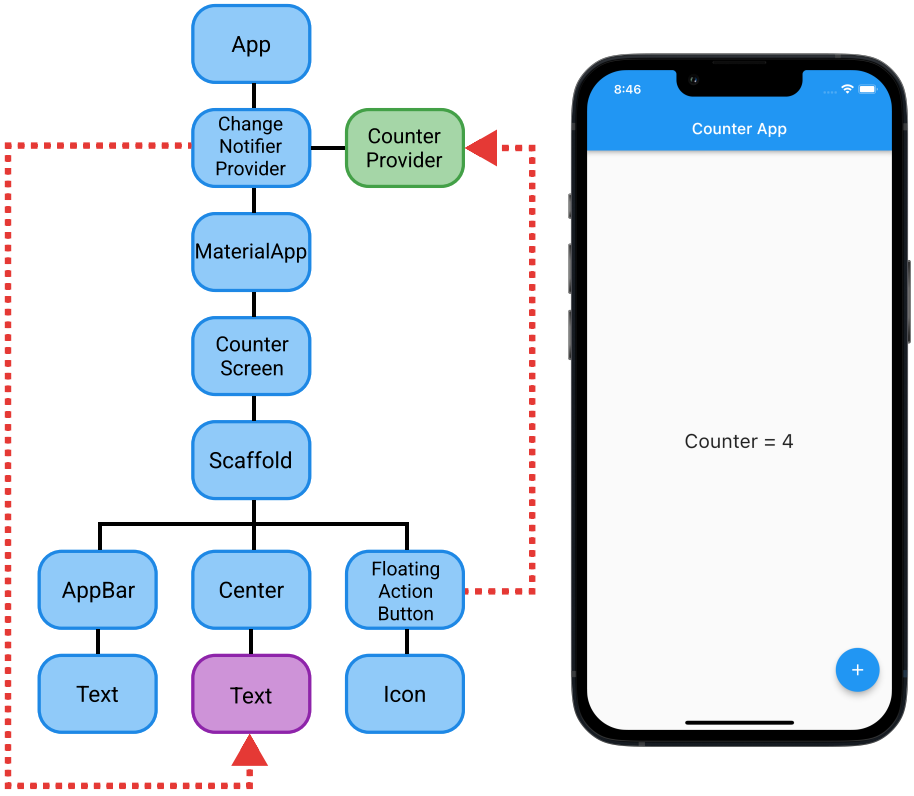
Obr. 5.10: Widget App s ChangeNotifierProvider pro správu stavu.

Posledním krokem je využívat Providera tam, kde potřebujeme. V tomto případě to bude pouze jeden widget, ale naprosto stejně se dá Provider používat v dalších widgetech, které mohou být různě hluboko ve stromu widgetu. Nejdříve je nutné definovat lokálního Providera, s kterým budeme pracovat ve widgetu. Pomocí tohohle Providera zavoláme na stisk tlačítka metodu increment() a zobrazíme aktuální stav pomocí getteru. Celý tento kód je vidět na výpisu 5.11 a strom widgetu spolu s výslednou aplikaci jsou vidět na obrázku 5.12.

```dart
class CounterScreen extends StatelessWidget {
  const CounterScreen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final _counterProvider = Provider.of<CounterProvider>(context);
    return Scaffold(
      appBar: AppBar(
        title: const Text("Counter App"),
      ),
      body: Center(
        child: Text("Counter = ${_counterProvider.count.toString()}"),
      ),
      floatingActionButton: FloatingActionButton(
        child: const Icon(Icons.add),
        onPressed: () => _counterProvider.increment(),
      ),
    );
  }
}
```

Obr. 5.11: Widget CounterScreen s využitím Providere pro správu stavu widgetu.



Obr. 5.12: Výsledný strom widgetu a výsledná aplikace.

6 Firebase

Firebase je v podstatě backend aplikace jak pro mobilní tak i pro webové. Firebase rovněž jako Flutter a Dart vlastní společnost Google. Z toho vyplývá, že propojení aplikace s Firebase je opravdu jednoduché. Firebase je s aplikaci spojený pomocí bundle ID, což je jedinečný identifikátor aplikace. Samotný Firebase se skládá z mnoha modulů, které lze jeden po druhém aktivovat či deaktivovat. Velká část modulů je zcela zdarma. Platí se pouze za složitější a náročnější moduly. Většinou se cena vypočítavá podle využití jednotlivých modulů, takže zde klasické předplatné nenajdeme. Komunikace mezi Firebase a aplikaci probíhá pomocí webSocket protokolu, který je detailně popsán v sekci 3.1.2.

Firebase nám nabízí nastroje na řešení nejčastějších problémů při vývoji aplikací, jako jsou autentizace uživatelů, implementace databáze, zasílaní notifikací atd. Dále v textu jsou podrobněji popsány čtyři moduly, které jsou využity i v mobilní aplikaci.

6.1 Autentizace

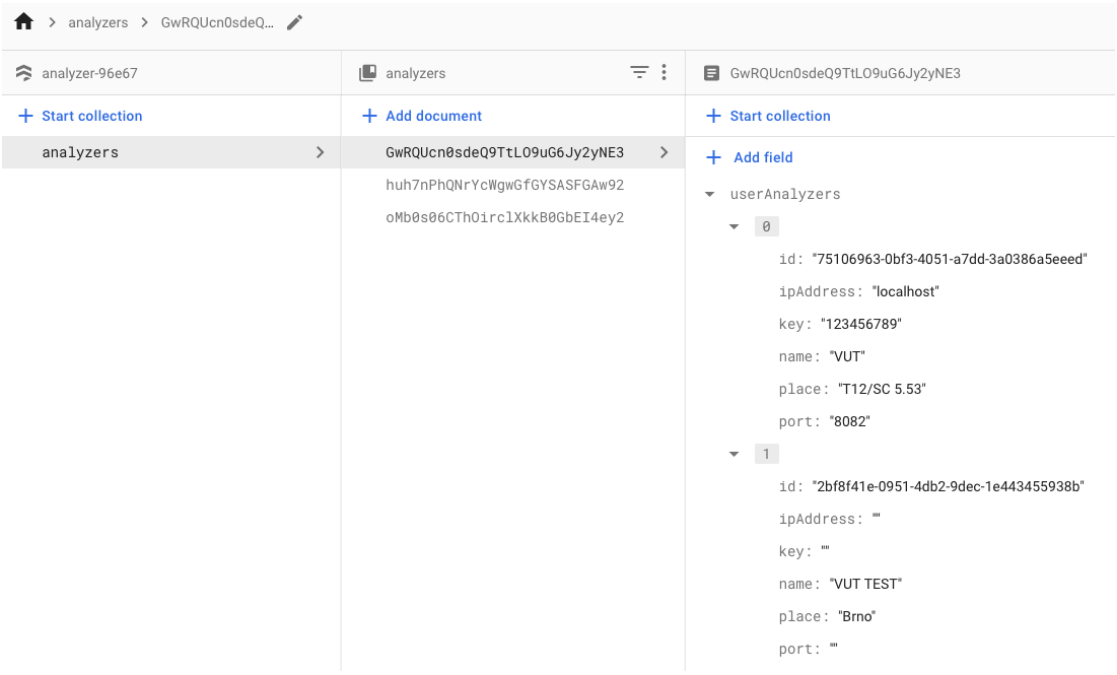
Autentizace uživatelů je jedna ze základních funkcionalit každé moderní aplikace, ale její implementace není jednoduchá. Firebase nám poskytuje velmi užitečný modul autentizace, který řeší většinu problémů, navíc se jedná o bezplatný modul. Lze zde využít od klasického přihlášeni pomocí emailu a hesla, přes mobilní číslo, až po anonymní přihlášení, které se hodí pro účely testování. Navíc nám Firebase nabízí využití přihlášení pomocí třetích stran, těmi mohou být Facebook, Google, Apple, Twitter a mnoho dalších.

Firabase nám poskytuje celou databázi uživatelů, kde můžeme jednotlivé uživatele editovat. Modul nám poskytuje i pokročilejší funkce jakou je obnova hesla, ověření či změna emailu. Obsah emailu, který je zasílán uživatelům např. z důvodu obnovy hesla, lze libovolně upravovat. Pro ověření uživatelé lze využít i dvoufaktorové ověření, kdy uživateli přijde na jeho telefonní číslo SMS zpráva s kódem pro ověření.

6.2 Firestore

Firestore je noSQL databáze nebo také nerelační databáze. U klasické SQL databáze jsou data uložená do tabulek. NoSQL databáze jsou strukturovaná odlišně. Existuje celá řada typů noSQL databází, ale mezi ty nejpoužívanější patří databáze klíč–hodnota, databáze sloupců, grafová databáze anebo databáze dokumentů. Právě databázi dokumentů využívá Firestore. Data jsou zde uloženy v dokumentech a následně jsou tyto dokumenty ukládané do kolekcí, tato struktura jde vidět na

obrázku 6.1. Jednotlivé dokumenty podporuji mnoho různých datových typů a jsou strukturované jako JSON. V rámci dokumentů můžeme také vytvářet podkolekce. Mezi hlavní výhody noSQL databází patří snadný návrh, škálovatelnost a rychlost [38] [39].

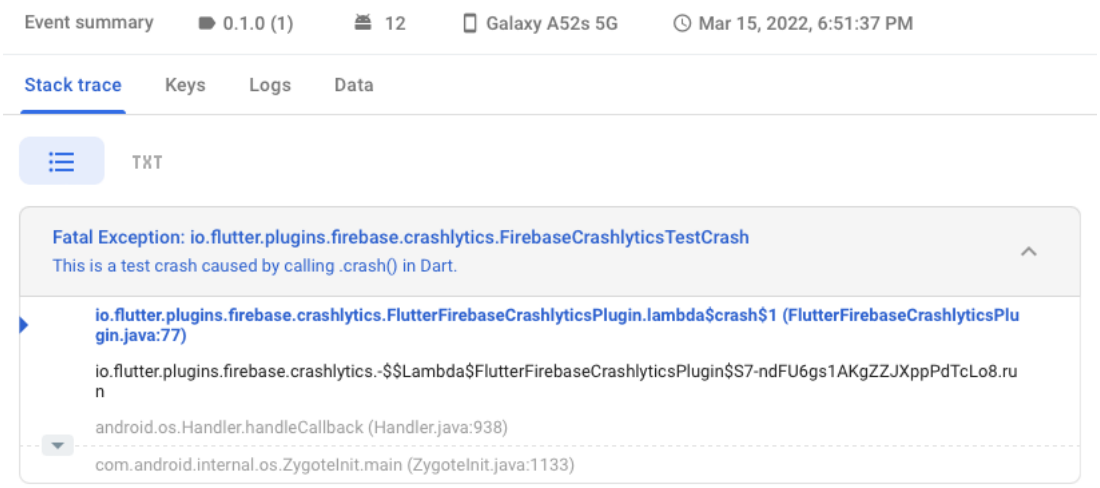


Obr. 6.1: Modul Firestore, databáze záznamů pro jednotlivé uživatele.

6.3 Crashlytics

Tento modul nám umožňuje sbírat záznamy od uživatelů, kterým se aplikace chová nestandardním způsobem. Crashlytics nám dokáže zasílat kompletní výpis chybového hlášení či výjimky, která při běhu aplikace nastala. Jak můžeme vidět na obrázku 6.2, tak nám Firebase přehledně zobrazí kompletní log spolu s verzi aplikace, verzí operačního systému, typem a značkou zařízení a časem výskytu chyby. V podrobnějším výpisu můžeme nalézt jakou má uživatel k dispozici velikost RAM paměti, i jak velkou část měl obsazenou při výskytu chyby.

Chyby jsou odesílány na Firebase okamžitě po výskytu, tedy pokud se jedná o chybu, která nezpůsobila pád aplikace, v tomto případě je chyba zaslána po opětovném spuštění aplikace.



Obr. 6.2: Modul crashlytics, výpis logu.

6.4 Distribuce aplikace

Firebase obsahuje modul pro distribuci aplikace, který lze využít pro testování. Myšlenka je jednoduchá: my jako vývojáři aplikací nikdy nebudeme schopní otestovat aplikaci na všech různých zařízeních, s různými verzemi operačního systému. Tento modul nám to ulehčuje tím, že nám dává jednoduchý nástroj, jak dostat naši aplikaci mezi velký počet testerů.

Do modulu stačí nahrát vyexportovanou aplikaci a následně přidat popis co jsou hlavní změny v současné verzi aplikace a co by měli testeři hlavně testovat. Jako poslední krok je přidat emaily testerů, kterým chceme současnou verzi poskytnout. Modul nám umožňuje vytvářet jednotlivé skupiny testerů a následně je i spravovat. Rovněž můžeme vygenerovat veřejný odkaz, kterým se do testování může zapojit široká veřejnost.

Tento modul skvělé funguje s předchozím modulem Crashlytics, jelikož nám testeři nemusí poskytovat rozsáhle zprávy o tom, jak a kdy došlo k pádu aplikace.

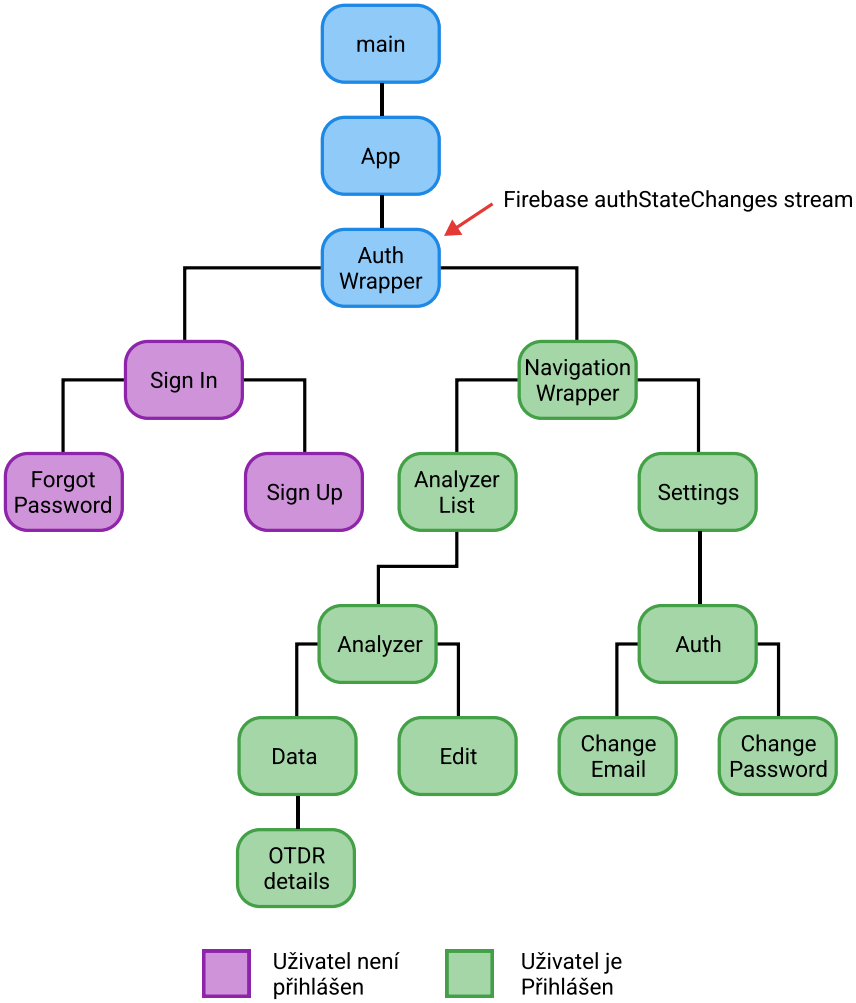
Distribuce funguje jak pro Android tak pro iOS, ale co se týče iOS, je proces poměrně zdlouhavý a ne moc přívětivý. Nicméně jde pro testování iOS aplikaci využít službu TestFlight, která je přímo zabudována do služby App Store Connect, odkud se vydávají aplikace určené pro normální uživatele.

7 Mobilní aplikace

Tato kapitola se věnuje hlavní části diplomové práce a to mobilní aplikaci pro platformy Android a iOS. Tato aplikace má za cíl uživateli zobrazovat realtime data z analyzátoru, ke kterému je vzdáleně připojena. První část kapitoly se věnuje architektuře aplikace viz 7.1. Druhá část je věnována struktuře projektu viz 7.2. Další části této kapitoly jsou věnované jednotlivým obrazovkám aplikace a jejich implementaci.

7.1 Architektura aplikace

Celá aplikace se dá rozdělit na dvě části a to pokud uživatel je nebo není přihlášený. Celá struktura aplikace jde vidět na obrázku 7.1.



Obr. 7.1: Architektura aplikace.

První modrý obdélník označuje funkci main(), kde dochází k inicializaci a načtení lokálních dat pomocí balíčku Shared Preferences, který nám umožňuje ukládat uživatelské nastaveni (lokalizaci, tmavý světlý režim, nastavení grafu) do paměti telefonu. Když dojde k zabití aplikace, tak aplikace při opětovném spuštění muže načíst uživatelské nastavení. Také zde dochází k inicializaci druhého balíčku a to Package Info Plus, který nám dovoluje získat verzi a build aplikace, které se následně zobrazují v nastavení. Jako poslední se zde provede volání funkce runApp(), kde jako parametr předáme naši aplikaci s názvem App.

App již není funkce, ale první widget naši aplikace. Dochází zde k inicializaci a připojení k Firebase. Jelikož se zde čeká na spojení s Firebase, což je asynchronní operace, tak je celý widget zabalený do FutureBuilder. Tento widget nám umožňuje v závislosti na stavu spojeni zobrazovat různé obrazovky. První varianta, která může nastat je ta, že nám může Firebase vrátit chybu, v tomto případě uživateli zobrazíme chybovou obrazovku. Druhá varianta je, že se povedlo navázat spojení, tudíž uživateli můžeme zobrazit stránku podle toho, jestli je nebo není přihlášený, viz další odstavec. Zde rovněž dochází k odesílání chybových hlášení na službu Crashlytics v případě, že aplikace spadla. Poslední varianta je ta, že připojování stále probíhá a tedy uživateli zobrazíme obrazovku s indikátorem načítaní.

V případě, že dojde k úspěšnému připojení, se nejdříve kontroluje zda je, nebo není uživatel přihlášen. Pro tento účel je zde authStateChanges stream, který nám vrací aktuální stav uživatele. Zde dochází k větvení aplikace, pokud uživatel není přihlášen, například po první instalaci je mu zobrazená fialová část na obrázku 7.1, kde se muže přihlásit nebo se registrovat. Pokud se uživatel úspěšně přihlásí, authStateChanges stream vrátí první uživatelova data a my je můžeme zobrazit. Z toho vyplývá, že samotný AuthWrapper widget je zabalen do StreamBuilder, který poslouchá příchozí data a podle naši logiky zobrazuje další widgety.

7.1.1 State management

Pro state managment je v aplikaci využito řešeni Provider, které je podrobně popsáno v sekci 5.7. Aplikace využívá pět Provideru, čtyři ChangeNotifierProvider a jeden StreamProvider. Tento StreamProvider pracuje se streamem dat z databáze Firestore. Další čtyři Provider jsou:

- ThemeProvider – Umožňuje změnu tmavého/světlého režimu aplikace.
- LocalizationProvider – Umožňuje změnu jazyka aplikace.
- NavigationProvider – Umožňuje přecházet mezi obrazovkami.
- ChartProvider – Umožňuje změnu nastavení grafu s OTDR daty.

Pro vytvoření všech Provideru je využit widget MultiProvider, který nám dovoluje definovat více Provideru najednou. V podstatě se jedná o pole Provideru, jak je

vidět na výpisu 7.2. Druhý parametr child už je widget MatterialAppWithTheme, který tvoří kostru celé aplikace.

```
return MultiProvider(
    providers: [
      ChangeNotifierProvider(create: (_) => LocalizationProvider()),
      ChangeNotifierProvider(create: (_) => ThemeProvider()),
      ChangeNotifierProvider(create: (_) => NavigationProvider()),
      ChangeNotifierProvider(create: (_) => ChartProvider()),
      StreamProvider<List<AnalyzerModel>>(
          create: (_) => FirestoreService().analyzerListStream(), initialData: const [])
    ],
    child: const MatterialAppWithTheme(),
);
```

Obr. 7.2: Definice všech Provideru v aplikaci.

7.2 Struktura projektu

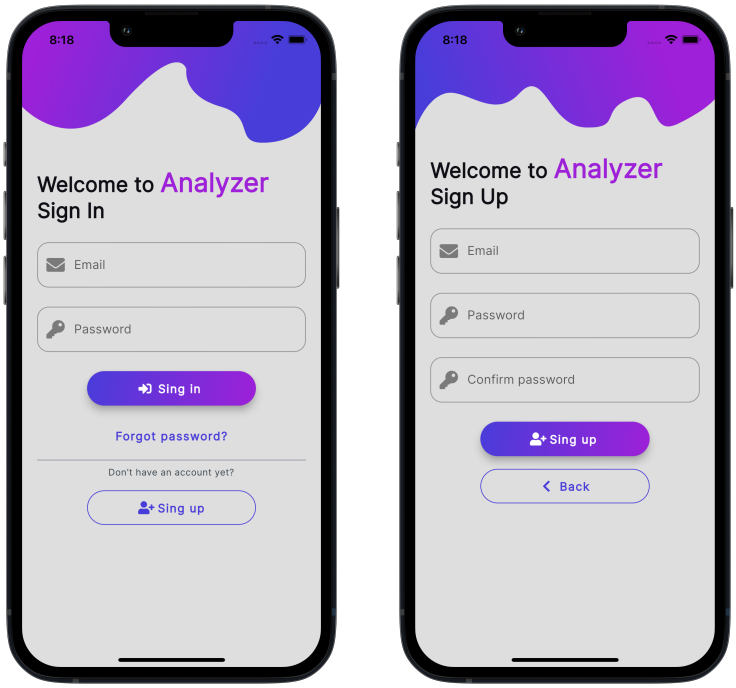
Po vytvoření nového projektu nám Flutter vygeneruje kompletní strukturu projektu, která se skládá z několika souboru a složek. Jedním z hlavních souborů, se kterým se během vývoje aplikace pracuje, je pubspec.yaml. V tomto souboru jsou definovány všechny balíčky, s kterými chceme pracovat a nejsou součásti Flutteru. Je zde definovaná verze aplikace, i jakou verzi programovacího jazyka Dart používáme. Dále je možné zde definovat složka s názvem assets, kde se dají ukládat obrázky a další věci které nejsou přímo kód.

Dále nám Flutter vygeneruje dvě složky s názvem android a ios. Jedná se o nativní část projektu, kde se definuje například ikona aplikace, nebo oprávnění pro používaní Bluetooth, kamery nebo GPS. A jelikož jsou tyto věci definované v každém operačním systému jinak, je potřeba je udělat pro každý systém zvlášť.

Nejdůležitější část projektu je složka lib. Jedná se o Flutter část projektu a zde jsou uloženy všechny soubory spojené s naší aplikací.

7.3 Obrazovky s přihlášením

Jako první obrazovku po spuštění aplikace uživatel uvidí právě obrazovku s přihlášením. Zde může zadat své přihlašovací údaje. Rovněž se odtud může dostat na dvě další stránky a to na stránku registrace a na stránku zapomenutého hesla. Pro všechny tyto funkcionality se používá modul autentizace z Firebase viz 6.1. Obrazovky přihlášení a registrace jsou vidět na obrázku 7.3.



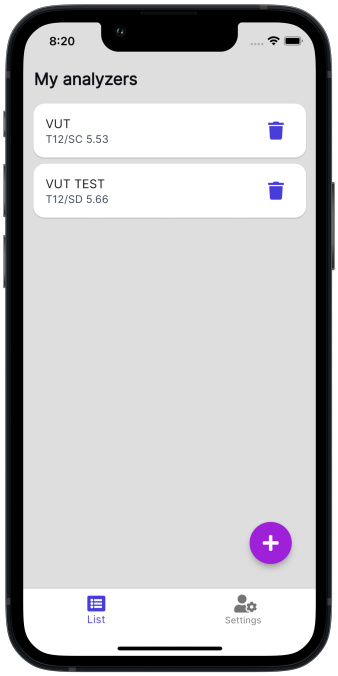
Obr. 7.3: Obrazovky přihlášení a registrace.

V případě, když uživatel zadá špatné heslo, email nebo při registraci zadá slabé heslo, tak jej aplikace upozorní vyskakovacím widgetem Snackbar. V tomto widgetu je popsána chyba, všechny tyto chyby jsou rovněž lokalizovány do obou podporovaných jazyku, tudíž čeština a angličtina.

7.4 Obrazovka se seznamem analyzátorů

Hlavní funkcionalita aplikace je spravovat jednotlivé analyzátory a vzdáleně se na ně připojovat. Všechno začíná vytvořením analyzátoru, kde je uživatel vyzván, aby zadal jméno analyzátoru. Uživatel může mít libovolné množství analyzátoru ve své správě, a není podmínkou že jméno musí být jedinečné, jelikož aplikace při vytváření objektu mu přiřadí jedinečné ID. Po vytvoření nového analyzátoru se uživateli objeví nová karta v jeho seznamu, která reprezentuje nový analyzátor, jak je vidět na obrázku 7.4. Uživatel má rovněž možnost jednotlivé analyzátory odstranit.

Všechny analyzátory jsou uloženy v databázi Firestore na backendu aplikace, takže pokud se uživatel přihlásí z jiného zařízení přes svůj účet, bude mít všechny své analyzátory k dispozici. Data jsou v databázi strukturována následujícím způsobem. V kolekci s názvem analyzers jsou vytvářeny jednotlivé dokumenty, kde jejich



Obr. 7.4: Obrazovka se seznamem analyzátoru.

název odpovídá ID uživatele, které mu bylo přiřazené během registrace. Tím máme pevně svázaného uživatele a jeho data v databázi. Pokud se podíváme přímo do dokumentu, zde nalezneme jediné pole s názvem userAnalyzers, kde jsou chronologicky na jednotlivých pozicích uloženy uživatelovy analyzátory. Celá struktura databáze je vidět na obrázku 6.1. Každý analyzátor má šest vlastností:

- id – Jedinečný identifikátor analyzátoru.
- name – Jméno analyzátoru.
- place – Poloha analyzátoru např. místnost atd, jedná se o nepovinné pole.
- ipAddress – IP adresa slouží k připojení k analyzátoru.
- port – Port služby slouží k připojení k analyzátoru, jedná se o nepovinné pole.
- key – Klíč který slouží k ověření uživatele na straně serveru.

7.5 Obrazovka analyzátoru

Po kliknutí na kartu analyzátoru se uživateli zobrazí obrazovka, která obsahuje dvě záložky data a editace. Aplikace se pokusí připojit na server, který je definovaný IP adresou a portem. Připojení a následná komunikace běží přes WebSocket protokol viz 3.1.2. Pokud uživatel nezadal IP adresu, port, nebo nastala v komunikaci

chyba, uživateli se zobrazí chybová obrazovka v záložce data, kde je popsána závada popřípadě celý výpis chyby.

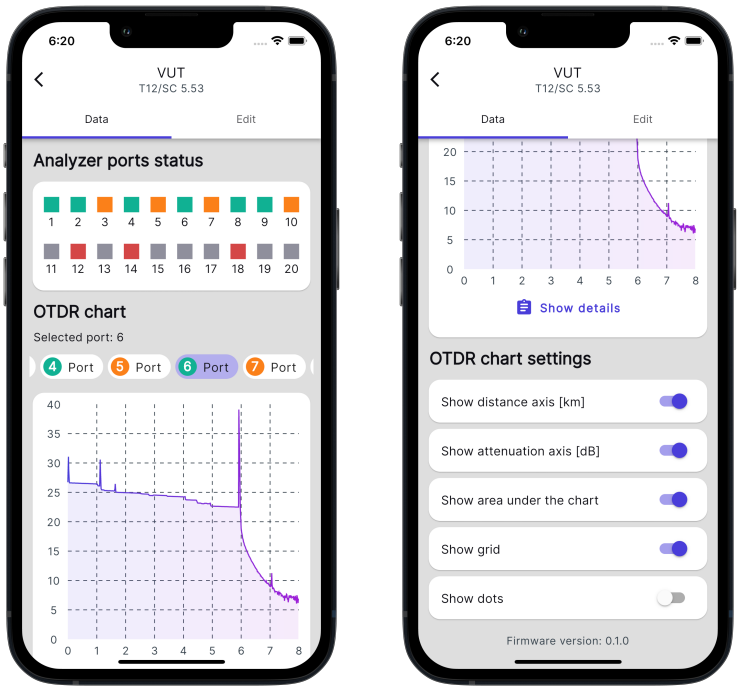
7.5.1 Záložka data

Na této záložce má uživatel k dispozici všechny data z analyzátoru. Nachází se zde dva hlavní widgety a to stav portů a OTDR graf s nastavením, navíc je úplně dole na stránce uvedená verze firmwaru analyzátoru. Celou obrazovku můžeme vidět na obrázku 7.5.

Widget stav portů

Tento widget zobrazuje uživateli aktuální stav jednotlivých portů na analyzátoru. Port se může nacházet v jednom ze čtyř stavů, kde každý stav je symbolizován jednou barvou:

- ON – Zelená barva.
- OFF – Šedá barva.
- ECHO – Oranžová barva.
- ERROR – Červená barva.



Obr. 7.5: Obrazovka analyzátoru, záložka data.

Widget graf OTDR s nastavením

Jelikož všechny porty jsou schopné provádět OTDR měření, tak si uživatel může vybrat, ze kterého portu chce data získat. Data jdou získat pouze z portů, které jsou ve stavu OK (zelená) nebo ECHO (oranžová). Pro výběr správného portu se zde nachází list portů, kde uživatel vidí číslo portu a v jakém stavu se nachází. Po kliknutí na vybraný port, aplikace zašle zprávu serveru, který port a jeho data chce získat.

Graf zobrazuje výsledky z posledního měření OTDR, který má server k dispozici. Během OTDR měření se vytváří řádově statisíce vzorků, které ovšem nemá smysl všechny přenášet k uživateli a to z několika důvodů. Za prvé by přenos dat dlouho trval, za druhé by byl datově náročný. Samotné zobrazení tolika dat by taky nebylo možné, jelikož dnešní displeje mají na šířku rozlišení zhruba 1200 pixelů. Když do toho ještě započítáme odsazení jednotlivých widgetu, tak se můžeme dostat na hodnotu 900 až 1000 pixelů. Zobrazovat jeden vzorek na pixel taky nedává smysl, proto se do aplikace odesílá pouze 500 vzorků. Server musí vždy data před samotným odesláním upravit. Pod samotným grafem je k dispozici tlačítko pro zobrazení detailů o měření, které uživatele přesune na další stánku viz 7.5.2.

Uživatel si může hodnoty útlumu a vzdálenosti prohlížet tím, že příloží prst na graf a zobrazí se mu aktuální hodnota na kterou ukazuje. Uživatel si rovněž může graf přizpůsobovat, všechny tyto změny se ukládají do shared preferences, tudíž uživatel o tyto změny nepřijde ani po vypnutí aplikace.

7.5.2 Obrazovka s detaily o OTDR měřením

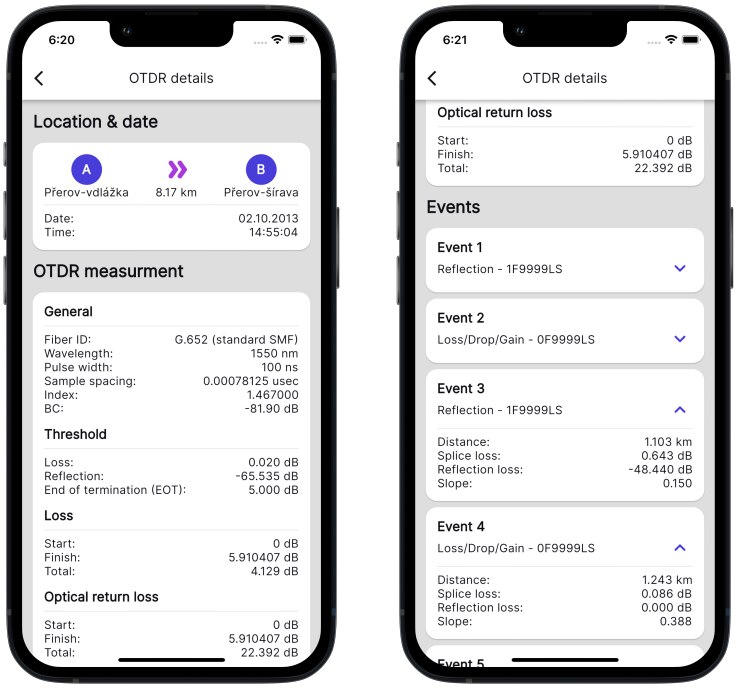
Tato obrazovka dává uživateli podrobnější informace ohledně OTDR měření. Obrazovka je rozdělená na tři hlavní sekce. Celá tato obrazovka je vidět na obrázku 7.6.

Widget lokalita a datum

Tento widget zobrazuje lokalitu A a lokalitu B kde se měření provádělo a jaká vzdálenost je dělila. Také je zde uvedeno datum i přesný čas měření.

Widget měření OTDR

Na tomto widgetu může uživatel vidět podrobné informace, které se týkají přímo OTDR měření. Jsou zde informace jako typ optického vlákna, použitá vlnová délka, délka pulzu, celkové hodnoty útlumu, prahové hodnoty a nebo hodnoty optického zpětného útlumu.



Obr. 7.6: Obrazovka s podrobnými informacemi o OTDR měření.

Widget událostí

Tato sekce zobrazuje jednotlivé události, které na trase nastaly. Jedná se hlavně o odrazy, ztráty a propady. Uživatel si jednotlivé widgety může rozbalit a získat více informací, např. vzdálenost na které událost nastala, nebo útlum který tato událost doprovázela.

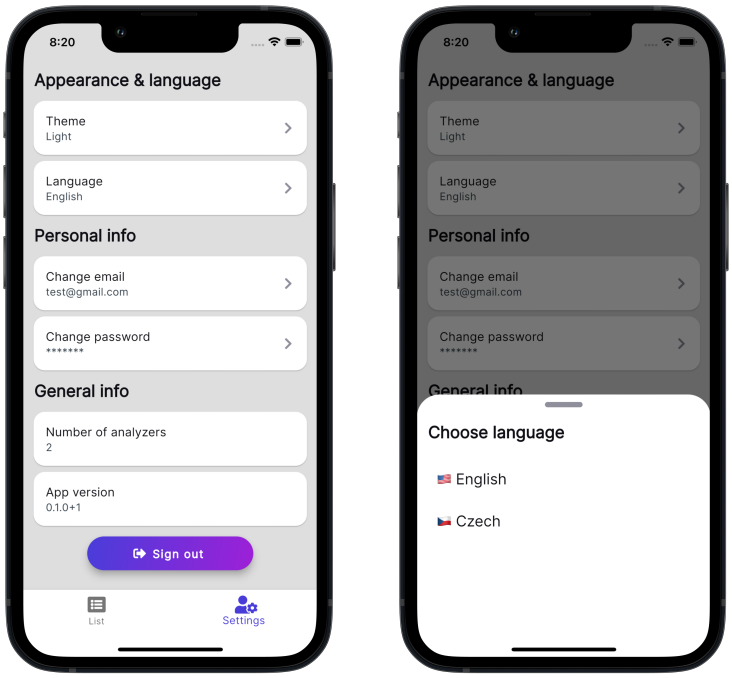
7.5.3 Záložka editace

Zde si uživatel může editovat anylyzér, po vytvoření nového analyzátoru je zde vyplněné pouze jméno. Ostatní pole si uživatel doplňuje právě zde, může zde doplnit umístění analyzátoru, IP adresu a port. Rovněž se zde zadává klíč k analyzátoru, tento klíč server při každém dotazu kontroluje. V případě že klíč není správný, server se automaticky od aplikace odpojí a v aplikaci se objeví chybová hláška.

Po dokončení editace se zde nachází tlačítko uložit, které odešle data do databáze Firestore. Jelikož aplikace a Firestore komunikují pomocí WebSocket protokolu, tak se změny projeví v aplikací okamžitě.

7.6 Obrazovka nastavení

Tato stránka dovoluje uživateli si aplikaci přizpůsobit, změnit přihlašovací údaje, nebo se i z aplikace odhlásit. Uživatel zde vidí kolik analyzátorů má vytvořených a rovněž jakou verzi aplikace používá. Celá obrazovka nastavení je vidět na obrázku 7.7.



Obr. 7.7: Levý obrázek: Obrazovka nastavení. Pravý obrázek: výběr lokalizace aplikace.

7.6.1 Lokalizace

Uživatel má možnost si v nastavení přepnout jazyk aplikace. Na výběr má ze dvou možnosti – výchozí nastavení po instalaci je angličtina, druhou možností je čeština.

Samotné překlady tvoří dva soubory s příponou arb. Jedná se o speciální soubory pro lokalizaci, které jsou založeny na JSON. Jeden soubor je určený pro anglické překlady a druhý pro české. Samotná struktura je klíč: překlad, tudíž klíče jsou ve všech souborech stejné a liší se pouze překlad. Proto přímo v kódu samotné aplikace se volá jen samotný klíč a podle toho jakou současnou hodnotu má nastavenou widget MaterialApp (en nebo cs), tak se použije správný překlad.

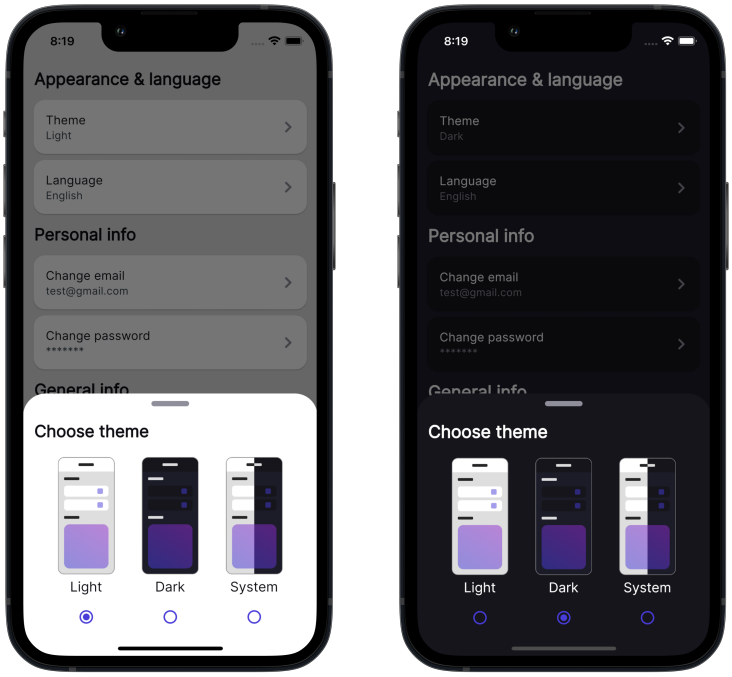
Aby bylo možné tyto překlady využít, je nutné ve widgetu MaterialApp, který tvoří celou kostru naši aplikace, vyplnit několik parametrů. Jedním je supportedLocales, kde aplikaci řekneme jaké lokalizace podporujeme. Dalším parametrem je locale, tento parametr určuje jaký jazyk má zrovna uživatel zvolen. Tento parametr si uživatel volí na widgetu, který je vidět na obrázku 7.7. Proto, abychom byli schopní změnit jazyk, je zde použit LocalizationProvider. Ten předá potřebné informace od widgetu kde si uživatel vybírá jazyk až k widgetu MaterialApp, který umožní použití správného překladu a přeložení aplikace.

7.6.2 Změna hesla a emailu uživatele

Uživatel si může během používaní aplikace změnit email nebo heslo. Pro obě tyto možnosti je na obrazovce nastavení připraveno tlačítko. Než dojde k samotné změně emailu nebo hesla, tak je po uživateli požadováno znovu přihlášení pro kontrolu jeho identity. V případe kladného ověření uživatele, je přesunut na další stránku, kde si své údaje může změnit.

7.6.3 Tmavý/světlý režim

Další funkcionalitou, kterou aplikace uživateli nabízí, je změna vzhledu aplikace. Uživatel má na výběr ze tří možností a to světlý, tmavý režim nebo systémový, zde se aplikace řídí podle operačního systému telefonu. Pro změnu režimu je zde opět použit provider, který předává potřebná data widgetu MaterialApp, který má na starost změnu vzhledu všech widgetů. Widget MaterialApp má opět k dispozici několik parametrů. Prvním z nich je theme, kde předáváme třídu ThemeData z názvem lightTheme. V této třídě jsou definovány všechny vlastnosti, jak má světlý režim vypadat. Jako jsou jednotlivé barvy, jednotlivé fonty i jak mají vypadat jednotlivé widgety. Druhý parametr je darkTheme, který dělá to samé, jenom pro tmavý režim. Poslední je themeMode, kde už aplikaci říkáme jaký režim má současné chvíli uživatel zvolen, právě pro tento parametr je využit provider ThemeProvider. Výběr režimu je vidět na obrázku 7.8.



Obr. 7.8: Výběr světlého, tmavého režimu aplikace v obou variantách.

8 Testování

Poslední část diplomové práce se věnuje testování, jak mobilní aplikace na obou platformách tak i komunikace se serverem. Testování mobilní aplikace se nachází v sekcí 8.1 a testování komunikace se serverem pak v sekci 8.2.

8.1 Testování mobilní aplikace

Testování mobilní aplikace probíhalo průběžně během celého vývoje. Jelikož se jedná o multiplatformní aplikaci, bylo nutné ověřovat všechny funkcionality na obou platformách. Během vývoje byly využity jak emulátory jednotlivých zařízení, tak i fyzické zařízení. Finální aplikace běží na nejnovějších dostupných verzích jednotlivých nástrojů (Flutter: 2.10.4, Dart: 2.16.2). Aplikace byla otestována na těchto fyzických zařízeních:

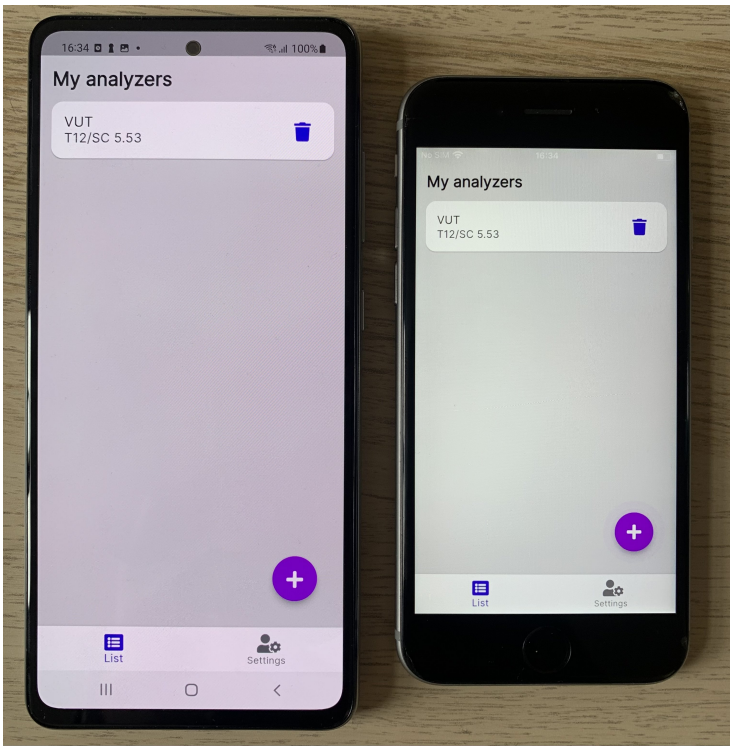
- Android:
  - Samsung Galaxy A52s 5G, OS: Android 12.
  - Samsung Galaxy J5, OS: Android 9.
  - Xiaomi Mi A1, OS: Android 9.
- iOS:
  - iPhone XR, OS: iOS 15.4.1.
  - iPhone 6, OS: iOS 15.4.1.

Na obrázku 8.1 je vidět aplikace běžící na androidu tak i na iOS zařízení. Pro jednoduché stažení testovací verze aplikace, jde využít přiložené QR kódy na obrázku 8.2 pro Android přes službu Firebase. Na obrázku 8.3 pro iOS přes službu TestFlight.

8.2 Testování komunikace

Jelikož v době vypracovaní této diplomové práce nebyl k dispozici hotový analyzer, proto se data musela simulovat. Náhodně jsou generovaná data pro list portů, kde se jejich aktuální status náhodně mění. Data svázaná s OTDR měřením jsou reálná, ale ne aktuální, jedná se o naměřená data z minulosti. Testování komunikace mezi aplikací a serverem probíhalo ve dvou krocích. První testování komunikace bylo započato už během vývoje aplikace a to pomocí virtuálního serveru. Tento server byl naprogramován pomocí nástroje node.js a komunikace probíhala přes localhost připojení. To znamená, že se k serveru mohl připojovat pouze emulátor fyzického zařízení.

Následně byl tento virtuální server převeden na fyzický server s veřejnou IP adresou. Pro tento účel byla využita webová služba Glitch.com. Tato služba nám umožňuje hostovat statické webové stránky nebo backendové aplikace běžící pomocí



Obr. 8.1: Aplikace běžící fyzických zařízení nalevo: Android, napravo: iOS.

Obr. 8.2: QR kód pro stažení Android verze aplikace.

node.js. Právě druhé možnosti lze využít pro hostování našeho serveru s daty, jelikož běží právě na zmiňovaném node.js. Glitch nám zpřístupní náš server kdekoliv na světě. Uživatel může tedy nepřetržitě kontrolovat stav jednotlivých analyzátorů.

Aplikace v obou případech jak přes localhost tak i při komunikaci se serverem hostovaným na Glitch.com fungovala naprosto bezchybně. U Glitch serveru se projevuje prvotní zdlouhavé načtení dat. Toto zpoždění je dáno tím, že používáme verzi Glitch free. V této verzi je uvedeno, že aplikace (server) po 5 minutové nečinnosti

Obr. 8.3: QR kód pro stažení iOS verze aplikace.

spadne do režimu spánku. Následné probuzení a znovu spuštění aplikace zabere čas. Načtení prvních dat po probuzení serveru trvá zhruba 8 sekund. Když je dotaz směřován na aktivní server, odpověď se pohybuje okolo 1 vteřiny. Adresa serveru pro testovací účely je analyzer-server.glitch.me.

Aplikace skvěle reaguje i na chybové situace, které by mohly nastat. Jako například špatný klíč, adresa nebo port serveru. Všechny tyto stavy jsou ošetřeny a aplikace se snaží dát uživateli co možná nejpodrobnější informace o nastalé situaci.

Závěr

V první kapitole teoretické části byly podrobně popsány optická vlákna a jejich základní princip. Dále jejich rozdělení, nejdůležitější vlastnosti a parametry. Rovněž jsou zde popsány nejpoužívanější metody měření útlumu a to především metoda OTDR. Druhá kapitola se věnovala zabezpečení optických vláken především na fyzické vrstvě. Třetí kapitola se věnovala komunikačním protokolům pro přenos dat mezi klientem a serverem. Byl zde podrobně popsán HTTP protokol a modernější protokol WebSocket. Dále jsou v textu popsaný nástroje Flutter a Firebase.

Hlavním cílem diplomové práce bylo navrhnout mobilní aplikaci, která komunikuje s analyzátorem optických vláken skrze webový server. Na aplikaci byl kladen nárok tak, aby byla spustitelná na obou hlavních mobilních platformách, tedy Android a iOS. Proto bylo rozhodnuto využít multiplatformní framework Flutter. Při vývoji aplikace byl kladen důraz na co možná nejčistější a čitelný kód. Pro backend aplikace byla využitá služba Firebase. Jeden z cílů dalších bylo navrhnout komunikaci mezi mobilní aplikaci a serverem. Aplikace využívá pro komunikaci se serverem i s backendem moderní webSocket protokol. Ta uživateli umožňuje bezpečně spravovat libovolné množství analyzátorů. Rovněž uživateli dovoluje vytvořit si vlastní účet a také si aplikaci různě přizpůsobit. Od výběru tmavého/světlého režimu až po výběr jazyka (angličtina, čeština).

Jelikož v době psaní diplomové práce nebyl k dispozici funkční prototyp analyzátoru, byly data simulována. Pro simulaci byly na webSocket serveru uloženy různé náměry OTDR. Tyto náměry jsou na požádaní odesilány do aplikace. Aktuální stav portu je náhodně generován tak, aby se simulovaly reálné podmínky provozu. Server je aktivně nasazen na platformě Glitch.com.

V rámci testování byla mobilní aplikace odzkoušena na obou platformách a na různých fyzických zařízeních s odlišnými verzemi operačního systému. Během testování nebyl zaznamenán žádný pád aplikace nebo nepředvidatelné chování. Také byla otestována komunikace se serverem. Zde se objevil nedostatek co se týče prvotní odezvy serveru. Tento problém je ale způsoben bezplatnou verzi hostingu serveru. V aplikaci byly ošetřeny různé chyby, které mohou uživatelé způsobit, jako je špatná adresa nebo špatný klíč k serveru.

Literatura

[1] STROBEL, Otto. *Optical and microwave technologies for telecommunication networks.* New Delhi, India: Wiley, 2016. ISBN 9781119971900.

[2] *Rodina protokolů TCP/IP.* EArchive [online]. [cit. 2021-12-13]. Dostupné z: <https://www.earchiv.cz/anovinky/ai1592.php3>.

[3] PROF. ING. FILKA, Miloslav, CSc. *Optické sítě - přednášky [online].* Brno, 2007 [cit. 2021-10-27]. Dostupné z: <https://optolab.utko.feec.vutbr.cz/wp-content/uploads/Opticke_site_prednasky_P.pdf>. VUT.

[4] *Vysvětlení principu šíření světla optickými vlákny.* Optoelektrotechnika [online]. Brno, 2015 [cit. 2021-10-28]. Dostupné z: <https://publi.cz/books/185/04.html>.

[5] PROF. ING. FILKA, Miloslav, CSc. *Přenosová media.* [online] Brno, 2011 [cit. 2021-10-23]. Dostupné z: <https://optolab.utko.feec.vutbr.cz/wp-content/uploads/BPRM.pdf>. VUT.

[6] *Druhy optických vláken.* Optoelektrotechnika [online]. Brno, 2015 [cit. 2021-10-26]. Dostupné z: <https://publi.cz/books/185/05.html>

[7] *Uspořádání přenosové cesty s optickým vláknem.* Optoelektrotechnika [online]. Brno, 2015 [cit. 2021-10-26]. Dostupné z: <https://publi.cz/books/185/03.html>

[8] BC. BALADA, Radek. *Klasifikace typu digitální modulace* [online]. Brno, 2010 [cit. 2021-10-26]. Dostupné z: <https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=27173>. Diplomová práce. VUT. Vedoucí práce Ing. Karel Povalač.

[9] *Difference between Optical Repeater vs Optical Amplifier.* RF Wireless World [online]. 2012 [cit. 2021-10-27]. Dostupné z: <https://www.rfwireless-world.com/Terminology/Optical-Repeater-vs-Optical-Amplifier.html>

[10] ŠIMKOVÁ, Barbora. *Laboratorní úloha - analyzátor optických tras OTDR* [online]. Brno, 2021 [cit. 2021-10-27]. Dostupné z: <https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=227687>. Bakalářská práce. VUT. Vedoucí práce Ing. Petr Dejdar.

[11] *Parametry optických vláken.* Optoelektrotechnika [online]. Brno, 2015 [cit. 2021-10-27]. Dostupné z: <https://publi.cz/books/185/06.html>

[12] PROF. ING. FILKA, Miloslav, CSc. *Optoelektronika pro telekomunikace a informatiku.* Brno: Centa, 2009. 369 s. ISBN 978-80-86785-14-1.

[13] REICHL, Jaroslav a Martin VŠETIČKA. *Rozptyl světla v optickém vláknu.* Encyklopedie fyziky [online]. 2006 [cit. 2021-10-28]. Dostupné z: <http://fyzika.jreichl.com/main.article/view/1671-rozptyl-svetla-v-optickem-vlaknu>

[14] *Understanding OTDRs.* [online] Anritsu, 2011 [cit. 2021-11-17]. Dostupné z: <https://rossfibersolutions.com/fiber-optic-pdfs/Anritsu-understanding-otdrs.pdf>

[15] *Technologie xWDM, (DWDM,CWDM).* Optoelektrotechnika [online]. Brno, 2015 [cit. 2021-11-01]. Dostupné z: <https://publi.cz/books/185/07.html>

[16] *Spectral grids for WDM applications: DWDM frequency grid.* ITU-T [online]. ITU-T, 2020 [cit. 2021-11-01]. Dostupné z: <https://www.itu.int/rec/T-REC-G.694.1-202010-I/en>

[17] *Systémy s vlnovým multiplexem – rastr CWDM a DWDM, požadavky na lasery pro DWDM.* Nové trendy v elektronických komunikacích Optické systémy a sítě [online]. Praha [cit. 2021-11-01]. Dostupné z: <https://publi.cz/books/241/07.html>

[18] *Spectral grids for WDM applications: CWDM wavelength grid.* ITU-T [online]. ITU-T, 2003 [cit. 2021-11-01]. Dostupné z: <https://www.itu.int/rec/T-REC-G.694.2-200312-I>

[19] *Měření optických vláken a optických kabelových tras.* Optoelektrotechnika [online]. Brno, 2015 [cit. 2021-11-23]. Dostupné z: <https://publi.cz/books/185/14.html>

[20] MGR. VONDRUŠKA, Pavel. *Crypto-World. Crypto-World* [online]. E-zin Crypto-World, 2007, 15.11.2007 [cit. 2021-11-24]. Dostupné z: <http://crypto-world.info/casop9/crypto11_07.pdf>.

[21] BURDA, Karel doc. Ing. CSc. *Ochrany komunikačních kanálů* [online]. Brno, 2021 [cit. 2021-11-24]. Dostupné z: <https://moodle.vut.cz/pluginfile.php/249157/mod_resource/content/1/MBKS%2007a.pdf>. VUT.

[22] *Management bezpečnosti fyzické vrstvy.* FAnzdoc [online]. 2013 [cit. 2021-12-13]. Dostupné z: <https://adoc.pub/management-bezpenosti-fyzicke-vrstvy.html>

[23] *Keyed LC System Secures Your Network.* Fiber Cabling Solution [online]. 2016 [cit. 2021-11-02]. Dostupné z: <https://www.fiber-optic-cable-sale.com/keyed-lc-system-secures-your-network.html>

[24] BC. ROZSYPAL, Ondřej. *Fyzická bezpečnost a management síťě na fyzické vrstvě* [online]. Brno, 2018 [cit. 2021-11-24]. Dostupné z: <https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=172878>. Diplomová práce. VUT. Vedoucí práce Doc. Ing. Jan Hajný, Ph.D.

[25] BURDA, Karel doc. Ing. CSc. *Návrh, správa a bezpečnost počítačových sítí: Fyzická vrstva v počítačových sítích.* Brno, 2020. VUT.

[26] *Fiber Monitoring: What is Fiber Monitoring?* Viavi [online]. [cit. 2021-11-30]. Dostupné z: <https://bit.ly/3rr4Omm>.

[27] *OTU-8000 Optical Test Unit.* Viavi [online]. [cit. 2021-11-30]. Dostupné z: <https://www.viavisolutions.com/en-us/products/otu-8000-optical-test-unit>.

[28] *What is HTTP Long Polling?* PubNub [online]. [cit. 2021-11-26]. Dostupné z: <https://www.pubnub.com/blog/http-long-polling/>.

[29] *What is HTTP Streaming and How Does it Work?* PubNub [online]. [cit. 2021-11-26]. Dostupné z: <https://bit.ly/2ZCHLJQ>.

[30] *IoT Hub: What Use Case for WebSockets?* Scaleway [online]. 2021 [cit. 2021-11-26]. Dostupné z: <https://blog.scaleway.com/iot-hub-what-use-case-for-websockets/>.

[31] PTERNEAS, Vangos. *Getting Started with HTML5 WebSocket Programming* [online]. UK: Packt Publishing, 2013 [cit. 2021-11-26]. ISBN 978-1-78216-696-2. Dostupné z: <https://bit.ly/3I6RkSM>.

[32] *Introducing JSON.* Json [online]. [cit. 2021-11-30]. Dostupné z: <https://www.json.org/json-en.html>.

[33] *XML - Overview.* Tutorial point [online]. [cit. 2021-11-30]. Dostupné z: <https://www.tutorialspoint.com/xml/xml_overview.htm>.

[34] *Flutter architectural overview.* Flutter [online]. [cit. 2022-04-10]. Dostupné z: <https://docs.flutter.dev/resources/architectural-overview>.

[35] *Dart overview.* Dart [online]. [cit. 2022-04-10]. Dostupné z: <https://dart.dev/overview>

[36] BC. NYMSA, Petr. *Coffee Time Mobile Application in Flutter* [online]. Praha, 2020 [cit. 2022-04-10]. Dostupné z: <https://dspace.cvut.cz/bitstream/handle/10467/87986/F8-DP-2020-Nymsa-Petr-thesis.pdf?sequence=-1&isAllowed=y.Diplomovápráce.ČVUT.>

[37] BC. POKORNÝ, Martin. *Multiplatformní mobilní aplikace pomocí technologie Flutter* [online]. Zlín, 2019 [cit. 2022-04-10]. Dostupné z: <https://digilib.k.utb.cz/bitstream/handle/10563/44468/pokorný_2019_dp.pdf?sequence=1&isAllowed=y.Diplomováprice.UTB.>.

[38] *Databáze NoSQL – co je NoSQL?* Azure [online]. [cit. 2022-04-10]. Dostupné z: <https://azure.microsoft.com/cs-cz/overview/nosql-database/>.

[39] *Cloud Firestore.* Firebase [online]. [cit. 2022-04-10]. Dostupné z: <https://firebase.google.com/docs/firestore>.

Seznam symbolů a zkratek

ADZ Attenuation Dead Zone – Útlumová mrtvá zóna

AIM Automated Infrastructure Management – Automatizovaný systém správy infrastruktury

AM Amplitude Modulation – Amplitudová modulace

AOT Ahead Of Time

API Application Programming Interface

ASK Amplitude Shift Keying – Amplitudové klíčován

BLOC Business Logic Components

CWDM Coarse Wavelength Division Multiplexing – Hrubý vlnový multiplex

DWDM Dense Wavelength Division Multiplexing – Hustý vlnový multiplex

EDZ Event Dead Tone – Událostní mrtvá zóna

FIFO First In First Out – První dovnitř, první ven

FM Frequency Modulation – Frekvenční modulace

FSK Frequency Shift Keying – Frekvenční klíčování

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IP Internet Protocol

IPC Inter Process Communication – Komunikace mezi procesy

JIT Just In Time

JSON JavaScript Object Notation – JavaScriptový objektový zápis

LAN Local Area Network – Lokální síť

MMF GI Multi Mode Fiber Graded Index – Mnohovidové vlákno s gradientní změnou indexu lomu

MMF SI	Multi Mode Fiber Step Index – Mnohovidové vlákno se skokovou změnou indexu lomu
NA	Numerical Apparatus – Numerická aparatura
ONMSi	Optical Network Management System
OTDR	Optical Time Domain Reflectometer – Optická reflektometrie v časové oblasti
PM	Phase Modulation – Fázová modulace
PSK	Phase Shift Keying – Fázové klíčování
QAM	Quadrature Amplitude Modulation – Kvadraturní amplitudová modulace
QR	Quick Response
REST	Representational State Transfer
RFID	Radio Frequency Identification – Identifikace na rádiové frekvenci
SMF	Single Mode Fiber – Jednovidové vlákno
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP	ransmission Control Protocol
UI	User Interface - Uživatelské rozhraní
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WDM	Wavelength Division Multiplex – Vlnový multiplex
WWDM	Wide Wavelength Division Multiplexing – Široký vlnový multiplex
XML	Extensible Markup Language
YAML	Ain't Markup Language
n	Index lomu [-]

NA	Numerická apertura [-]
V	Normalizovaná frekvence [-]
α	Útlum $[dB]$