ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra počítačů

Diplomová práce

Systém pro automatické nahrávání přednášek

Bc. Jiří Fryč
Otevřená informatika
Softwarové inženýrství

Květen 2019
Vedoucí práce: Ing. Jan Kubr, PhD.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Fryč** Jméno: **Jiří** Osobní číslo: **425077**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Systém pro automatické nahrávání přednášek

Název diplomové práce anglicky:

Automatic Tutorial Recording System

Pokyny pro vypracování:

Analyzujte požadavky kladené na systém pro automatické nahrávání přednášek na ČVUT FEL. Analyzujte technické vybavení dostupné a plánované pro tento účel. Navrhněte a implementujte systém umožňující automatické nahrávání přednášek. Systém umožní zaznamenání přednášky podle informací z rozvrhu a převod videa do vhodných formátů pro sledování a archivaci. Navrhněte vhodné postupy pro testování jednotlivých komponent systému i celého systému. Testy proveďte.

Seznam doporučené literatury:

KOSapi, https://kosapi.fit.cvut.cz/projects/kosapi/wiki
NDI Technical Brief, https://www.newtek.com/ndi/sdk/
D. Grois, T. Nguyen and D. Marpe; Coding efficiency comparison of AV1/VP9, H.265/MPEG-HEVC, and H.264/MPEG-AVC encoders; 2016 Picture Coding Symposium (PCS); Nuremberg, 2016, pp. 1-5.;doi: 10.1109/PCS.2016.7906321

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jan Kubr, Ph.D., katedra počítačové grafiky a interakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **04.02.2019** Termín odevzdání diplomové práce: **24.05.2019**

Platnost zadání diplomové práce: **20.09.2020**

Ing. Jan Kubr, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. Datum převzetí zadání

Podpis studenta

Poděkování / Prohlášení

Chtěl bych poděkovat členům střediska výpočetní techniky a informatiky naší fakulty, jmenovitě Ivo Hulínský a Ing. Martin Samek za jejich spolupráci na řešení, věcné připomínky a umožnění vzniku této diplomové práce. S jejich pomocí jsem mohl provést analýzu fakultních zařízení pro nahrávání. Taktéž bych rád poděkoval vedoucímu této práce Ing. Jan Kubr, PhD. za čas, ochotu a zkušenosti které této práci věnoval.

A v neposlední řadě mojí přítelkyni za trpělivost a shovívavost během studia a během tvorby této práce.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 23. 5. 2019

.....

Abstrakt / Abstract

Tato diplomová práce se zabývá analýzou, návrhem a implementací řešení pro automatické nahrávání přednášek pro Fakultu elektrotechnickou Českého vysokého učení technického (dále jen ČVUT).

Řešením je distribuovaný systém uzlů a pokrývá komunikaci uzlů, záznam z přednáškové místnosti, uložení záznamů a zpracování záznamu do výstupní kvality. Navržené řešení je modulární a rozšiřitelné pomocí rozhraní knihoven, zároveň velká část implementovaného řešení je implementována právě jako knihovny a může sloužit jako ukázka pro případné rozšíření.

Klíčová slova: distribuované výpočty, nahrávání, eLearning, nahrávání přednášek, zpracování videa

This thesis deals with the analysis, design and implementation of the solutions for automatic recording of lectures, primary for the Faculty of Electrical Engineering of the Czech Technical University (CTU).

The solution is a distributed system of nodes and covers: nodes communication, recording from a lecture room, storing records and processing a record into output quality. The proposed solution is modular and extensible by libraries. Large part of the implemented solution is already implemented as a libraries and can serve as an example for future extensions.

Keywords: distributed computing, video recording, eLearning, lecture streaming, lecture capture, video editing

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod do problematiky

V této práci se budeme zabývat problematikou nahrávání audio a video záznamů z přednáškových sálů, primárně z těch na naší fakultě elektrotechnické a implementací systému „Kazeta“.

Naše motivace pro tento systém je doposud chybějící řešení pro nahrávání přednášek na fakultě. V řešení se pokusíme pokrýt prvotní část pro vytvoření tohoto systému a to jest proces zachytávání záznamu, uložení záznamu a jeho zpracování do výstupních formátů.

Systém kazeta je decentralizovaný systém tvořený uzly s rozdílnými moduly pro zpracování videa. *Capture modul* (pro záznam z místnosti), *Process modul* (pro zpracování záznamu), *Persist modul* (pro uložení záznamu), *Deliver modul* (pro distribuci záznamu koncovým uživatelům) a *Network modul* (pro komunikaci uzlů).

V rámci práce byly implementovány moduly *capture*, *process*, *persist* a *network*.

1.1 Struktura práce

Práce je rozdělena do následujících kapitol:

- **Kapitola 1 - Úvod do problematiky**: V této kapitole bude čtenář seznámen s problematikou nahrávání z přednáškových sálů a motivací která vedle k vytvoření této práce. Současně v této kapitole rozebere i problematiku, která prací není řešena.
- **Kapitola 2 - Analýza**: Tato kapitola je zaměřená na analýzu a srovnání současných standardů, dostupného vybavení fakulty a požadavků fakulty na nahrávací software.
- **Kapitola 3 - Architektura řešení**: V této kapitole bude čtenář seznámen s naším návrhem řešení problematiky. Jsou zde detailně popsány potřebná zapojení místností, komunikace, možnosti řešení a v neposlední řadě se zde zabýváme i škálováním a stabilitě celého navrženého řešení.
- **Kapitola 4 - Implementace**: Tato kapitola se zabývá samotnou implementací navrženého řešení. A problémy, které vznikli během dané implementace.
- **Kapitola 5 - Konfigurace**: V této kapitole seznamujeme čtenáře s konfiguračními soubory, instalací a nastavením systému.
- **Kapitola 6 - CLI**: Kapitola představuje čtenáři konzolové rozhraní přes které lze ovládat celý řešení „Kazeta“. Konzole je v současnosti jediné uživatelské rozhraní systémů.
- **Kapitola 7 - Testování**: Tato kapitola obsahuje informace o testování práce. Zároveň čtenáře seznámí s experimenty, které byly provedeny.
- **Kapitola 8 - Závěr**: V závěru čtenáře seznamujeme s budoucností práce a s dosaženými cíli.

1.2 Motivace

V současnosti neexistuje žádné dostupné řešení, které by plně pokrývalo specifické potřeby vysokých škol, či alespoň částečně umožňovalo distribuované ovládání nahrávání z jednotlivých sálů a zpracování do námi požadovaných výstupních formátů.

Díky tomu jsme se rozhodli pro implementaci vlastního řešení, které pokryje naše specifické požadavky, bude přizpůsobitelné a napojitelné na fakultní systémy. V cílovém stavu bychom také měli rádi výstupy dostupné přímo ve fakultním LMS.

1.3 Problematika

V této sekci rozebere největší problémy spojené s audio-video záznamy, které v této práci musí buďto řešit, či alespoň zohlednit při návrhu řešení.

1.3.1 Výpočetní a přenosové nároky

Přenos a záznam audio-video záznamů patří i v současné době k nejvíce náročným operacím na poli informačních technologií. Nekomprimovaný záznam přednášky pomocí kamery s vysokým rozlišením (FullHD) vyvíjí stejné zatížení na přenos po síti jako 450 uživatelů přistupující každou sekundu na fakultní web[1]. Toto odpovídá 2 430 000 uživatelům při standardní délce přednášky (90 minut).

1.3.2 Rozdílnost technologií

Problematika technologií v audio-video záznamech je značně rozsáhlá, máme zde rozdílné technologie pro ovládání kamer/světel/matic, hardware technologie pro přenos videa, formát pro přenos videa, formát pro kódování videa, formát pro ukládání videa, formát pro editaci videa a to jsme se zatím bavili pouze o video části záznamu.

Díky rozsáhlosti se primárně zaměříme jen na technologie, které jsou již přítomné na fakultě, či na takové, které se v blízké době plánují pořídit. A z informačního hlediska zmíníme i ty, které se na fakultě využívaly v minulosti.

1.3.3 Rozdílnost místností

Rozdílnost místností, ve kterých se záznamy nahrávají, je pro nahrávání často problematičtější než rozdílnost technologií. Musíme se potýkat s různým natočením místností a tím rozdílně dopadajícím světlem. Dále různě rozestavěnými plátny, tabulemi a pulty pro přednášející. Problémy můžeme samozřejmě jmenovat dále.

Tyto problémy spojuje jeden faktor a to ten, že je nemůžeme ovlivnit, tudíž vyžadují aby s nimi automatické nahrávání počítalo a umělo se přizpůsobovat podle místnosti.

Vyjímku tvoří nové budovy, kde se často staví identické přednáškové sály, příkladem může být nová budova ČVUT na adrese Jugoslávských partyzánů 1580/3.

1.3.4 Lidský faktor

V neposlední řadě hraje roli lidský faktor, nejčastěji přednášející. Každý přednášející se o přednášce chová jinak. Někteří přednášející jsou statičtí a stojí u pultíku či tabule, jiní cestují po pódiu a pak jsou i někteří, kteří chodí mezi studenty. V některých případech zde máme i přednášky s více přednášejícími, či se studenty u tabule.

Taktéž někteří přednášející využívají různé pomůcky. Bohužel ne všechny pomůcky dobře korelují s nahráváním přednášek. Příkladem mohou býti ukazovátka. Teleskopická ukazovátka jsou dobře viditelná i na video-záznamu, nicméně laserová nejsou na video-záznamu viditelná pokud nemáme k dispozici vysoce kvalitní nekomprimovaný záznam.

V práci na některé tyto věci narazíme, například chodící přednášející značně ovlivňuje velikost záznamu oproti statickému, nicméně se tuto problematiku nesnažíme řešit. Tvorba záznamu by neměla omezovat přednášejícího a jeho preferovaný styl výuky.

[1] `https://fel.cvut.cz/`

1.4 Co v této práci není řešeno

Tato práce se zaměřuje pouze na technickou stránku problému a neřeší požadavky ze strany legislativy, či požadavků konečných uživatelů záznamu.

1.4.1 GDPR

Práce neřeší požadavky spojené s GDPR při nahrávání audio a video záznamů a to hned z několika důvodů. Primárně nemáme dostatečné právnické zkušenosti a znalosti abychom mohli navrhnout řešení pro tvorbu a zpracování záznamů a paralelně toto řešení problematiky GDPR by muselo býti schváleno vedením fakulty.

Nicméně bych rád v této práci odkázal na možné řešení[1], které implementovali jiné vysoké školy a instituce. Řešení se skládá z vyhrazení takzvané „hluché zóny“ v přednáškovém sále.

Hluchá zóna je viditelně oddělená od zbytku místnosti a není v ní pořizován zvukový a ani obrazový záznam. Toto má nicméně nevýhody a to, že zóna může být buď příliš velká či příliš malá pro účastníky, kteří nechtějí býti nahráváni.

Druhá část řešení je automatické informování účastníků před startem nahrávání. Toto informování obsahuje informace o tom, jak bude záznam zpracováván a k jakým účelům bude záznam použit. Záznam se z právního hlediska nesmí využít k jinému účelu než který byl takto oznámen účastníkům.

1.4.2 Výstupní streamování koncovým uživatelům

Práce se zaměřuje pouze na záznam, zpracování záznamu a uložení záznamu ve formátu vhodným ke streamování/stažení koncovým uživatelem. Tudíž neřeší webové rozhraní a protokol pro koncové uživatele. Práce je nicméně použitelná pro většinu soudobých protokolů jako *RTMP*, *HLS*, či *WebRTC* a počítáme, že tato funkcionalita bude implementována v rámci další diplomové práce.

1.4.3 Sestřihy záznamů

Práce v současné verzi neřeší ruční sestřihy záznamů a záznamy nabízí takzvaně „as-is“ (tak jak jsou nahrané), práce počítá s co nejvíc automatickým procesem a ruční sestřihy by tento proces značně narušili.

1.4.4 Otáčení kamery za přednášejícím

Práce neřeší otáčení kamery během záznamu, pouze otočení před a po záznamu. Sledování přednášejícího je složitá funkcionalita vyžadující pokročilou analýzu obrazu. Nicméně si myslíme, že by mohla být řešena jako samostatná navazující práce.

[1] `https://www.jisc.ac.uk/guides/recording-lectures-legal-considerations`

Kapitola 2
Analýza

V této kapitole se zaměříme na analýzu fakultě dostupného vybavení, standardů a protokolů. Rozebereme jejich jednotlivé výhody a nevýhody a pokusíme se najít ideální řešení, které navrhneme v následující kapitole.

2.1 Dostupné vybavení fakulty

Fakulta disponuje značně rozmanitým vybavením na poli audio/video techniky, lze říci, že většina sálů byla do nedávné doby unikátní zapojením, vybavením a i kvalitou výstupu a až během posledních let dochází ke standardizaci vybavení a zapojení.

2.1.1 Minulost

K minulosti výukových prostor a jejich stavu záznamových zařízení jsme bohužel nenašli žádnou pevnou dokumentaci a opíráme se tudíž pouze o informace které jsme zjistili při konzultacích na SVTI s panem Ing. Martinem Samkem, panem Ivo Hulínským a paní Marcelou Charvátovou.

Stav byl založený na technologii S-VIDEO pro přenos do ovládací místnosti, kde docházelo k záznamu na externí disky či kazety, tyto byly následně manuálně odneseny k archivaci, či k editaci. Tento systém nebyl nicméně příliš často využíván kvůli velice špatné kvalitě způsobené technologií S-VIDEO.

2.1.2 Současný stav

Většina nahrávání je v současnosti stále řízena manuálně, částečně z nahrávacích místností vedle přednáškových sálů a následně i přímo v přednáškovém sále pomocí kamery na stativu. Pouze minimum nahrávání se děje automaticky bez přítomnosti obsluhy a i pro toto nahrávání je potřeba manuální spuštění a ukončení.

V dnešní době probíhá snaha standardizovat všechny sály a vytvoření decentralizovaného systému pro záznam, ukládání a přehrávání záznamů.

2.1.3 Blízká budoucnost

Fakulta plánuje v nejbližší době rozšiřování audio/video techniky v fakultních přednáškových sálech a to jak v těch umístěných v Dejvicích, tak i na Karlově náměstí. Jakožto standard pro kvalitu nahrávání budou voleny rozlišení FullHD, či 4k a pro přenos byla zvolena technologie NDI. V příloze C můžete nalézt bližší informace k plánu modernizace výukových prostor.

Pro instalaci bylo nově nakoupeno pět 4k kamer typu „Datavideo BC-200T“, které budou rozmístěny po jednom kusu v následujících místnostech:

- Velká posluchárna *T2:D3-209*
- Velká posluchárna *T2:D3-309*
- Velká posluchárna *KN:E-107*
- Velká posluchárna *KN:E-301*

- Zasedací místnost *T2:A4-7*

Zároveň došlo k dokoupení dvanácti FullHD kamer typu „NewTek NDI HX PTZ1“, které budou rozmístěny následovně:

- Velká posluchárna *KN:E-107* (Tři kusy)
- Velká posluchárna *T2:D3-209* (Dva kusy)
- Velká posluchárna *T2:D3-309* (Jeden kus)
- Velká posluchárna *KN:E-301* (Jeden kus)
- Malá posluchárna *T2:C3-132* (Jeden kus)
- Malá posluchárna *T2:C3-135* (Jeden kus)
- Malá posluchárna *T2:C3-337* (Jeden kus)
- Malá posluchárna *T2:C3-34* (Jeden kus)
- Zasedací místnost *T2:A4-7* (Jeden kus)

Díky těmto nákupům je po hardwarové stránce, fakulta připravena na velmi kvalitní záznamy z poslucháren.

2.2 Požadavky na automatický záznam

V této sekci se zaměříme na požadavky, které jsou ze strany školy na systém pro automatickou tvorbu záznamů přednášek.

2.2.1 Nastavení sálu pro nahrávání

Sály vyžadují před samotným nahráváním nastavení. V první řadě se toto týká zresetování kamer do jejich správného natočení a přiblížení.

Některé kamery neumožňují nastavení přesného natočení/přiblížení, nicméně v těchto případech můžeme využít zarážek kamery. Zarážkami myslíme maximální/minimální přiblížení a vertikální/horizontální natočení.

Zarážky využíváme tím způsobem, že kameru co nejvíce oddálíme a nastavíme rohovou pozici natočení, následně odkrokujeme do požadované pozice. Tímto způsobem dosáhneme relativně přesného nastavení.

Následně je třeba provést úkony spojené s nastavením mikrofonů, světel, oken, maticových přepínačů a dalších zařízení. Pro toto musíme využívat specifické protokoly/knihovny daných zařízení. V případě naší fakulty se nejvíce jedná o protokol artnet[1] pro světla a knihovny společnosti blackmagic pro maticové přepínače.

Stejným způsobem je třeba i provést akce po ukončení nahrávání a to i v případě že, nahrávání selhalo či bylo předčasně ukončeno.

2.2.2 Záznam ze sálu

Řešení musí umožnit nahrávat záznam z přednáškových sálů v dostatečné, nejlépe vstupní kvalitě a tento záznam kódovat kodekem nenáročným na místo a přenos.

2.2.3 Zpracování záznamu

Zpracování záznamu do výstupního vhodného pro přehrávání ve webových prohlížečích s dostatečnou kvalitou a co nejlepším poměrem kvality a velikosti záznamu.

2.2.4 Uložení záznamu

Uložení záznamu pro umožnění archivace a pro přehrávání koncovými uživateli.

[1] `https://www.lightjams.com/artnet.html`

2.2.5 Plánování nahrávání přednášek a cvičení předmětů

Plánování nahrávání předmětů dle školního systému KOS. Řešení musí umět se dotazovat školních systémů na místnosti a časy, ve kterých se fakultní předměty vyučují. A řešit i případné změny v rozvrhu jako svátky a náhradní výuka.

2.2.6 Ovládání

Řešení musí obsahovat uživatelské rozhraní pro přidávání/odebírání a nastavování celého řešení a záznamů. Ovládání má v současnosti sloužit pouze jako „proof-of-concept“, neboli ukázat schopnost tyto úkony vykonávat.

2.3 Kodeky videa

Kódování videa dělíme z hlediska použití na dva druhy: **kodeky pro editaci** a **kodeky pro distribuci**. V první řadě musíme omezit výběr na několik kodeků, které si blíže probereme.

Kodeky pro editaci jsme vybrali pomocí článku „Video format conversion“[1] zveřejněného v magazínu „Broadcast Engineering (World Edition)“. Dle něho jsme vybrali pro srovnání formáty *ProRes*,*H.264*,*bez kodeku (8bit)* a *bez kodeku (10bit)*.

Kodeky pro distribuci jsme vybrali pomocí stránky `caniuse`[1] dne 1.3.2019. Tato stránka obsahuje aktuální informace o podpoře různých formátů a standardů v prohlížečích koncových uživatelů. Zajímala nás podpora v prohlížečích Edge, Firefox, Chrome, Safari, Opera a v mobilních prohlížečích Android browser, Chrome a iOS Safari. Na základě těchto požadavků jsme pomocí stránky vybrali kodeky *H.264*, *H.265*, *VP8*, *VP9*, *AV1*.

2.3.1 H.264

H.264 kodek[2] je dlouhodobě nejrozšířenějším kodekem na trhu díky jeho jednoduchosti, universálnosti a relativně dobré kompresi obrazu. Poprvé se tento formát začal používat v květnu roku 2003 a od té doby našel uplatnění na blue-ray, pozemním digitálním vysílání, internetovém streamování a i satelitním vysílání. Díky své rozšířenosti obsahují moderní procesory i nové sety instrukcí, které zefektivňují procesorový encoding a decoding do tohoto kodeku.

2.3.2 H.265

H.265 kodek[3] je nástupce kodeku H.264. Od H.264 se kodek odlišuje lepšími kompresními vlastnostmi od 25% až po 70% oproti H.264 se stejným bitrate[4]. Toto je vykoupeno vyššími nároky na výkon při kódování i přehrávání videa.

Video kodek	Průměrná bitrate redukce H.264			
	480p	720p	1080p	2160p
H.265	52%	56%	62%	64%

Tabulka 2.1. Srovnání H.264 a H.265 při různých rozlišení

[1] `https://caniuse.com/#search=video%20format`

2.3.3 VP8

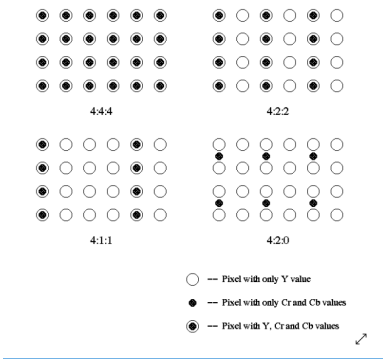
Kodek VP8[5] je následník kodeku VP7 vyvinutý společností On2 Technologies pod uzavřenou licencí. Po odkoupení společnosti společností Google byla licence změněna na otevřenou, následně byl kodek implementován do většiny přehrávačů a webových prohlížečů. Dle Google je kodek v současné době primárně využíván jako nástupce formátu GIF pro krátké animace na webových stránkách[6].

2.3.4 VP9

Kodek VP9[7] je následník kodeku VP8. Původně vznikl pro potřeby platformy Youtube[1], postupně se ale rozšířil na všechny platformy a stal se standardem pro přenos 4k a 8k záznamů. Bylo počítáno, že tento kodek plně nahradí standard VP8, nicméně společnost Apple a její webový prohlížeč Safari, jakožto poslední, ještě tento standard nevyužívají. Více se lze dočíst v článku[8] na webu streamingmedia[2], kde autor řeší nevůli Applu implementovat VP kodeky z důvodů licenčních výhod, které společnosti Apple vychází z využívání H.264 a H.265 kodeků.

2.3.5 Prores

Prores[9] je vysoko kvalitní video kodek vytvořený společností Apple pro post-produkci záznamů. Prores dělíme na několik podtypů, hlavní rozdělení je na formáty 422 a 4444. Tyto se od sebe liší takzvaným „chroma subsampling“, blíže popsaným v literatuře[10] či na obrázku 2.1. Následně dělíme podle kvality výstupu.



Obrázek 2.1. Ukázka chroma subsample

2.3.6 AV1

AV1 kodek[11] je nejnovějším kodekem, který se na trhu poprvé objevil v březnu 2018, původně byl označován jako VP10 coby nástupce kodeků VP8 a VP9, nicméně od tohoto názvu se upustilo po zahájení spolupráce ostatních společností s Google (vlastník VP8 a VP9).

Kodek je vyvíjený aliancí `Alliance for Open Media`[3]. Tato aliance spojuje většinu velkých hráčů na poli audio-video, od Amazonu, Adobe přes Google, Netflix až po IBM. Formát je zamýšlen jako náhrada VP9 a hlavní předností je jeho otevřenost a lepší komprese nežli nejpopolárnější formát H.264.

Tento formát, ačkoliv ještě ne plně podporovaný v prohlížečích, má největší šanci prorazit jako univerzální kodek pro distribuci videa ke koncovým uživatelům přes internet,

[1] `https://youtube.com`
[2] `https://www.streamingmedia.com`
[3] `https://aomedia.org/`

jelikož se na jeho vývoji podílí všichni tvůrci webových prohlížečů, oproti dřívějším kodekům, které byly ve velké míře uzavřeným vlastnictvím společností, které za využívání v aplikacích požadovali poplatky.

2.3.7 Výpočty velikostí souborů

V níže uvedené tabulce 2.2 naleznete výpočet velikostí videa pro jednotlivé formáty. Jako délku záznamu jsme volili 90 minut (standardní délka přednášky) a rozlišení FullHD. Vzhledem k velikosti formátů jsme se rozhodli udělat pouze minutové sample záznamy z přednášek místnosti „KN:E-107“ a výsledek vynásobit 90.

Formát	Rozlišení	Framerate	Velikost
H.264	1920x1080	23.98	52.03 GB
H.264	1920x1080	29.97	65.03 GB
H.264	1920x1080	59.94	130.07 GB
ProRes 422 LT	1920x1080	23.98	53.80 GB
ProRes 422 LT	1920x1080	29.97	67.24 GB
ProRes 422 LT	1920x1080	59.94	134.47 GB
ProRes 422	1920x1080	23.98	77.53 GB
ProRes 422	1920x1080	29.97	96.90 GB
ProRes 422	1920x1080	59.94	193.80 GB
ProRes 422 HQ	1920x1080	23.98	116.03 GB
ProRes 422 HQ	1920x1080	29.97	145.02 GB
ProRes 422 HQ	1920x1080	59.94	290.04 GB
ProRes 4444	1920x1080	23.98	174.05 G B
ProRes 4444	1920x1080	29.97	217.53 GB
ProRes 4444	1920x1080	59.94	435.06 GB
Bez komprese (8bit)	1920x1080	23.98	750.00 GB
Bez komprese (8bit)	1920x1080	29.97	937.00 GB
Bez komprese (8bit)	1920x1080	59.94	1830.00 GB
Bez komprese (10bit)	1920x1080	23.98	937.78 GB
Bez komprese (10bit)	1920x1080	29.97	1140.00 GB
Bez komprese (10bit)	1920x1080	59.94	2290.00 GB

Tabulka 2.2. Vypočtené velikosti souborů v různých formátech s různým nastavením.

2.3.8 Srovnání kodeků

Srovnání kodeků je problematické, většina dostupné literatury se ve výsledcích rozchází, což je primárně způsobeno rozdílnými vzorky videa určenými pro testování.

Bohužel většina literatury tyto vzorky popisuje pouze slovně, například: „Nahrávka v parku“, toto je silně nedostačující, jelikož nemáme informace o tom, jestli je ve scéně pohyb, jestli se s kamerou otáčí/zoomuje a také neznáme kompozici. Jarní scéna je povětšinou tvořena podobnými odstíny zelené, kdežto podzimní scéna je daleko bohatší na barvy a tím hůře komprimovaná některými kodeky.

Problematiku videa řeší i velké společnosti, například Netflix. Pravidelně zveřejňuje články o svých výzkumech a pro nás nejzajímavější se ukázal článek se srovnáním kodeků a proč společnost Netflix nevybírá pouze jeden kodek[12]. Rozhodli jsme se tudíž vycházet z těchto výsledků a místo jediného ideálního řešení hledat takové které je nejvíce podporované u koncových uživatelů a mít řešení připravené na případnou změnu kodeku.

Kodek	Firefox	Chrome	Safari	Opera	Edge
H.264	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**
H.265	Ne	Ne	**Ano**	Ne	**Ano**[1]
VP8	**Ano**	**Ano**	**Ano**	**Ano**	**Ano**
VP9	**Ano**	**Ano**	Ne	**Ano**	Ne
AV1	**Ano**	**Ano**	Ne	**Ano**	**Ano**[2]

[1] Pouze pokud je podporováno hardwarovou akcelerací
[2] Musí se povolit v prohlížeči

Tabulka 2.3. Podpora kodeků v prohlížečích

2.3.9 Závěr

Je pro nás nutné zvolit dva kodeky, první, do kterého se bude video nahrávat a zároveň ve kterém se bude video editovat, a druhý, který bude sloužit k distribuci koncovým uživatelům a skladování.

V samotné aplikaci toto ponecháme jako volitelný parametr, nicméně pro distribuci koncovým uživatelům prozatím doporučujeme, dle tabulky 2.3, **kodek** *H.264* **pro videa o maximálním rozlišení FullHD a kodek** *VP9* **pro videa o rozlišení 4k a 8k**. V okamžiku plné podpory a bezproblémového běhu *AV1* ve webových prohlížečích doporučujeme přejít na tento kodek.

Pro záznam a editaci nemáme úplně jasnou odpověď, na základě srovnání a parametrů vychází nejlépe kodeky ze skupiny ProRes, nicméně přesný kodek bude záviset na možnostech serverů a sítě. Nicméně si myslíme že kodek *ProRes 422 ve standardní kvalitě* by měl plně vyhovovat.

2.4 Rozhraní pro přenos záznamu

V této sekci se zaměříme pouze na rozhraní, která se již na fakultě využívají a jsou využívány ve větší míře. Je nicméně nutné čtenáře informovat, že toto nejsou jediná rozhraní mimo fakultu, či i na fakultě. Nicméně jsme se rozhodli tento výběr omezit na trojci rozhraní, která se na fakultě využívají u naprosté většiny zařízení a pro srovnání rozhraní S-Video, které se na fakultě využívalo v minulosti.

2.4.1 S-Video

Toto, v dnešní době již zastaralé, rozhraní využívá k přenosu analogový signál vedený po dvou na sobě nezávislých linkách. Maximální rozlišení tohoto rozhraní je výrobcem uváděno na 720x480px. Nicméně toto je limit pro chroma/barevný kanál a černobílý/luministence kanál může dosahovat vyšších rozlišení[13]. Mimo to je díky analogovému přenosu dat ovlivněna výstupní kvalita v závislosti na vzdálenosti přenosu. Rozhraní zde uvádíme pouze z informativních důvodů, jelikož v minulosti tvořila primární část nahrávacích kamer na fakultě.

2.4.2 HDMI

HDMI rozhraní patří mezi nejznámější rozhraní pro přenos zvuku a videa ve výpočetní technice. Ve velkém se využívá pro přenos videa a zvuku mezi počítačem a monitorem, nicméně své využití má i u kamerových systémů. Rozhraní přenáší nekomprimované video a nekomprimované (či i komprimované) audio. Od svého začátku prošlo několika iteracemi, které toto rozhraní udržují stále relevantní.

Rozhraní vzniklo v roce 2003 spoluprací 7 společností a v současné době se o jeho rozvoj stará 83 společností spolupracujících v rámci skupiny „HDMI Forum“.

Nejnovější verze rozhraní zvládá přenášet 8k (7680x4320) obraz při 30Hz obnovovací rychlosti, případně až do rychlosti 120Hz za použití takzvané technologie „Display Stream Compression“.[1]

2.4.3 SDI

SDI (zkratka pro „Serial Digital Interface“, v překladu „Sériové digitální rozhraní“) je rozhraní vytvořené v roce 1989 a několikrát rozšiřované pro podporu vyšších rozlišení.

Signál se přenáší za pomocí koaxiálního kabelu, umožňuje přenos nekomprimovaného a nekódovaného video signálu s možností přenosu audio signálu. SDI většinou vidíme jen u profesionálních zařízení z důvodu licence rozhraní.

2.4.4 NDI

NDI (zkratka pro „Network Device Interface“, v překladu „Síťové rozhraní zařízení“) je rozhraní vytvořené a licencované firmou Newtek a k přenosu využívá počítačovou síť.

V síti NDI rozhraní podporuje multicast, nicméně v defaultním nastavení pouze pro akce „discovery“ (vyhledávání NDI zařízení po síti) a „register“ (přihlašování k NDI zdroji), nikoliv pro samotný přenos záznamu, který běží v unicast režimu. Nastavení lze změnit na novějších NDI zařízení tak, aby i přenos záznamu probíhal v multicast režimu.

2.4.5 Srovnání

V tabulce 2.4 naleznete srovnání jednotlivých rozhraní podle jejich podpory různých rozlišení. Můžeme vidět, že pro naše cílené formáty 1080p a 4k je dostačující pouze NDI a HDMI, SDI je použitelné pouze pro 1080p. Informace jsou založené na informacích z webů organizací `Newtek - NDI`[2], `HDMI Forum - HDMI`[3], `SMPTE - SDI`[4]. S-Video je založené na své fyzikální limitaci analogovaného signálu přenášeného po dvou kabelech, dosahující průměrně rozlišení 480p.

Rozlišení	S-Video	SDI	NDI	HDMI
720p	Ne	**Ano**	**Ano**	**Ano**
1080p	Ne	**Ano**	**Ano**	**Ano**
1440p	Ne	**Ano**	**Ano**	**Ano**
4k	Ne	**Ne**	**Ano**	**Ano**
5k	Ne	Ne	Ne	**Ano**
8k	Ne	Ne	Ne	**Ano**

Tabulka 2.4. Podporované rozlišení videa (Obnovovací rychlost 30Hz)

2.4.6 Závěr

Každá zmíněná technologie (až na zastaralé *S-Video*) má své výhody i nevýhody. Pro případy, kdy nemusíme zpracovávat více 4k video zdrojů, plnohodnotně stačí využívání *HDMI* a *SDI* svedené do jediné stanice.

[1] `https://vesa.org/vesa-display-compression-codecs/`
[2] `https://www.newtek.com/`
[3] `https://hdmiforum.org/`
[4] `https://www.smpte.org/`

Nicméně v případě, že video zdroj je využívaný více zařízeními/aplikacemi tak je ideální technologie *NDI*, která podporuje multicast, neboli posílání dat mezi více stroji v síti. Rozdvojení signálu lze dosáhnout i u jiných technologií, nicméně za cenu přídavného hardware a kabeláže mezi všemi zařízeními.

V případě situace, kde bychom chtěli využívat 8k kamery, tak jako jediná možná technologie vychází *HDMI*. Nicméně nesmíme opomenout ani *SDI*, které má své výhody v podobě dlouhého dosahu. Za pomocí optických kabelů může SDI dosahovat až na vzdálenost 5km, nicméně pro naše potřeby toto není potřeba.

Konečný výběr mezi technologiemi tedy necháváme bez odpovědi, jelikož výběr bude ve velké míře záviset na účelu a vybavenosti dané místnosti.

2.5 Souborový formát pro ukládání záznamu

Souborový formát udává, jak jsou záznamy uloženy na disku, může se zdát že tato práce je nadbytečná a že by mělo dostačovat uložení binárního streamu záznamu spolu s metadaty o typu kodeku, nicméně opak je pravdou. Souborové formáty umožňují uložení více záznamů v jednom souboru (například rozdílné jazykové audio stopy), barevné profily, titulky a mnoho dalšího.

Výběr formátů pro naše potřeby byl jednoduchý, dle dostupné literatury[14], **jsme se rozhodli pro formát Matroska (.mkv)**. K tomuto kroku nás vedlo hlavně to, že všechny vyjmenované kodeky v minulé sekci podporuje pouze Matroska a MPEG-4 Part 14 (.mp4).

Nicméně jak se z literatury[15] ukázalo, nahrávání do MPEG-4 Part 14 je nebezpečné. Dovolím si vypsání krátkého odstavce z této literatury vysvětlující situaci:

```
   The reason MP4 is so terrible for direct recording is
because MP4, much like MOV before it, requires a
finalization process to take place before the file
is usable. What this process does is write the "atoms"
of the file, which are bits of information about what
the MP4 file actually contains, including the codec that
it's encoded in and various other bits of metadata.
   The atoms, and more specifically the "moov" atom
or movie atom, are a key part of the structure of
an MP4 file, and the file is incomplete without them.
   They serve as an index for the contents of the entire
file, and if your video player can't find an index,
it doesn't know what to do with the file, and it becomes
pretty useless.
```

V překladu se jedná o problém, že do souboru jsou zapisovány kódované části obrazu bez jejich vzájemné pozice. V případě, že se záznam nedokončí, neuloží se záznam o pozicích a soubor je nepoužitelný.

Problém si můžeme představit jako puzzle, máme sice všechny dílky, ale netušíme kam daný dílek patří. A jediným řešením je postupné slepování dílků, které obsahují podobné části.

2.6 Nástroj pro tvorbu záznamu

Jakožto nástroj pro tvorbu záznamu jsme hledali freeware nástroj, který by umožňoval:

- Hardwarovou akceleraci

- Multiplatformní nasazení
- Využívání souborového formátu Matroska
- Nahrávání/konvertování do H.264, VP9 a nejlépe i AV1
- Headless ovládání, neboli ovládání bez uživatelského rozhraní

Toto splňovali tři nástroje: ffmpeg, libav a OBS. Níže si blíže tyto nástroje přiblížíme a provedeme jejich srovnání na základě našich požadavků.

2.6.1 Ffmpeg

Software ffmpeg patří k jedním z nejstarších a nejrozšířenějších nástrojů na úpravu a převod videa. Spousta, hlavně grafických editorů videa, na pozadí využívá právě ffmpeg.

Během našeho prohledávání literatury jsme nenarazili na žádnou funkcionalitu kterou by ffmpeg neuměl, či neměl pro ní přídavnou knihovnu.

2.6.2 Libav

Software Libav vznikl odtržením části vývojářů *ffmpeg*. Základ zdrojového kódu libav je kopie kódu *ffmpeg*, který je postupně přepisován do optimalizovanější podoby. Vývoj software probíhá odlišně od *ffmpeg*, vývojáři se snaží klást důraz na kvalitu dokumentace, API a kódu za cenu méně funkcí a delšími termíny na vydání nové verze.

Bohužel mezi oběma tábory stále panují neshody, tým *ffmpeg* kopíruje veškerou práci týmu *Libav* do svého projektu a může se díky tomu věnovat novým funkcionalitám. Zatímco tým *Libav* se plně distancuje od jakýchkoliv změn a nových funkcionalit vytvořených týmem *ffmpeg*.

2.6.3 OBS

OBS (zkratka pro „Open Broadcast System“) je software, který se ve velké míře rozšířil až v posledních letech s nástupem herního streamování na platformě `twitch.tv`[1]. V současné době je primárním zdrojem živých záznamů na již zmíněném twitch.tv a na platformách `facebook`[2], `youtube`[3].

V základní instalaci obsahuje pouze kodeky pro *H.264* a *H.265*, nicméně je rozšiřitelný o kodeky a formáty z balíčku *libavcodec* a *libavformat*, či pomocí knihoven třetích stran.

Staví na jednoduchosti a za uživatele řeší automaticky spoustu problémů, které se v jiných softwarech musí řešit manuálně. Například i upozorňuje uživatele, že jím zvolené nastavení je nevhodné, což je obrovským přínosem pro méně zkušené uživatele. Nicméně toto je zároveň i jeden z hlavních problémů *OBS*, jelikož pro většinu svého nastavení vyžaduje využití svého grafického rozhraní.

2.6.4 Srovnání

Pro naše potřeby automatického nahrávání vyžadujeme, aby daný software umožňoval takzvaný „headless“ mód, neboli podporu plného ovládání přes příkazovou řádku, api, či scriptovací jazyk. Bez této možnosti by vzdálené nahrávání bylo značně složité a vyžadovalo emulaci myši pro grafické rozhraní software.

Díky tomuto požadavku jsme zamítli použití *OBS*. OBS umožňuje částečné ovládání přes API a příkazovou řádku, nicméně toto částečné ovládání nevyhovuje. Jako největší

[1] `https://www.twitch.tv/`
[2] `https://www.facebook.com/`
[3] `https://www.youtube.com/`

problém je nemožnost detekce chyby na vstupním streamu pomocí API a zároveň i nemožnost přenastavení vstupního streamu na jiný pomocí API či konzole.

Při srovnání *Ffmpeg* a *Libav* vidíme jako hlavní přednosti Ffmpeg širokou podporu kodeků, více funkcionalit, rychlejší hotfixy a slučování všech nových funkcionalit libav. Na druhé straně hlavní přednost Libav je čistota API a kódu, která často vede k efektivnějšímu využití výkonu. Nicméně pro naše potřeby z předchozí sekce požadujeme podporu NDI technologie a tato bohužel v současné chvíli v *Libav* chybí. **Tudíž jako software pro tvorbu záznamu volíme Ffmpeg.**

2.7 Požadavky na nahrávací hardware

Požadavky na nahrávací hardware jsou špatně specifikovatelné a ve velké míře bude záležet na záznamovém vybavení jednotlivých místností. Například některé NDI zařízení umí vlastní H.264 kódování a nahrávací zařízení tedy pouze ukládá přijímaný záznam.

V případě potřeby kódování na nahrávacím zařízení narážíme na několik problémů. Prvním je přítomnost dedikované grafické karty (nejlépe karta vyšší řady, například Nvidia Titan) pokud je přítomna tak nemusíme řešit dále. V opačném případě nás začne zajímat procesor a jeho hardwarová akcelerace pro kódování videí.

V tabulce 2.5, jsme se rozhodli zmínit pouze procesory od společnosti Intel, které v fakultních strojích a serverech převažují drtivou většinou. Tato tabulka byla převzata přímo ze stránek společnosti Intel. Jak můžeme vidět, již relativně staré série *Intel Braswell* a *Intel Skylake* obsahují hardwarovou akceleraci pro námi zvolené kodeky *H.264* a *VP9*.

Bohužel jsme ale až během testování zjistili, že více-socketové varianty procesorů Xeon tyto hardwarové akceleraci neposkytují. Toto byl zároveň i jeden z důvodů proč došlo ke zvolení Prores kodeku jako mezistupně pro nahrávání a editaci.

Prores kodek neklade vysoké nároky na dedikované grafické karty a postačí si i se silnějšími procesory, nicméně díky velikosti záznamu (viditelné v tabulce 2.2) vyžaduje využití rychlých SSD disků. Toto je dle našeho mínění dostatečný kompromis, který bude vyhovovat ve většině scénářů.

	H.264	H.265	VP8	VP9	AV1
Intel Sandy Bridge	**Ano**	Ne	Ne	Ne	Ne
Intel Ivy Bridge	**Ano**	Ne	Ne	Ne	Ne
Intel Haswell	**Ano**	Ne	Ne	Ne	Ne
Intel Broadwell	**Ano**	Ne	Ne	Ne	Ne
Intel Braswell	**Ano**	Ne	**Ano**	Ne	Ne
Intel Cherry Trail	**Ano**	Ne	**Ano**	Ne	Ne
Intel Skylake	**Ano**	**Ano**	**Ano**	Ne	Ne
Intel Apollo Lake	**Ano**	**Ano**	**Ano**	Ne	Ne
Intel Kaby Lake	**Ano**	**Ano**	**Ano**	**Ano**	Ne
Intel Gemini Lake	**Ano**	**Ano**	**Ano**	**Ano**	Ne
Intel Coffee Lake	**Ano**	**Ano**	**Ano**	**Ano**	Ne
Intel Cannon Lake	**Ano**	**Ano**	**Ano**	**Ano**	Ne
Intel Ice Lake	**Ano**	**Ano**	**Ano**	**Ano**	Ne
Intel Ivy Bridge	**Ano**	**Ano**	**Ano**	**Ano**	Ne

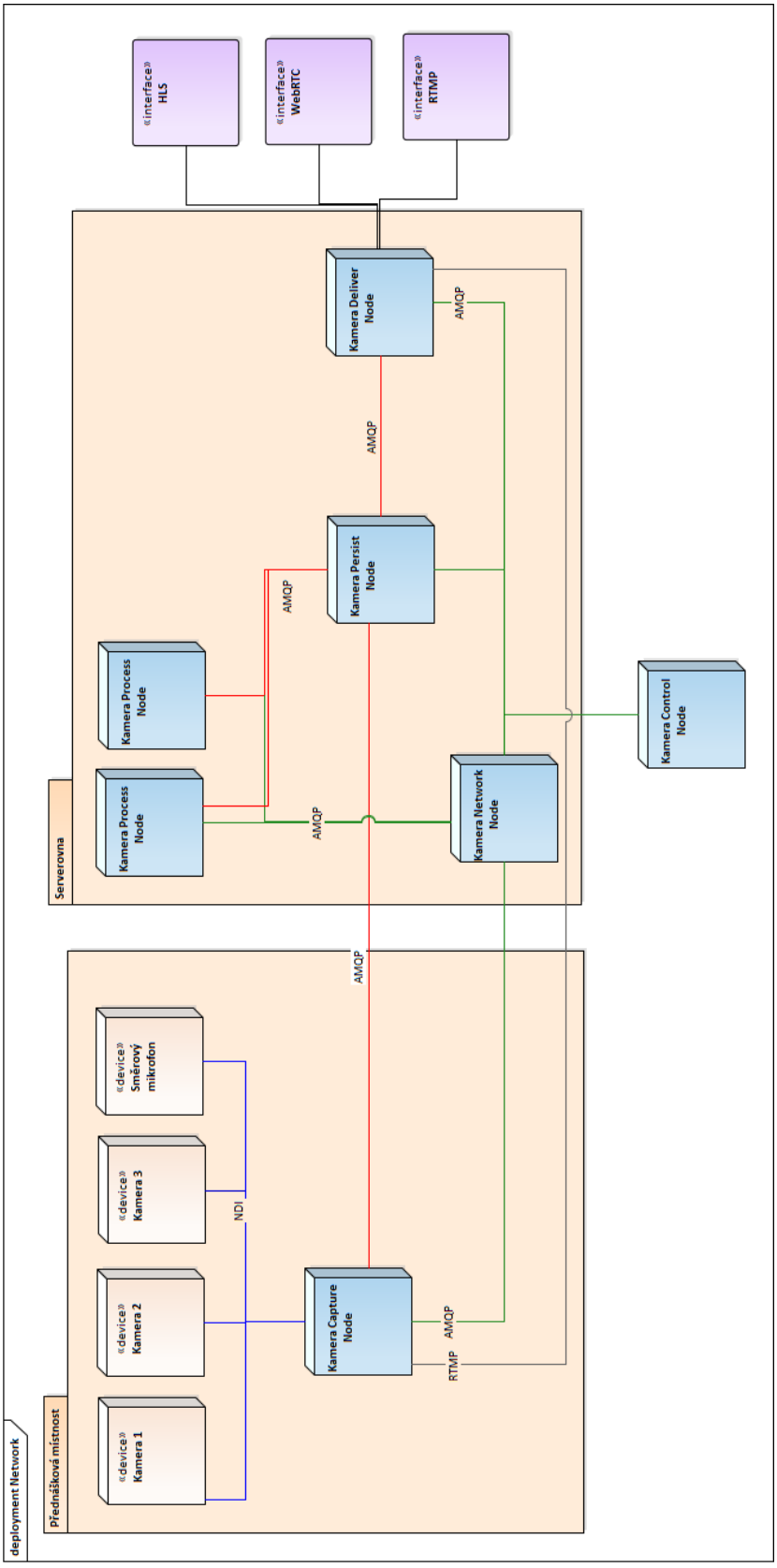
Tabulka 2.5. Hardwarová akcelerace v procesorech (Více-socketové řady proceserů Xeon nepodporují žádnou formu hardwarové akcelerace)

Kapitola 3
Architektura řešení

Jak jsme zjistili v předchozí kapitole. Nároky na přenosové rychlosti, výkon a datové prostory jsou značně vysoké. Vyšší než jaké bychom dokázali poskytnout pouze s jedním výpočetním strojem na všechny místnosti. Rozhodli jsme se tudíž pro distribuovaný systém, skládající se z takzvaných uzlů.

V této kapitole se budeme často odvolávat na obrázek 3.1 s navrženou architekturou sítě a doporučujeme si ho dopředu projít. Pro přehlednost jsme ho umístili na celou následující stránku.

V této a dalších kapitolách budeme často zmiňovat pojem „**síť**“. Tímto pojmem myslíme naší vlastní implementovanou síť mezi uzly, nikoliv síť ve smyslu „internetová síť“.



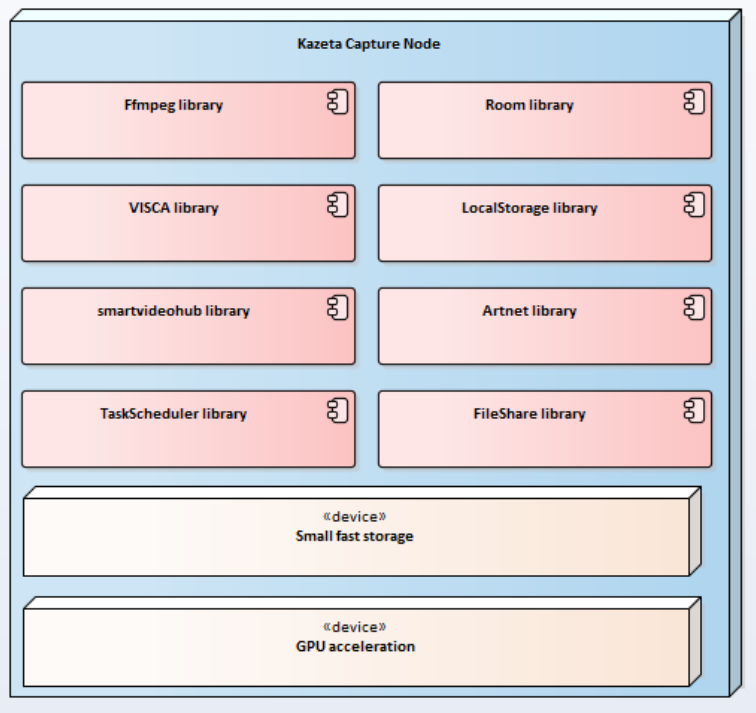
Obrázek 3.1. Struktura distribuované sítě „Kazeta“

Jak můžeme vidět na obrázku 3.1, architekturu dělíme do dvou logických celků. Část která je umístěna v nahrávané místnosti a část která je někde v pozadí a je sdílená napříč místnostmi. Část v nahrávací místnosti se stará o nastavení zařízení místnosti a nahrávání záznamů z dané místnosti. Po úspěšném nahrání, odesílá celý záznam do druhé části, která se stará o archivaci, zpracování do výstupního formátu a distribuci.

3.1 Druhy uzlů

Práce, které jsou potřeba pro celý proces záznamu, následně rozdělujeme na 3 primární druhy (capture, process, deliver) a 3 sekundární druhy (control, network, persist). Každý uzel naplňuje jeden či více druhů prací.

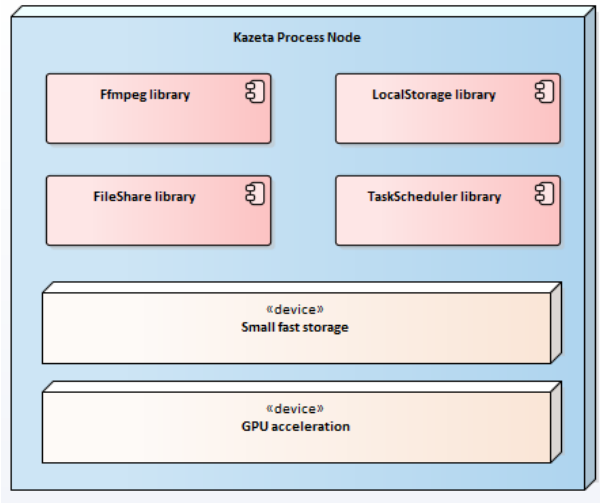
3.1.1 Primární druh uzlu „Capture“



Obrázek 3.2. Knihovny a vybavení capture uzlu

Takzvané „Capture“ uzly se starají o proces nahrávání v sále a následné odeslaní na přidělený „Persist“ uzel. Většinou budou umístěny přímo v místnostech, či uzavřeny ve vlan síti se zařízeními místnosti, kvůli omezení přístupu na zařízení.

3.1.2 Primární druh uzlu „Process“



Obrázek 3.3. Knihovny a vybavení process uzlu

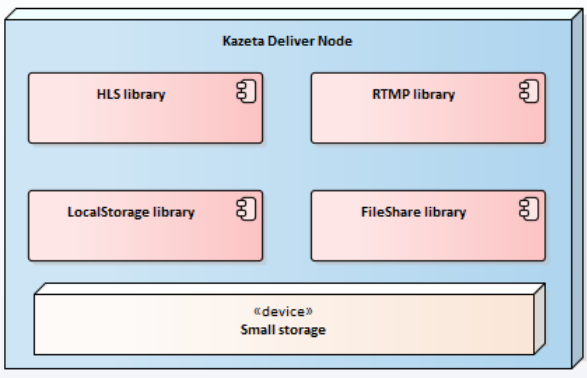
„Process“ uzly mají za úkol upravovat nahrané záznamy. Mezi jejich běžné operace patří slučování více zvukových a video záznamů do jednoho, střih videa a konvertování do výstupních formátů.

Na rozdíl od „Capture“ uzlů od nich není vyžadováno, aby dané operace probíhaly v reálném čase. Tyto uzly také mohou mít rozdílnou hardwarovou, či operační vybavenost, nicméně všechny dokáží vykonávat stejnou práci, liší se pouze rozdílným časem pro vykonání operace.

Nové operace si uzly stahují z fronty požadavků na zpracování. Tato fronta je přednastavená jako takzvaná „priority queue“. Tudíž v budoucích verzích programu (nebylo implementováno jako součást diplomové práce) bude možné například upřednostňovat zpracování speciálních akcí před přednáškami předmětů.

Je také nutné podotknout, že tato fronta bude muset být i sledována, může dojít k situaci, kdy počet nahrávaných záznamů převýší výpočetní schopnosti a síť přestane stíhat zpracovávat nové požadavky.

3.1.3 Primární druh uzlu „Deliver“

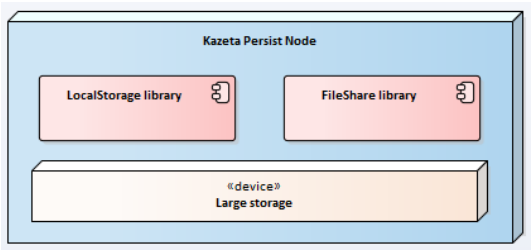


Obrázek 3.4. Knihovny a vybavení deliver uzlu

„Deliver“ uzly nebyly implementovány jako součást diplomové práce, nicméně budou sloužit k finální distribuci záznamů k uživatelům. Zde jsou uváděny pouze pro úplnost.

Uzly budou mít k dispozici záznamy uložené na „Persist“ uzlech a při požadavku od koncového uživatele, přes zatím nespecifikovaný protokol (například WebRTC), mu budou záznamy poskytovat. S největší pravděpodobností budou obsahovat i cachování, jelikož je pravděpodobné že nově nahrané záznamy budou častěji žádané a cachováním dojde k úspoře přenosové kapacity sítě.

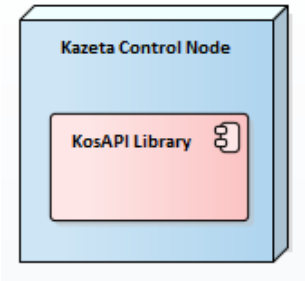
3.1.4 Sekundární druh uzlu „Persist“



Obrázek 3.5. Knihovny a vybavení persist uzlu

„Persist“ uzly slouží jako dlouhodobá úložiště pro záznamy a zajišťují posílání záznamů mezi uzly. Úložiště jako takové se definuje v konfiguračním souboru popsaném v kapitole 5.

3.1.5 Sekundární druh uzlu „Control“

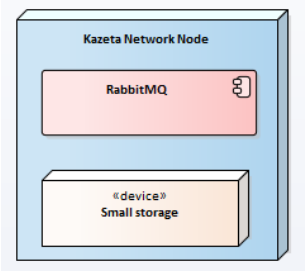


Obrázek 3.6. Knihovny a vybavení control uzlu

Uzel označujeme jako „Control“ pokud je využíván správcem pro správu sítě. Nejedná se tedy o nijak specifický uzel, nemusí být ani nasazen na serveru a může být spouštěn z počítače správce.

Tento druh existuje pouze z důvodu záznamu provedených akcí v síti pro situace kdy došlo k napadení sítě, či je potřeba odhalit příčinu chyby.

3.1.6 Sekundární druh uzlu „Network“



Obrázek 3.7. Knihovny a vybavení network uzlu

Uzly označené „Network“ nejsou plnohodnotnými uzly naší aplikace, jelikož na nich ve většině případů nepoběží naše řešení a budou pouze obsahovat instanci RabbitMQ, která poskytuje komunikační schopnosti řešení. Díky tomu jsou neodmyslitelnou součástí sítě.

3.2 Komunikace uzlů

Uzly mezi sebou komunikují pomocí protokolu AMQP, implementovaného v RabbitMQ nástroji. Protokol je založený na systému kanálů a front, kde kanál ovládá logiku směrování zpráv a fronta ovládá logiku přijímaní zpráv. Počítáme s tím, že naše řešení poběží v takzvaném virtuálním hostu a nebude rušeno jinými aplikacemi.

Kanály slouží k rozeslání zpráv vícero uzlům s určitou specifikací, například uzlům určeným pro zpracovávání videa či pro uzly, které dokáží nahrávat určitou místnost. V případě že chceme odeslat zprávu všem, tak využíváme „discovery“ kanál.

V našem řešení budeme využívat jediný kanál „kazeta.network“ popsaný v tabulce 3.1, jedná se o takzvaný Topic kanál, což znamená že lze nastavit cesty podle kterých budou zprávy chodit různým uzlům. Předdefinované uzly naleznete v tabulce 3.2.

Proměnná	Hodnota	Význam
Type	Topic	Cesty do front se nastavují podle názvů topic
Durability	Durable	Kanál je ukládán a dokáže se obnovit v případu pádu sítě
Auto delete	No	Kanál nebude smazán v případě že žádný uzel není připojen

Tabulka 3.1. Nastavení komunikačního kanálu „kazeta.network“

Cesta	Použití
kazeta.route.discovery	Pro komunikaci se všemi uzly
kazeta.route.capture	Pro komunikaci s capture uzly
kazeta.route.capture.room.[roomid]	Pro komunikaci s captury uzly které dokáží nahrávat z místnosti [roomid]
kazeta.route.process	Pro komunikaci s process uzly
kazeta.route.persist	Pro komunikaci s persist uzly
kazeta.route.deliver	Pro komunikaci s deliver uzly
kazeta.route.control	Pro komunikaci s control uzly

Tabulka 3.2. Nastavení komunikačních cest

Fronty dělíme na veřejné a privátní. Každý uzel má svojí vlastní privátní frontu, přes kterou s ním lze komunikovat přímo, či do ní dostává zprávy z kanálu „kazeta.network“.

Veřejné fronty slouží k rozdělování práce, každý uzel si z nich bere novou práci v případě že současnou již dodělal, případně se přepne do čekacího režimu, dokud do fronty nepřijde nová zpráva.

Seznam front můžete nalézt v tabulce 3.3 a základní nastavení těchto front naleznete v tabulce 3.4.

Fronta	Veřejná	Použití
kazeta.queue.node.[nodeid]	Ne	Privátní fronta uzlu
kazeta.queue.jobs.persist	Ano	Fronta pro zisk prací na uložení projektu
kazeta.queue.jobs.process	Ano	Fronta pro zisk prací na zpracování projektu
kazeta.queue.jobs.deliver	Ano	Fronta pro zisk prací na doručení projektu

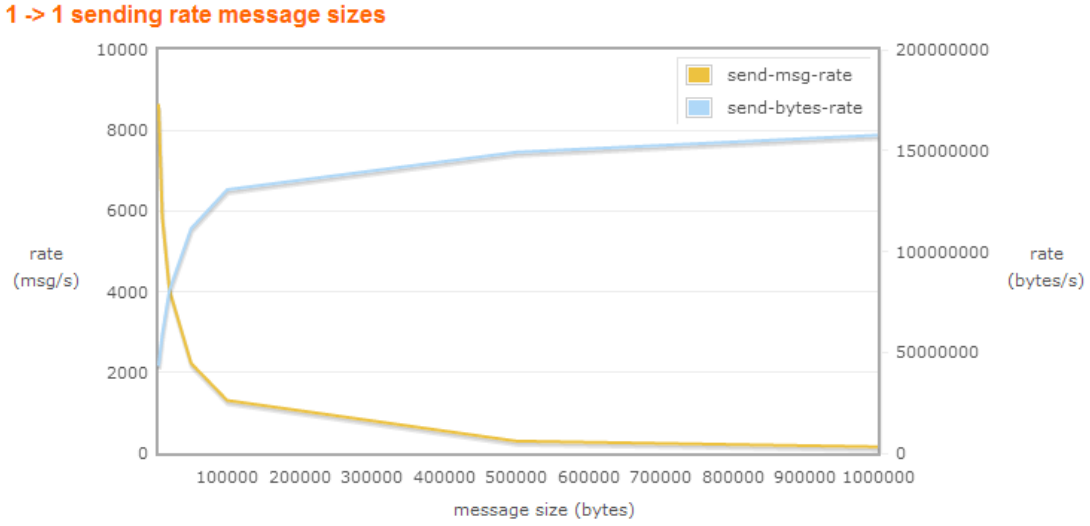
Tabulka 3.3. Komunikační fronty

Proměnná	Hodnota	Význam
Durability	Durable	Fronta je ukládána a dokáže se obnovit v případu pádu sítě
Auto delete	No	Fronta nebude smazána v případě že žádný uzel není připojen
Message TTL	Unlimited	Zpráva zůstane ve frontě dokud není vyzvednuta uzlem
Auto expire	72h	Jak dlouho má fronta existovat bez připojeného uzlu (72 hodin)
Max length	8192	Fronta může obsahovat až 8192 zpráv než začne odebírat nejstarší zprávu
Max length bytes	250MB	Zpráva může mít maximální délku 250MB
Maximum priority	10000	Lze nastavit priority zpráv od 0 až do 9999
Dead letter exchange	kazeta.dead	Do kanálu „kazeta.dead“ budou přeposílány nevyzvednuté zprávy

Tabulka 3.4. Základní nastavení komunikačních front

3.2.1 Rychlost sítě

Jak můžeme vidět na obrázku 3.8 a v literatuře[16] RabbitMQ nemá problémy s rychlostí zpráv. Pro naše potřeby, a to i ty na posílání 4k videí mezi uzly, je naměřená hodnota 150MB dat za sekundu plně dostačující. Mimo jiné proto, že nevyžadujeme živé vysílání, či živý přesun záznamů.



Obrázek 3.8. Rychlost posílání zpráv přes RabbitMQ (Převzato z VMware[16])

Zvýšení počtu uzlu	Následek
capture	Lepší pokrytí místností
process	Rychlejší zpracování záznamů
deliver	Zvýšení kapacit pro streamování uživatelům
persist	Zvýšení rychlosti, stability a velikosti úložiště záznamů
network	Stabilnější a rychlejší síť

Tabulka 3.5. Škálovatelnost sítě

3.3 Škálovatelnost

Škálovatelnost patří k největším výhodám této sítě. Dokážeme se plně přizpůsobit uživatelským požadavkům podle jednoduchého vzoru, jelikož každý modul odpovídá jiné části škálovatelnosti, viz tabulka 3.5.

3.4 Stabilita

Díky navržené architektuře oddělujeme zdlouhavé procesy stříhání a úprav již nahraného videa od nahrávání nového, či od distribuce výstupních videí koncovým uživatelům. Díky tomuto nejsou procesy navzájem ovlivňované.

Z hlediska sítě dokáže každý uzel fungovat odděleně až do doby, než je síť znovu dostupná. Vyjímku tvoří pouze přidávání nových žádostí o nahrání, které, pokud nevidí žádný uzel, který by dokázal žádost o nahrání splnit, skončí chybou.

Kapitola 4
Implementace

V této kapitole probereme proces a průběh implementace řešení.

4.1 Použité nástroje a technologie

4.1.1 Vývojové nástroje

Při návrhu řešení byl využit Enterprise Architect od společnosti Sparx Systems. Společnost poskytuje studentské licence, nicméně byla využita privátní licence na používání systému.

Při vývoji řešení bylo použito vývojové prostředí Microsoft Visual Studio 2017, které společnost Microsoft poskytuje studentům zdarma ve verzi Ultimate. Ke spravování zdrojových kódů byl využit verzovací nástroj Git. Repozitář obsahující zdrojové kódy aplikace je umístěný na fakultní instanci nástroje Gitlab.

4.1.2 Jazyk C# a prostředí .NET Core 2.1

Rozhodli jsme se implementovat nástroj v jazyce C# díky zkušenostem s jazykem a možnostem jazyka při práci s C++ pluginmi a ostatními běžícími procesy.

Prostředí .NET Core 2.1 je universální mezi operačními systémy Linux, Windows a Mac OS. Všechny námi potřebné funkcionality, které jsou rozdílně řešené napříč operačními systémy, implementuje přímo v sobě a my tudíž nemusíme řešit rozdílné spouštění a kontrolu procesů, ovládání socketů a mnoho dalšího.

4.2 Nasazení RabbitMQ

V první fázi implementace jsme nasadili službu RabbitMQ na testovací server. A začali testovat jednotlivé protokoly, které RabbitMQ poskytuje, primárně AMQP rozhraní.

Pro naše potřeby jsme primárně potřebovali otestovat:

- Maximální přenosovou velikost
- Chování při ztrátě spojení
- „Heartbeat“ funkce, která se periodicky dotazuje připojených uzlů pro otestování spojení.
- Jak přesně fungují kanály a fronty

K našemu překvapení vše funguje s pluginmi od RabbitMQ hladce a sama plugin řeší většinu problémů se spojením, přenosové velikosti taktéž nejsou problémové až do velikosti 500 MB. Což pro naše potřeby plně dostačuje.

Díky správnému nastavení tudíž nemusíme řešit například duplicitní uzly, síť nás sama upozorní, že daný uzel v síti již existuje, viz obrázek 4.1.

```
Kazeta node (version 0.2)
=========================================
Reading configuration file: OK
Connecting to network:
Failed
RabbitMQ.Client.Exceptions.OperationInterruptedException: The AMQP operation was interrupted: AMQP close-reason, initiat
ed by Peer, code=405, text="RESOURCE_LOCKED - cannot obtain exclusive access to locked queue 'app.kazeta.node.Node_1' in
 vhost 'kazeta'", classId=50, methodId=10, cause=
   at RabbitMQ.Client.Impl.SimpleBlockingRpcContinuation.GetReply(TimeSpan timeout)
   at RabbitMQ.Client.Impl.ModelBase.QueueDeclare(String queue, Boolean passive, Boolean durable, Boolean exclusive, Boo
lean autoDelete, IDictionary`2 arguments)
   at RabbitMQ.Client.Impl.AutorecoveringModel.QueueDeclare(String queue, Boolean durable, Boolean exclusive, Boolean au
toDelete, IDictionary`2 arguments)
   at KazetaNode.Coms.Network.Connect() in C:\Users\frycj\Documents\Projekty\kazeta\Coms\Network.cs:line 81
   at KazetaNode.Program.Main(String[] args) in C:\Users\frycj\Documents\Projekty\kazeta\Program.cs:line 69
```

Obrázek 4.1. Ukázka zamykání instancí uzlů v síti

4.3 Tvorba jádra uzlu

Jádro uzlu se stará o hlavní běh programu, načítání konfiguračních souborů, spouštění/vypínání pluginů, vzájemné propojení uzlů a v neposlední řadě poskytuje konzolové rozhraní.

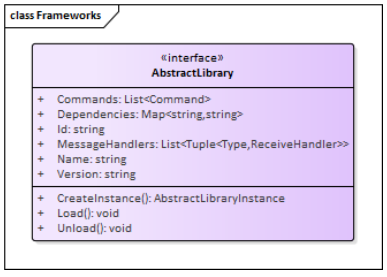
4.3.1 Tvorba rozhraní pluginů

Rozhraní pro pluginy je důležitou součástí programu, jelikož většina problematiky je řešená právě pomocí pluginů. Od lokálního ukládání souborů, přes stahování informací o kurzech z KOS, až po samotné nahrávání.

Rozhraní je defakto jednoduché a lze rozdělit na tři části. První část jsou proměnné ID, Name, Version, Dependencies sloužící k identifikaci a registraci pluginu. Dependencies obsahuje jako klíč id požadovaného pluginu a jako hodnotu regex pro matchování verze pluginu.

Druhou část tvoří metoda Load a Unload sloužící k načtení a deaktivaci pluginu. Metoda pro načítání navíc jako parametr dostává konfigurační hodnoty daného pluginu.

Poslední částí jsou seznamy příkazů a metody pro zpracování zpráv. Tyto se po načtení pluginu automaticky registrují do jádra a při deaktivaci odstraňují.



Obrázek 4.2. Rozhraní knihoven

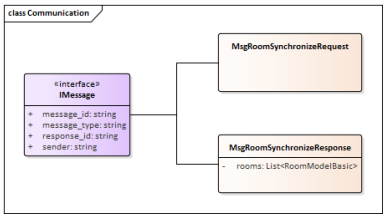
Díky tomuto jednoduchému rozhraní by mělo být pro většinu vývojářů snadné budoucí rozšiřování řešení.

4.4 Tvorba komunikace

Komunikace je základem pro distribuované řešení. Jak již bylo zmíněno vybrali jsme technologii AMQP implementovanou pomocí RabbitMQ.

4.4.1 Formát zpráv

Zprávy posílané přes síť jsou posílány ve dvou formátech. První formát je typu json a má pevně definovanou základní strukturu. Zprávy v tomto formátu musí obsahovat „message_id“ neboli GUID zprávy,



Obrázek 4.3. Ukázka podoby zpráv ve formátu json.

Druhý využívaný formát je binární a slouží k přenosu binárních souborů mezi jednotlivými uzly.

4.4.2 Entita projekt

Entita projekt je naše označení pro nahrávaný záznam. Obsahuje definici projektu, skládající se z id, nastavení záznamu a zpracování, uzel na kterým má být konečný výsledek uložen a mnoho dalšího. Plnou specifikaci můžete nalézt v sekci 5.3.

Entita projekt taktéž obsahuje dva druhy souborů, „temp“ soubory používané jako dočasné soubory využívané při konvertování a záznamu videa a následně klasické soubory. Pouze klasické soubory se distribuují mezi uzly.

ID projektu není náhodné číslo či string, ale jedná se o spojení důležitých vlastností projektu, zakončené náhodnými 6 znaky, viz:

```
String.Join("_", room, year, month, day, hour_start, random)
```

4.4.3 Druhy záznamu

V následujících sekcích se často setkáme s pojmem „druh záznamu“, toto je pro nás logické rozdělení záznamů podle počtu a druhu vstupů. Jedná se o kategorizaci samotného druhu přednášky, na který jsme naráželi v úvodní kategorii této práce.

Druh záznamu určujeme podle druhu vstupu, či kombinace vstupů, kde „SPEAKER“ označuje kamerový vstup zaměřený na přednášejícího, „PRESENTATION“ označuje vstup z promítacího plátnam „BLACKBOARD“ označuje kamerový vstup zaměřený na tabuli a „AUDIENCE“ označuje kamerový vstup zaměřený na publikum.

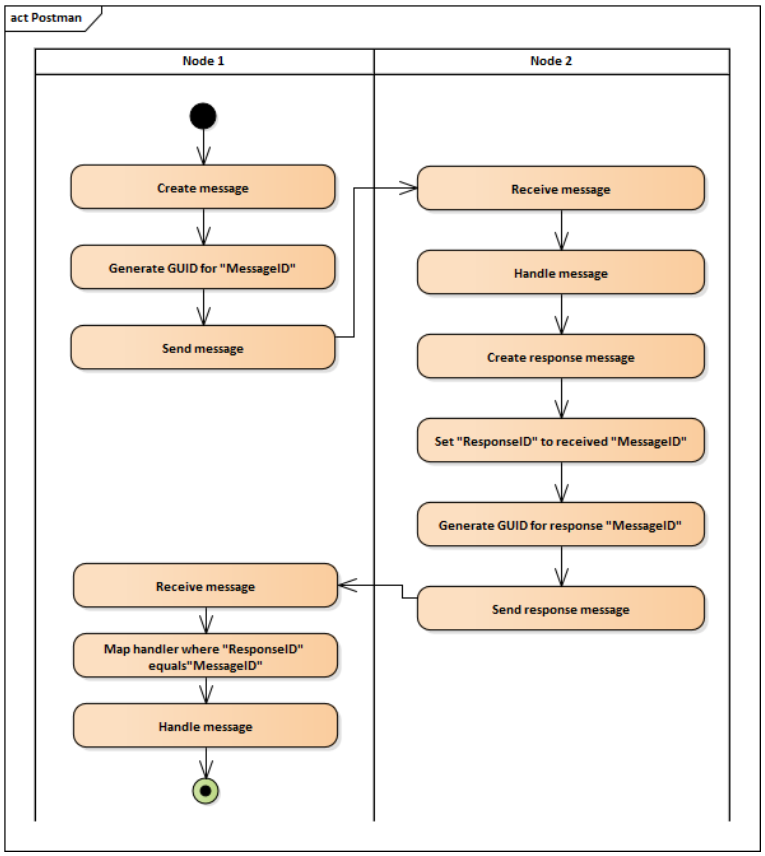
Tudíž máme tyto druhy:

- ALL
- SPEAKER_PRESENTATION_BLACKBOARD_AUDIENCE
- SPEAKER_PRESENTATION_BLACKBOARD
- SPEAKER_PRESENTATION_AUDIENCE
- SPEAKER_BLACKBOARD_AUDIENCE
- SPEAKER_PRESENTATION
- SPEAKER_BLACKBOARD
- SPEAKER_AUDIENCE
- BLACKBOARD_AUDIENCE
- SPEAKER
- BLACKBOARD
- AUDIO

4.4.4 Postman komunikace

Postman komunikace je, jak můžeme vidět v Obrázku 4.4, jednoduchý protokol, který umožňuje mapování odpovědí na jednotlivé zprávy mezi více uzly. Komunikace je založena na principu unikátních GUID, které se generují pro každou zprávu.

Při návratové zprávě uvádíme toto GUID původní zprávy jako ResponseID. Díky tomuto jednoduchému principu dokážeme párovat zprávy i v situacích, kdy po síti běhá několik nezávislých zpráv najednou.



Obrázek 4.4. Průběh postman komunikace

Na ukázku využití se můžete podívat v Obrázku 4.5. Zde na prvním řádku definujeme kanál, přes který chceme komunikovat a odesílanou zprávu. Na dalších řádcích můžeme vidět zpracování odpovědí a na posledním řádku definici timeoutu a že očekáváme vícero odpovědí.

```
Postman.SendToChannel(Channels.Discovery, new MsgRoomSynchronizeRequest()).During(
    msg_raw =>
    {
        MsgRoomSynchronizeResponse msg = (MsgRoomSynchronizeResponse)msg_raw;
            foreach (RoomModelBasic room in msg.Rooms)
            {
                if (!RemoteRooms.ContainsKey(room.Id))
                    RemoteRooms.Add(room.Id, Tuple.Create<HashSet<string>, RoomModelBasic>(new HashSet<string>() { msg.Sender }, room));
                RemoteRooms[room.Id].Item1.Add(msg.Sender);
                ConsoleP.AppendLine("Room '" + room.Id + "' capturable from Node '" + msg.Sender + "'.");
            }
    }).ExpectMultiple().Timeout(5_000).Send();
```

Obrázek 4.5. Synchronizace místností mezi uzly pomocí postman

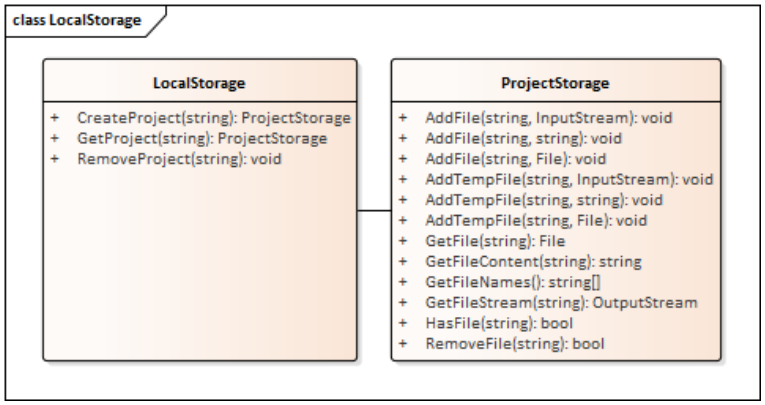
4.5 Tvorba knihoven

Samotná tvorba pluginů byla ve většině případů přímočará ve většině případů pouze implementujeme již definovaný protokol, či napojení knihoven, nicméně pro úplnost zde zmíníme všechny pluginy implementované v rámci DP.

4.5.1 LocalStorage

LocalStorage plugin se stará o lokální úložiště uzlu, ostatním pluginům poskytuje rozhraní pro ukládání, mazání a čtení projektů. Toto slouží nejenom pro dlouhodobé ukládání záznamů, ale taktéž pro krátkodobé, například při záznamu či konvertování videa na uzlu.

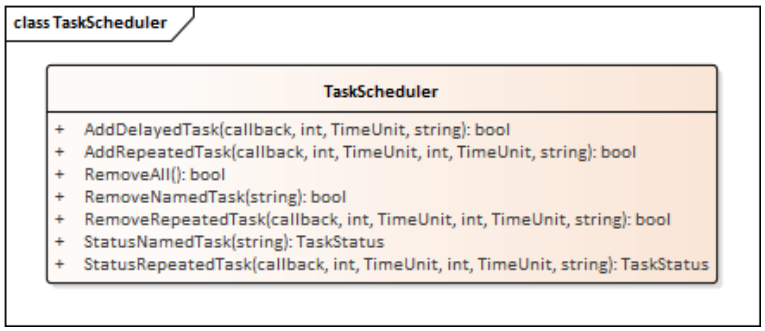
Pro ukládání rozlišuje běžné soubory a dočasné soubory. Dočasné soubory jsou promazávané po stanovené době (záleží na nastavení, výchozí hodnota je sedm dní). A zároveň dočasné soubory nejsou kopírovány při přesunu projektu mezi uzly. Rozhraní pluginu najdete v Obrázku 4.6.



Obrázek 4.6. Rozhraní LocalStorage pluginy

4.5.2 TaskScheduler

TaskScheduler plugin se stará o plánování akcí uvnitř uzlu (akce nicméně může vyvolat komunikaci mimo uzel). Ostatním pluginům poskytuje rozhraní, ve kterém mohou plánovat vlastní akce, dotazovat se na jejich stav, či je odebírat.

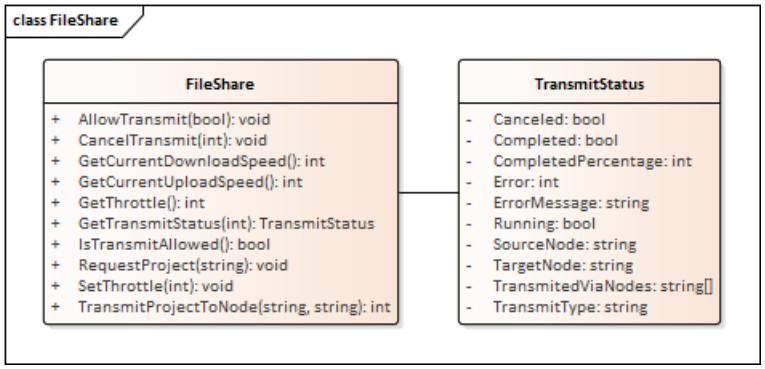


Obrázek 4.7. Rozhraní TaskScheduler pluginy

4.5.3 FileShare

FileShare plugin slouží k posílání souborů projektů mezi jednotlivými uzly sítě. V základu poskytuje rozhraní umožňující vyslat požadavek ke stažení projektu na lokální uzel, příkaz k odeslání projektu na vybraný uzel a sledování stavu nahrávání projektu.

Mimo jiné, plugin umožňuje omezení rychlosti odesílání/stahování projektů, či úplný zákaz. Za pomoci pluginu TaskScheduler tudíž lze nastavit omezení rychlosti nahrávání během dne, kdy je síť využívána i uživateli a neomezenou rychlost během noci, kdy je menší provoz na síti. Rozhraní pluginy je popsáno v Obrázku 4.8.



Obrázek 4.8. Rozhraní FileShare pluginy

4.5.4 Room

Room plugin slučuje lokální audio a video vstupy uzlu do logických celků místností. Toto využijeme v případě kdy uzel ovládá nahrávání z více místností, zároveň umožňuje vydefinovat příkazy, které se mají provést před začátkem a po konci nahrávání z dané místnosti. Bližší nastavení tohoto pluginu probíráme v sekci *5.2 - Konfigurační soubor místnosti.*

4.5.5 KosApi

KosApi plugin implementuje KosApi REST rozhraní, které bylo vytvořeno Ing. Jakubem Jirutkou na fakultě informačních technologií. Toto rozhraní umožňuje přístup k datům aplikace Kos, strukturu aplikace Kos můžete najít v Obrázku 4.9.

Kvůli struktuře Kosu bohužel potřebuje implementovat větší část rozhraní. Pro zisk událostí v místnostech musíme implementovat:

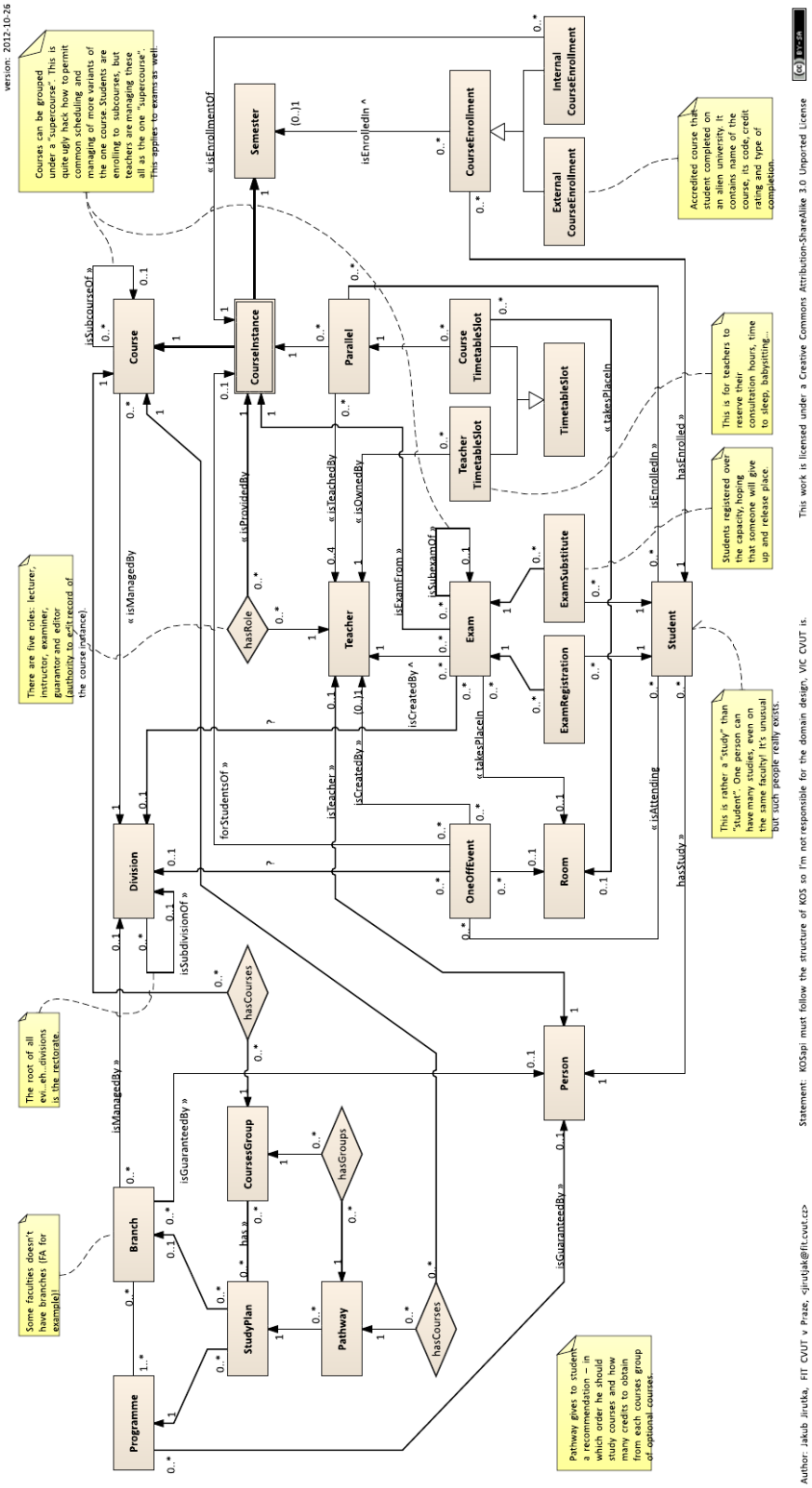
- Předměty - Pro zisk předmětů v semestru
- Instance předmětů - Pro zjištění jestli jsou v daném časovém období (semestru) vyučovány a seznamu učitelů
- Paralelky instance předmětu - Pro zjištění místnosti a časového slotu paralelky
- Zkoušky instance předmětu - Pro zjištění místnosti a časového slotu zkoušky
- Jednorázové události - Pro zjištění místnosti a časového slotu jednorázové události
- Harmonogram - Pro zjištění začátku a konce semestru, svátků a náhradní výuky.

4.5.6 Ffmpeg

Ffmpeg plugin se stará o vytvoření, sledování a ukončení procesu s programem „ffmpeg“. Sledování dosahuje pomocí sledování klasického a chybového výstupu programu.

Plugin v sobě má přednastavené parametry ffmpegu, které vybírá podle nastavení projektu. Například tedy máme přednastavené parametry pro:

- Nalezení NDI zařízení
- Výpis knihoven ffmpegu
- Vytvoření screenshotu z NDI zařízení do jpeg
- Vytvoření screenshotu z běžných zařízení (ffmpeg se sám pokusí určit typ zařízení, toto neplatí v případě NDI) do jpeg



Obrázek 4.9. Struktura KOS (Author Ing. Jakub Jirutka)

- Vytvoření záznamu z NDI zařízení v kvalitě 4k do mkv (Prores)
- Vytvoření záznamu z NDI zařízení v kvalitě FullHD do mkv (Prores)
- Vytvoření záznamu z NDI zařízení v vstupní kvalitě do mkv (Prores)
- Vytvoření záznamu z běžného zařízení v kvalitě 4k do mkv (Prores)
- Vytvoření záznamu z běžného zařízení v kvalitě FullHD do mkv (Prores)
- Vytvoření záznamu z běžného zařízení v vstupní kvalitě do mkv (Prores)
- Vytvoření audio záznamu z NDI zařízení
- Vytvoření audio záznamu z běžného zařízení

V případě že potřebuje zaznamenat více zařízení současně, využíváme více procesů. Toto není jediné řešení, ffmpeg nabízí možnost záznamu více zařízení současně v jednom procesu. Nicméně toto jsme nezvolili kvůli jednoduchosti, možnosti různých startovních a koncových časů a hlavně kvůli tomu že chyba jednoho zařízení by ovlivnila i záznamy ostatních zařízení.

4.5.7 ArtNet

ArtNet plugin implementuje Art-Net protokol, který slouží k přenášení DMX512-A paketů pomocí UDP po síti. DMX512-A se primárně využívá k nastavování světel v místnostech.

Implementace je velice jednoduchá, jelikož protokol neřeší žádné navázání spojení s autorizací a pouze vysílá stateless pakety, viz tabulka 4.1. Tudíž implementačně stačí otevřít UDP spojení, poslat paket a zavřít spojení.

offset (bytes)	0	1	2	3
0	'A'	'r'	't'	'-'
4	'N'	'e'	't'	0x00
8	0x50	0x00	0x00	0x0E
12	Sequence	Physical	Universe	
16	Length		Data	
20	Data			

Tabulka 4.1. Definice Art-Net protokolu

K tabulce je ještě třeba dodat že parametr „Length“ udává délku dat bez hlavičky. Význam „Universe“, „Physical“ je často rozlišný podle zapojení místnosti. Nicméně ve většině případů označuje určitou podmnožinu světel.

Parameter „Sequence“ opět záleží na implementaci, nicméně ve většině případů slouží k vytvoření animace světel za pomocí n packetů kde „Sequence“ udává jejich pořadí, toto je třeba, jelikož packety běhají přes UDP které nezaručuje, že packety k cíly dorazí ve stejném pořadí, jako v kterém byly odeslány.

4.5.8 Visca

Visca plugin implementuje Visca protokol. Stejně jako v případě Art-Net protokolu se jedná o jednoduchý protokol, nicméně na rozdíl od Art-Net protokolu komunikuje po sériové lince, pomocí packetů, viz tabulka 4.2. Parametr „Data“ je specifický podle daného zařízení.

Visca protokol se v nejvíce případech používá pro ovládání kamer. Ovládáním máme na mysli jejich natočení, přiblížení, zaostření a v některých případech i další specifické vlastnosti kamery. Nicméně lze (a často i je) využíván pro ovládání jiných zařízení jako například rolety na oknech.

offset (bytes)	0	1	2	3	4	5	6	7
0	1	Source			Broadcast	Destination		
1..14	Data							
15	0xFF							

Tabulka 4.2. Definice Visca protokolu

Výraz „packet“ může být zde lehce zavádějící, jelikož se nejedná o klasický udp/tcp packet, ale jedná se o vydefinované pořadí a počet bytů posílaných přes sériovou linku po bitech. Jestli je daný bit nula či jedna poznáme podle vzorkování aktuálního napětí, jak můžeme vidět na obrázku 4.10 .

Obrázek 4.10. Komunikace po sériové lince

4.5.9 BlackMagicVideoHub

BlackMagicVideoHub plugin implementuje rozhraní matrix přepínačů vstupních záznamů SmartVideoHub. Je nutno říci, že implementace této pluginy není plně funkční, jelikož poskytnuté zdrojové kódy od společnosti BlackMagicDesign pravděpodobně podporují pouze nová zařízení. Tudíž jsme bohužel nemohli správnost implementace otestovat a prozatím tuto knihovnu nepovažujeme za aktivní v rámci řešení.

Kapitola 5
Konfigurace

V této kapitole probereme konfigurační soubory. Všechny konfigurační soubory programu jsou pro snadnou distribuci mezi uzly serializovatelné do dvou formátů json a yaml. Díky tomu je možné automaticky aktualizovat nastavení celé sítě a nemusí aktualizovat každý uzel zvlášť.

5.1 Konfigurační soubor uzlu

Tento konfigurační soubor (kazeta.yml či kazeta.json) nalezneme ve stejné složce jako samotné řešení. V případě, že bychom potřebovali mít tento soubor umístěný v jiné složce, lze využít parametru „configuration“, viz:

```
./kazeta --configuration {cesta k souboru}
```

Soubor slouží primárně k nastavení unikátního id uzlu, nastavení modulů a knihoven, konfiguraci připojení k síti a další nastavení. V současné fázi projektu obsahuje soubor pouze několik parametrů, ale počítá se s jeho rozšířením v budoucnosti. Ukázku naleznete v Obrázku 5.1.

```yaml
id: Node_1
network:
  user: kazeta
  password: alphabetagama
  virtual_host: kazeta
  hostname: ndi-rabbit.felk.cvut.cz
  port: 5672
modules:
  capture:
    enabled: true
  process:
    enabled: true
  persist:
    enabled: true
  deliver:
    enabled: true
libraries:
  kosapi:
    clientid: 314b28ac-b629-424d-b33b-ee1eec37596e
    clientsecret: dLSsmSehALSSwW5x3DnUwxSz7GsaQFPS
  localstorage:
    - path: ./data
      primary: true
      max_usage: 2048
experimentional:
  net_timeout: 250
commands:
  ndi_scan: true
  ndi_snapshot: true
  ndi_capture: true
  test_network: true
  network_discovery: true
```

Obrázek 5.1. Konfigurační soubor uzlu

5.2 Konfigurační soubor místnosti

Tento konfigurační soubor (room.yml) se stejně jako kazeta.yml či kazeta.json nachází ve stejné složce jako samotné řešení. Opět je zde možnost na něj odkazovat do jiné složky pomocí parametru „rooms“, viz:

```
./kazeta --rooms {cesta k souboru}
```

Soubor pouze obsahuje definice místností uzlu, nikoliv místností celé sítě. Místnosti ostatních uzlů nejsou ukládány a jsou pouze uložené v paměti za běhu programu. Toto řešení jsme primárně zvolili, jelikož nám přijde logičtější se nespoléhat na existenci ostatních uzlů v distribuovaném řešení.

U místností v souboru definujeme akce před, při a po nahrávání, popis podporovaných módů nahrávání a kvality výstupu. A v neposlední řadě k definici vstupů, které má daná místnost k dispozici. Ukázku naleznete v Obrázku 5.2.

```yaml
rooms:
- id: "KN:E-301"
  modes:
    - SPEAKER_PRESENTATION_BLACKBOARD
    - SPEAKER_PRESENTATION
    - SPEAKER_BLACKBOARD
    - SPEAKER
    - BLACKBOARD
    - AUDIO
    - ALL
  initialize:
    - only: [SPEAKER_PRESENTATION_BLACKBOARD,SPEAKER_BLACKBOARD,BLACKBOARD]
      script:
        - artnet turn-off 0x01 0x02 0x03 0x04 0x05
        - artnet turn-on 0x06 0x07
        - artnet intensity .50 0x06 0x07
  capture:
    - only: []
      script:
        - ffmpeg %%
  terminate:
    - only: [SPEAKER_PRESENTATION_BLACKBOARD,SPEAKER_BLACKBOARD,BLACKBOARD]
      script:
        - artnet turn-on 0x01 0x02 0x03 0x04 0x05
        - artnet turn-off 0x06 0x07
  capability: [v480p,v720p,v1080p,v2k,v4k]
  sources:
    - type: NDI
      ip: 10.0.0.0.1
      port: 5040
      id: primary_camera
      display_name: "Speaker camera"
      priority: 1000
      video_capture: true
      audio_capture: false
    - type: NDI
      ip: 10.0.0.0.1
      port: 5041
      id: primary_microphone
      display_name: "Audio in 1"
      priority: 1000
      video_capture: false
      audio_capture: true
    - type: NDI
      ip: 10.0.0.0.1
      port: 5042
      id: test4
      display_name: "Audio in 1"
      priority: 1000
      video_capture: false
      audio_capture: true
```

Obrázek 5.2. Konfigurační soubor místnosti

Konfigurační soubor projektu

5.3 Konfigurační soubor projektu

Konfigurační soubor projektu je odlišný od ostatních, jelikož je většinou tvořen samotným řešením a je průběžně aktualizován během zpracovávání projektu. Soubor slouží k definici celého životního cyklu projektu od nahrávání, přes zpracování a uložení až po distribuci. Ukázku naleznete v Obrázku 5.3.

```yaml
id: test
stage: NOT_STARTED
notify: ["frycjiri@fel.cvut.cz"]
author: frycjiri
files:
  - filename: test.mp4
    size: 1024
    checksum: asdasdasdasdjiojpojpoi
capture:
  room: KN:E-301
  from: 2019-04-10T12:59:36+0000
  to: 2019-04-10T18:59:36+0000
  mode: SPEAKER_PRESENTATION_BLACKBOARD
  quality: v1080p
persist:
  at: [Node_2]
  communications:
    - type: ARQP
      max_bandwith: 128MB
  keep_raw_files: false
  keep_help_files: false
  keep_processed_files: true
process:
  mapped_files:
    - filename: output_1080.mp4
      quality: v1080p
    - filename: output_720.mp4
      quality: v720p
    - filename: output_480.mp4
      quality: v480p
  script:
    - ffmpeg -i %primary_video% -i %primary_audio% -c:a aac -c:v libx264 -strict experimental merged.mp4
    - ffmpeg -i merged.mp4 -filter:v scale=1080:-1 output_1080.mp4
    - ffmpeg -i merged.mp4 -filter:v scale=720:-1 output_720.mp4
    - ffmpeg -i merged.mp4 -filter:v scale=480:-1 output_480.mp4
```

Obrázek 5.3. Konfigurační soubor projektu

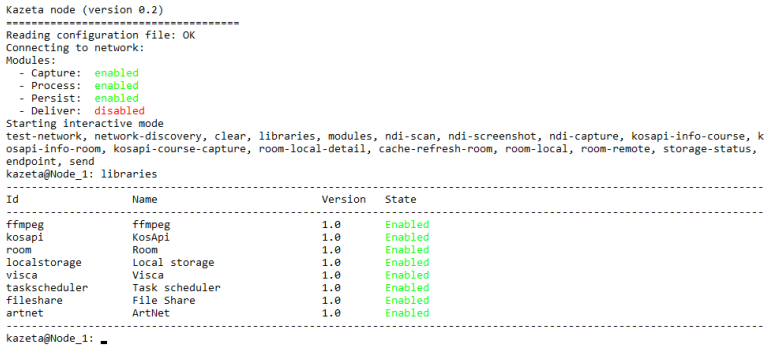
Kapitola 6
CLI

Command language interface, neboli rozhraní příkazové řádky, slouží v současné verzi řešení jako jediné rozhraní pro ovládání. V budoucích verzích bude z velké části nahrazeno webovým rozhraním.

6.1 Implementace

Běžné CLI rozhraní v prostředí .NET Core je pro naše potřeby nevyhovující, například nepodporuje čtení speciálních znaků jako tabulátor. Tudíž jsme se rozhodli implementovat vlastní CLI rozhraní aplikace. Díky tomu můžeme nasimulovat stejné chování na všech platformách a vytvořit funkcionality, které by nám jinak chyběli jako:

- Automatické doplňování názvů příkazů a parametrů po stisknutí tabulátoru
- Historie příkazů při pohybu šipkami nahoru a dolů
- Přepisování příkazů pomocí klávesy backspace
- Copy&Paste z konzole pomocí pravého tlačítka myši
- Obarvení textu a pozadí konzole pro výpisy chybových hlášek a dalších informačních zpráv
- Přepisování již vypsaných řádků pro tvorbu ukazatele načítání



Obrázek 6.1. Ukázka CLI rozhraní

6.2 Příkazy

V tabulkách **6.1 - CLI Příkazy v jádru** a **6.2 - CLI Příkazy v knihovnách** naleznete implementované příkazy. Tyto příkazy jsou využitelné i z konfiguračních souborů místností a projektů, které jsme probírali v předchozí kapitole. Bližší popis příkazů lze získat přímo v programu pomocí:

```
{příkaz} -?
```

Příkaz	Funkce
clear	Vyčistí konzoli
connection-test	Testování připojení do sítě
connection-discovery	Vyhledání ostatních uzlů v síti
exit	Ukončení aplikace
libraries	Vypíše knihovny
modules	Vypíše moduly a jejich stav
nodes-list	Výpis seznamů viditelných uzlů v síti

Tabulka 6.1. Implementované CLI příkazy v jádru

Knihovna	Příkaz	Funkce
artnet	artnet-global	Nastavení globálních proměnných
artnet	artnet-send	Odeslání artnet příkazu
blackmagic	blackmagic-connect	Připojení k blackmagic zařízení
blackmagic	blackmagic-switch	Přepnutí matice
blackmagic	blackmagic-disconnect	Odpojení od blackmagic zařízení
ffmpeg	ffmpeg-capture	Spustí nahrávání přes ffmpeg
ffmpeg	ffmpeg-convert	Spustí převod přes ffmpeg
ffmpeg	ndi-find-sources	Nalezne NDI vstupy
ffmpeg	ndi-take-screenshot	Vygeneruje screenshot z NDI
ffmpeg	ndi-capture	Spustí nahrávání přes ndi
fileshare	share-send-project	Odešle projekt
kosapi	kosapi-room-info	Zobrazí informace k místnosti
kosapi	kosapi-course-info	Zobrazí informace ke kurzu
kosapi	kosapi-course-capture	Naplánuje nahrávání kurzu
localstorage	localstorage-status	Stav lokálního úložiště
room	room-cache-refresh	Obnoví seznam místností ze sítě
room	room-local-info	Zobrazí detailní informace o lokální místnosti
room	room-info	Zobrazí informace o místnosti
taskscheduler	scheduler-info	Zobrazí naplánované úlohy
taskscheduler	scheduler-add	Přidá naplánovanou úlohu
visca	visca-connect	Připojí k visca zařízení
visca	visca-disconnect	Odpojí od visca zařízení
visca	visca-move	Pohne visca zařízením
visca	visca-zoom	Zazoomuje visca zařízení

Tabulka 6.2. Implementované CLI příkazy v knihovnách

6.2.1 Rozhraní

Rozhraní na implementaci příkazů je jednoduché jak lze vidět na ukázce v obrázku 6.2. Pro jejich implementaci stačí rozšířit abstraktní třídu „Command“ a implementovat pole stringů „Aliases“, které určuje zkratky přes které lze příkaz volat v konzoli. A následně metodu „Execute“, která se stará o samotný průběh příkazu.

Hojně se také v rozhraní příkazů využívá naše třída „ConsoleP“ s rozšiřujícími metodami pro výpisy do konzole. Jedná se například o vypisování tabulek, obarveného textu, atd.

```
class CmdKosApiRoomInfo : Command<KosApi>
{
    public CmdKosApiRoomInfo(KosApi library) : base(library)
    {
    }

    public override string[] Aliases => new string[] { "kosapi-info-room" };

    public override void Execute(AbstractLibraryInstance instance, CommandInput input)
    {
        var res = library.Endpoint.RoomEvents(input.Arguments[0]);
        res.Wait();
        var response = res.Result;
        ConsoleP.AppendDelimeter();
        ConsoleP.AppendCols(new int[] { 20, 20, 20, 10 }, "course","from","to", "type");
        foreach(KosRoomEvent ev in response.Events)
        {
            ConsoleP.AppendCols(new int[] { 20, 20, 20, 10 }, ev.Links.Course, ev.StartsAt.ToString("dd/MM/yy HH:mm"), ev.EndsAt.ToString("dd/MM/yy HH:mm"), ev.EventType);
        }
        ConsoleP.AppendDelimeter();
    }
}
```

Obrázek 6.2. Ukázka kódu implementace příkazového rozhraní

Kapitola 7
Testování

V této kapitole se zaměříme na testování aplikace. Vzhledem k tomu, že se jedná o distribuovaný systém, je testování těžší než u centralizovaných systémů.

7.1 Výběr druhu testování

Na testování jsme vybírali z několika druhů[17], bohužel jak se ukázalo ne všechny jsou pro naše řešení použitelné.

Unit testování a z větší části i integrační testování je pro většinu naší implementace nepoužitelné. Většina našeho řešení implementuje propojení s programy typu ffmpeg, lokálními zařízeními a vzdálenou komunikaci. Všechny tyto věci znemožňují unit a integrační testování.

Další možností bylo systémové testování, nicméně je bohužel časově náročné na nastavení a zároveň jsme si nebyli dostatečně jisti našimi znalostmi potřebných technologií.

Došli jsme k rozhodnutí testovat většinu funkcionalit pomocí průchodových scénářů. Použití našeho řešení je přímočaré, tudíž bylo jednoduché vytvořit scénáře, testující jednotlivé komponenty, a tyto následně spojit do jednoho velkého scénáře, testující celý systém.

7.2 Testovací prostředí

Jakožto testovací prostředí využíváme několik stanic. Dva v kompetenci SVTI a dva v naší kompetenci:

- ndi-stream-server.felk.cvut.cz
  - CPU: 8x2.2GHZ
  - RAM: 2GB
  - Jediná stanice s připojenými místnostmi
  - Stanice s moduly capture,process,deliver a persist
- ndi-rabbit.felk.cvut.cz
  - CPU: 1x2.0GHZ
  - RAM: 1GB
  - Stanice s modulem network neboli RabbitMQ software
- server.jirifryc.cz
  - CPU: 2x2.4GHZ
  - RAM: 2GB
  - Stanice s moduly capture,process,deliver a persist
- home.jirifryc.cz
  - CPU: 4x3.5GHZ

- RAM: 32GB
- Domácí stanice, slouží pro testování vzdáleného ovládání

Tato menší síť sloužila jak pro fázi vývoje tak pro fázi testování. Vzhledem k tomu že tvorba „virtuálních místností s virtuálními kamerami“ by byla časově a výpočetně náročná, rozhodli jsme se k využívání první stanice jako primárního zdroje záznamů z reálné učebny.

7.3 Scénář testování sítě

Scénář testuje schopnost uzlu se připojit k síti a získat z ostatních uzlů informace o jejich schopnostech.

7.3.1 Požadavky scénáře

Nastavená síť s uzly:

- Node_1: S aktivním capture módem a místností označenou „KN:E-107“ s alespoň jedním kamerovým a jedním zvukovým vstupem
- Node_Comm: S aktivním network módem
- Node_Storage: S aktivním persist a process módem
- Node_2: Na kterém je přihlášen uživatel

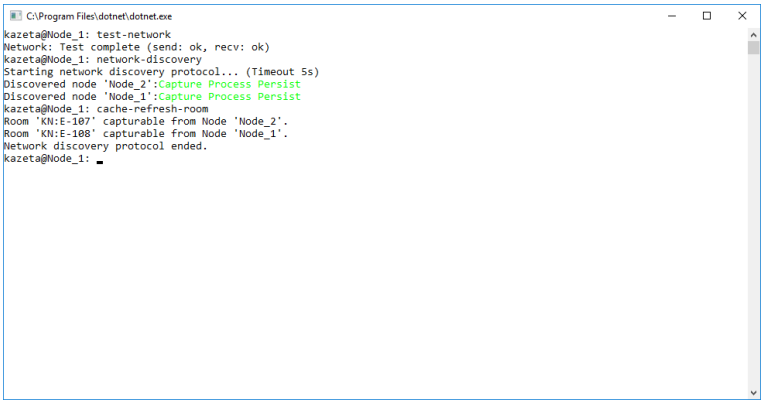
7.3.2 Kroky scénáře

1. Spustíme program v interaktivním módu pomocí „./KazetaNode -i“
2. Počkáme dokud nedojde k plnému načtení programu, plné načtení poznáme podle zobrazení „kazeta@ID_UZLU:“ na posledním řádku.
3. Do konzole zadáme „test-network“ a stiskneme ENTER
4. V konzoli by se měl zobrazit řádek „Network: Test complete (send: ok, recv: ok)“
5. Do konzole zadáme „network-discovery“ a stiskneme ENTER
6. Konzole vypíše „Starting network discovery protocol... (Timeout 5s)“
7. Do konzole zadáme „cache-refresh-rooms“ a stiskneme ENTER

7.3.3 Očekávaný výsledek

Konzole by měla vypsat všechny aktivní uzly naší sítě ve formátu: „Discovered node ’Node_1’:Capture Process Persist“

Konzole po timeout (při běžném nastavení pět sekund) vypíše „Network discovery protocol ended“. Výsledek můžete taktéž vidět na obrázku 7.1.



Obrázek 7.1. CLI - Testování sítě

7.4 Scénář testování integrace KosAPI

Scénář testuje napojení řešení na rektorátní systém KOS a správné čtení informací o místnostech/předmětech.

7.4.1 Požadavky scénáře

Nastavená síť s uzly:

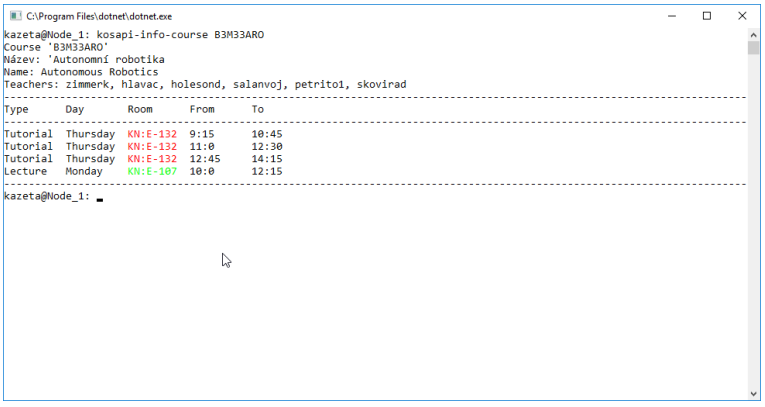
- Node_1: S aktivním capture módem a místností označenou „KN:E-107“ s alespoň jedním kamerovým a jedním zvukovým vstupem
- Node_Comm: S aktivním network módem
- Node_Storage: S aktivním persist a process módem
- Node_2: Na kterém je přihlášen uživatel

7.4.2 Kroky scénáře

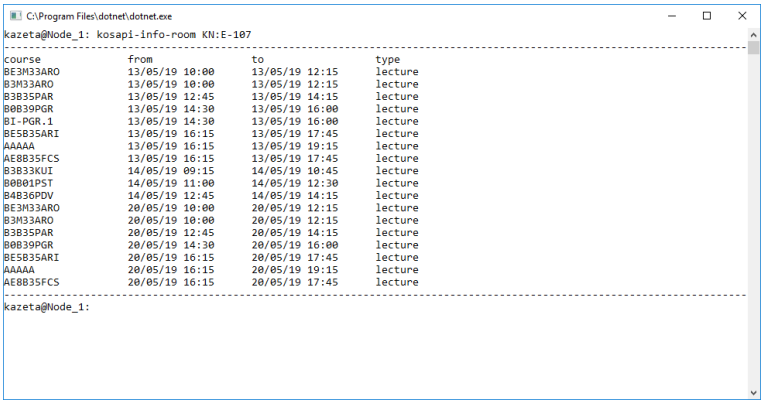
1. Spustíme program v interaktivním módu pomocí „./KazetaNode -i“
2. Počkáme dokud nedojde k plnému načtení programu, plné načtení poznáme podle zobrazení „kazeta@ID_UZLU:“ na posledním řádku.
3. Do konzole zadáme „kosapi-info-course B3M33ARO“ a stiskneme ENTER
4. Do konzole zadáme „kosapi-info-room KN:E-107“ a stiskneme ENTER

7.4.3 Očekávaný výsledek

Konzole by měla vypsat informace o kurzu B3M33ARO a o místnosti KN:E-107 z KOS. Jak by tento výpis měl vypadat můžeme vidět na obrázku 7.2 a obrázku 7.3.



Obrázek 7.2. CLI - Vypsání informace o kurzu z KOS



Obrázek 7.3. CLI - Vypsání informace o místnosti z KOS

7.5 Scénář testování lokální funkcionality

Scénář testuje lokální funkcionalitu a nastavení uzlu.

7.5.1 Požadavky scénáře

Nastavená síť s uzly:

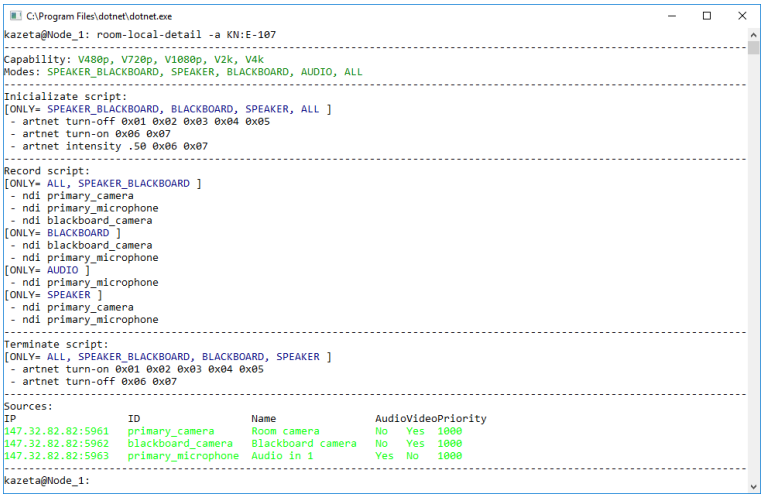
- Node_1: S aktivním capture módem a místností označenou „KN:E-107“ s alespoň jedním kamerovým a jedním zvukovým vstupem
- Node_Comm: S aktivním network módem
- Node_Storage: S aktivním persist a process módem
- Node_2: Na kterém je přihlášen uživatel

7.5.2 Kroky scénáře

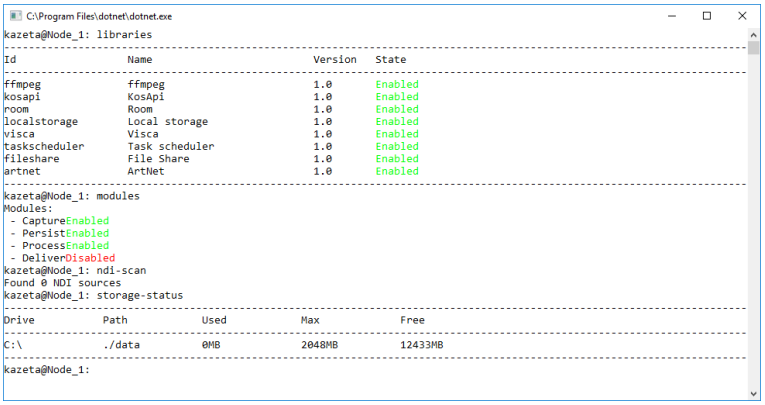
1. Spustíme program v interaktivním módu pomocí „./KazetaNode -i“
2. Počkáme dokud nedojde k plnému načtení programu, plné načtení poznáme podle zobrazení „kazeta@ID_UZLU:“ na posledním řádku.
3. Do konzole zadáme „room-local-detail -a KN:E-107“ a stiskneme ENTER
4. Do konzole zadáme „libraries“ a stiskneme ENTER
5. Do konzole zadáme „modules“ a stiskneme ENTER
6. Do konzole zadáme „ndi-scan“ a stiskneme ENTER
7. Do konzole zadáme „storage-status“ a stiskneme ENTER

7.5.3 Očekávaný výsledek

Konzole by měla vypsat informace o připojených místnostech, načtených knihovnách, atd. Jak by tento výpis měl vypadat můžeme vidět na obrázku 7.4 a obrázku 7.5.



Obrázek 7.4. CLI - Načtení detailu místnosti



Obrázek 7.5. CLI - Lokální nastavení uzlu

7.6 Scénář naplánování záznamu podle údajů z KosAPI

Scénář testuje správné naplánování záznamu dle zadaných údajů. Tento scénář není statický jako ostatní a vyžaduje výběr kurzu který se daný semestr vyučuje v místnosti připojené v uzlu. Tudíž pro jeho správný průběh je třeba takový kurz zvolit.

7.6.1 Požadavky scénáře

Nastavená síť s uzly:

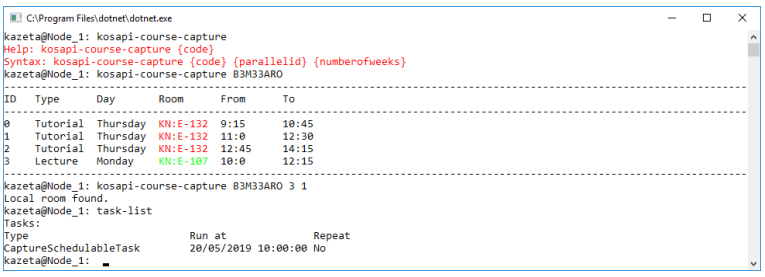
- Node_1: S aktivním capture módem a místností označenou „KN:E-107“ s alespoň jedním kamerovým a jedním zvukovým vstupem
- Node_Comm: S aktivním network módem
- Node_Storage: S aktivním persist a process módem
- Node_2: Na kterém je přihlášen uživatel

7.6.2 Kroky scénáře

1. Spustíme program v interaktivním módu pomocí „./KazetaNode -i“
2. Počkáme dokud nedojde k plnému načtení programu, plné načtení poznáme podle zobrazení „kazeta@ID_UZLU:“ na posledním řádku.
3. Do konzole zadáme „kosapi-course-capture B3M33ARO 3 1“ a stiskneme ENTER
4. Do konzole zadáme „task-list“ a stiskneme ENTER

7.6.3 Očekávaný výsledek

Mělo by dojít k vytvoření naplánovaného úkolu pro nahrávání tak jak můžeme vidět na obrázku 7.6.



Obrázek 7.6. CLI - Naplánování záznamu

7.7 Scénář pro testování celého systému

Tento scénář kombinuje všechny předchozí v pořadí v jakém byli definovány v tomto dokumentu. A následně obsahuje několik dalších kroků.

7.7.1 Požadavky scénáře

Nastavená síť s uzly:

- Node_1: S aktivním capture módem a místností označenou „KN:E-107“ s alespoň jedním kamerovým a jedním zvukovým vstupem
- Node_Comm: S aktivním network módem
- Node_Storage: S aktivním persist a process módem
- Node_2: Na kterém je přihlášen uživatel

A proběhly úspěšně všechny předchozí scénáře.

7.7.2 Kroky scénáře

1. Počkání než proběhne naplánovaný záznam z „scénáře pro naplánování záznamu“ podle údajů z KosAPI.
2. Ověření vytvoření záznamu na uzlu „Node_1“
3. Ověření přesunu na uzel „Node_Storage“
4. Počkání na zpracování do výstupních formátů
5. Ověření vytvoření výstupních formátů na uzlu „Node_Storage“
6. Ověření kvality nahraného videa a zvuku

7.7.3 Očekávaný výsledek

Mělo by dojít k vytvoření všech záznamů a výstupních formátů. Video by mělo odpovídat kvalitě FullHD a 4k záznamů. Toto je subjektivní, nicméně uvažujeme zatím za nekvalitní záznam takový, který je kostičkovaný či jinak porušený.

7.8 Závěr z testování

Testování na pozitivní průběhy proběhly v pořádku. Během testování se objevili jen drobné problémy spojené s plánovanou odstávkou rektorátních systémů. Řešení v tu dobu nedokáže tvořit nové požadavky na záznam. Již zanesené požadavky by nicméně měli proběhnout bez problému. Řešení jinak ale plně zvládá průchod všemi scénáři a považujeme testování za úspěšné.

Kapitola 8
Závěr

V této kapitole zhodnotí celé řešení a nastíníme budoucnost tohoto projektu a s ním spojeným dalším vývojem na fakultě.

8.1 Licencování řešení

Řešení je v současné verzi vydáváno pod licencí MIT, pod kterou počítáme že bude vydáváno i v budoucích verzích. Pro tuto licenci jsme se rozhodli, protože povoluje tvorbu placených pluginů, které by mohly být cestou k financování projektu pro další vývoj.

Zároveň licence MIT nelimituje a nepřikazuje pouze licenci MIT, jako například licence GPL. Ale při tvorbě změn a rozšíření mohou být změny pod jinou licencí. To povoluje implementaci více rozhraní a typů pluginů.

8.2 Možné dopady na výuku

V celé práci jsme neřešili jeden z hlavních dopadů, který plná automatizace a nahrávání přednášek mohou mít. To jest snížení docházky na přednášky. Některé předměty mají již nyní problémy s docházkou studentů na přednášky. Chodí jim například jen deset studentů z osmdesáti. Nahrávání by pro tyto předměty mohlo způsobit že studenti přestanou chodit úplně.

Náš osobní názor nicméně je, že toto nemusí být špatně, může se jednat o transformaci výuky některých předmětů, kde bude větší důraz na interaktivní online přednášky a na cvičení.

8.3 Pokračování práce na projektu

Počítáme s tím že toto není konec naší práce na projektu a že se budeme projektu věnovat dále. Rádi bychom viděli integraci do fakultního Moodle a automatické nahrávání z většiny přednáškových sálů.

Toto nicméně naráží na podstatný problém pro který zatím nemáme řešení. A to je kapacita pro ukládání všech přednášek. Bavíme se zde o desítkách a možná i stovkách terabajtech potřebného místa, navíc ještě s ochranou před ztrátou dat.

Takováto kapacita bohužel na FEL v současnosti chybí a dokonce i fakultní systémy typu Moodle, živoří jen s pár desítkami gigabajtů. Ale pevně věříme, že toto bude řešitelný problém.

8.4 Zhodnocení diplomové práce

V práci jsme úspěšně analyzovali požadavky kladené na systém, současné a budoucí technické vybavení fakulty a dostupné technologie pro záznam, přenos a kódování audio-video záznamů.

Navrhli jsme robustní, rozšiřitelné řešení, které jsme následně implementovali. Toto řešení plně pokrývá požadavky kladené na systém. Vybrali jsme způsob testování vhodný pro takto distribuovaný systém, vytvořili testy a testy úspěšně provedli.

Práce zabrala mnohem více času nežli jsme původně očekávali, nicméně v práci vidíme smysl a nyní již máme na dosah řešení které půjde využívat po celé fakultě a díky univerzálnosti našeho návrhu i mimo ní.

Literatura

[1] David Austerberry. Video format conversion. *Broadcast Engineering (World Edition)*. 2010,

[2] ITU-T. *H.264 : Advanced video coding for generic audiovisual services*. 2017. https://www.itu.int/rec/T-REC-H.264-201704-I.

[3] Madhukar Budagavi Vivienne Sze. *Design and Implementaion of Next Generation Video Coding Systems (H.265/HEVC Tutorial)*. 2014. http://www.rle.mit.edu/eems/wp-content/uploads/2014/06/H.265-HEVC-Tutorial-2014-ISCAS.pdf.

[4] V. Baroncini T. K. Tan, M. Mrak. *Report on HEVC compression performance verification testing*. 2014. http://phenix.it-sudparis.eu/jct/doc_end_user/current_document.php?id=9089.

[5] Google Inc. *VP8 Data Format and Decoding Guide*. 2011. https://datatracker.ietf.org/doc/rfc6386/.

[6] Alex Converse. *VideoLAN Dev Days 2015: New video compression techniques under consideration for VP10*. 2015. https://www.youtube.com/watch?v=gkz1ZvejmEc.

[7] Google. *RTP Payload Format for VP9 Video*. 2018. https://tools.ietf.org/html/draft-ietf-payload-vp9-06.

[8] Jan Ozer. *YouTube Stops H.264 4K Encoding; Will Apple Get on Board?* 2017. https://www.streamingmedia.com/Articles/News/Online-Video-News/YouTube-Stops-H.264-4K-Encoding%3b-Will-Apple-Get-on-Board-115970.aspx.

[9] Apple. *Apple ProRes*. 2018. https://www.apple.com/final-cut-pro/docs/Apple_ProRes_White_Paper.pdf.

[10] Douglas A. Kerr. *Chrominance Subsampling in Digital Images*. 2012. http://dougkerr.net/Pumpkin/articles/Subsampling.pdf.

[11] Jack Haughton Peter de Rivaz. *AV1 Bitstream & Decoding ProcessSpecification*. 2019. https://aomediacodec.github.io/av1-spec/av1-spec.pdf.

[12] Andrey Norkin Joel Sole, Liwei Guo. *Performance comparison of video coding standards: an adaptive streaming perspective*. 2018. https://medium.com/netflix-techblog/performance-comparison-of-video-coding-standards-an-adaptive-streaming- perspective-d45d0183ca95.

[13] Ben Waggoner. *Compression for Great Digital Video*. 2002. https://books.google.cz/books?id=4yXVZYMd-q4C&dq=adb+s-video&source=gbs_navlinks_s.

[14] Hui He Asif Ali Laghari. *Impact of Video File Format on Quality of Experience* . 1 vydání. Kwangwoon University and Springer-Verlag GmbH Germany: 3D

Display Research Center , 2018 . ISBN 2092-6731.
https://doi.org/10.1007/s13319-018-0191-x.

[15] Jessica Alouette. *Should you be recording to MP4 with OBS?■*. 2019.
https://medium.com/@tielqt/should-you-be-recording-to-mp4-with-obs-obs-mythbusters-fc8513851170.

[16] VMware Adam Bloom. *How fast is a Rabbit? Basic RabbitMQ Performance Benchmarks*. 2013.
https://blogs.vmware.com/vfabric/2013/04/how-fast-is-a-rabbit-basic-rabbitmq-performance-benchmarks.html.

[17] Boris Beizer. *Software Testing Techniques* . Edition 2 vydání. Itp - Media , 1990 . ISBN 1850328803.

Příloha A
Slovník

Fakulta	
ČVUT	České vysoké učení technické v Praze
FEL	ČVUT Fakulta elektrotechnická
FIT	ČVUT Fakulta informačních technologií
LMS	Learning management system - Platforma pro výuku a materiály
KOS	Komponenta studenta - Centrální systém pro správu předmětů a studia na ČVUT
Rozlišení	
240p	352 x 240 pixelů
360p	480 x 360 pixelů
480p	858 x 480 pixelů
720p	1280 x 720 pixelů
1080p	1920 x 1080 pixelů
1440p	2560 x 1440 pixelů
4k	3840 x 2160 pixelů
5k	5120 x 2880 pixelů
8k	7680 x 4320 pixelů
Přenos	
S-Video	Zastaralé rozhraní pro přenos záznamu pomocí dvou analogových linek
SDI	Serial digital interface. Sériové digitální rozhraní využívající k přenosu koaxiální kabel.
NDI	Network device interface. Síťové rozhraní zařízení využívající k přenosu klasické ethernetové rozhraní
HDMI	V současnosti nejrozšířenější digitální rozhraní pro přenos záznamu
Síť	
unicast	Komunikace mezi dvěma koncovými zařízeními
multicast	Komunikace mezi zařízeními, které se přihlásili ke stejné skupině
broadcast	Komunikace mezi všemi zařízeními ve stejné síti
UDP	User datagram Protocol - Pro rychlou ale nespolehlivou komunikaci
TCP	Transmission Control Protocol - Pro spolehlivou ale pomalejší komunikaci
Datagram	Krátké zprávy posílané přes síť, obsahují hlavičku a tělo zprávy
AMQP	Protokol pro posílání zpráv mezi zařízeními či aplikacemi
RabbitMQ	Zprostředkovatel zpráv implementující AMQP a jiné protokoly.

channel — Kanál pro posílání zpráv s nastavenými koncovými frontami (queue)

queue — Fronty pro čtení zpráv zařízeními/aplikacemi

RTMP — Real time messaging protocol je tcp protokol pro sdílení videa, zvuku a dat přes internet. Protokol podléhá licencím společnosti Adobe.

HLS — HTTP Live Streaming je protokol pro streamování videa a zvuku pře internet, licenčně podléhá společnosti Apple

WebRTC — WebRTC je projekt umožňující komunikaci mezi dvěma zařízeními ve webovém prohlížeči a byl standardizován do většiny webových prohlížečů

Formáty

mp4 — Formát pro ukládání videa v souborovém systému

mov — Formát pro ukládání videa v souborovém systému

mkv — Formát pro ukládání videa v souborovém systému

flv — Formát pro ukládání videa v souborovém systému

json — Json je jednoduchý a minimalistický formát pro ukládání a přenos dat

yaml — Yaml je formát pro ukládání dat ve formě strukturovaných textových dokumentů

xml — Xml je formát pro přenos a ukládání dat zaměřený na čitelnost dat

Video kodeky

H.264 — Kódovací formát pro videa, primárně využívaný pro rozlišení FullHD a menší, distribuované přes síť

H.265 — Kódovací formát pro videa, primárně využívaný pro záznamy na Blue-ray a DVD

VP8 — Kódovací formát pro videa, primárně využívaný pro krátké animace

VP9 — Kódovací formát pro videa, primárně využívaný pro rozlišení vyšší než FullHD

AV1 — AOMedia Video 1, Kódovací formát pro videa nové generace

ProRes — Kódovací formát pro videa, využívaný primárně v post-produkci

bitrate — Bitrate je počet bitů za sekundu které obsahuje záznam, či které je schopná síť či jiné zařízení zpracovat

chroma subsample — Chroma subsample je metoda pro ztrátovou kompresi videa. Metoda využívá vlastností lidského oka, které je méně citlivé na rozdíly barev oproti světelnosti

Kazeta

uzel — Uzlem se myslí koncové zařízení komunikující pomocí Mycelium protokolu

discovery — Discovery značí metodu či protokol pro dotazování zařízení v síti za účelem nalezení zařízení které splňuje naše požadavky

Post-produkce

ffmpeg — Aplikace pro práci se zvukem, obrazem a videem. Podporuje veškeré známé operace od záznamu, přenosu až po úpravy.

libav — Aplikace pro práci se zvukem, obrazem a videem. Podporuje veškeré známé operace od záznamu, přenosu až po úpravy.

obs	Open Broadcast System je aplikace sloužící k záznamu, kompozici a distribuci záznamu
Ostatní	———
proof-of-concept	Ukázka funkčnosti návrhu/řešení
GDPR	GDPR je obecné nařízení o ochraně osobních údajů

Příloha B
Obsah přiloženého DVD

- **diploma_thesis/** - Tato diplomová práce
  - **source/** - Zdrojové soubory práce ve formátu TEX
  - **thesis.pdf** - Tato diplomová práce
- **modernization/** - Přiložené dokumenty plánu modernizace výukových prostor.
- **source_code/** - Zdrojové soubory
- **readme.txt** - Popis obsahu DVD
- **instalace.txt** - Popis instalace řešení

Příloha C
Plán modernizace výukových prostor na ČVUT FEL

Dokumentaci k plánu modernizace výukových prostor na ČVUT FEL naleznete na DVD ve složce „modernizace“