České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačové grafiky a interakce

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Anastasia Kuznetsova**

Studijní program: Softwarové technologie a management
Obor: Web a multimedia

Název tématu: **Distribuovaný přehrávač videa**

Pokyny pro vypracování:

Na základě svého předchozího projektu navrhněte a implementujte distribuovanou verzi videopřehrávače. Architektura přehrávače bude založená na tzv. uzlech, kde každý uzel je počítač zodpovídající za přehrávání příslušného výřezu zadané části videa. Přehrávač bude umožňovat distribuci videa na více než dva uzly a umožní jejich synchronizaci při přehrávání. V prvním přiblížení uvažujte, že každý uzel má lokálně k dispozici vlastní kopii videa, ale uvažujte i variantu, kdy vstupem uzlu je video-stream ze sítě. Ovládání celého systému bude formou sady příkazů nebo grafického rozhraní z jednoho z uzlů, který zajistí propagaci povelů k ostatním uzlům. Navrhněte metody testování a okomentujte a zdůvodněte výsledné chování aplikace. Implementujte v prostředí jazyka C++.

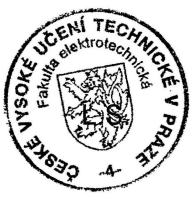
Seznam odborné literatury:

Ajay D. KSHEMKALYANI and Mukesh SINGHAL: Distributed computing: principles, algorithms, and systems. Cambridge: Cambridge university Press, 2008, 1-239. ISBN 978-0-521-87634-6.

Sungwon NAM, Sachin DESHPANDE, Venkatram VISHWANATH, Byungil JEONG, Luc RENAMBOT and Jason LEIGH: Multi-application inter-tile synchronization on ultra-high-resolution display walls. In: Proceedings of the first annual ACM SIGMM conference on Multimedia systems - MMSys´10. New York, USA: ACM Press, 2010. ISBN 9781605589145.

Vedoucí: Ing. Roman Berka, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

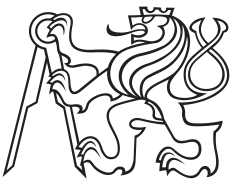


prof. Ing. Jiří Žára, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 25. 3. 2015

Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Distribuovaný přehrávač videa

Anastasia Kuznetsova

Letní semestr, 2015
Vedoucí práce: Ing. Roman Berka, Ph.D.

Prohlášení

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

V Praze dne 20. 5. 2015

.....

Abstrakt / Abstract

Distribuované systémy pro práci s multimediálním obsahem, jejichž hlavní výhodou je možnost rozdělit jednu úlohu mezi několik zdrojů, nachází uplatnění ve vzdělávacích institucích, v reklamě, ve vizuálním umění a mnoha dalších oblastech. Cílem této bakalářské práce je navrhnout model synchronizace obrazu v distribuovaných systémech a následně tento model implementovat jako distribuovaný videopřehrávač.

V úvodu práce je rozebrána problematika synchronizace v distribuovaných systémech a řešení použitá v existujících nástrojích. Na základě získaných poznatků je navrženo několik způsobů synchronizace, jejichž základem je komunikace mezi jednotlivými uzly. V implementační části je popsán návrh a implementace datové, synchronizační, ovládací a prezentační vrstvy přehrávače.

Závěrečná část práce se věnuje testování nástroje v reálném prostředí distribuovaného systému.

Klíčová slova: distribuované systémy, synchronizace multimediálního obsahu, synchronizované přehrávání přes síť, vzdálené přehrávání videí, C++, potvrzovací protokoly.

Distributed systems for working with multimedia content, the main advantage of which is possibility to divide the task into one or more parts, are used in institutes, advertising, visual arts, etc. The aim of this bachelor thesis is to design a model of frame synchronization in the distributed systems and then to implement this model as distributed video player.

The problematic of the synchronization in distributed systems is analyzed first, as well as solutions used in a current settings. The foundation of some synchronization methods is communication by exchanging messages. These methods are designed based on acquired knowledge. Description of designed and implemented data, synchronization, control and presentation layers is described in the implementation part of the thesis.

The final part of the thesis is dedicated to testing of the player in real distributed environment.

Keywords: distributed systems, synchronization of multimedia content, synchronized playback over a network, remote video playback, C++, commit protocols.

Obsah /

Obrázky

Kapitola 1
Úvod

Neustálý pokrok v oblasti multimediálních technologií zvyšuje požadavky na interakci příslušných systémů s člověkem, avšak tyto požadavky často znamenají vyšší nároky na výpočetní výkon těchto systémů. Nemáme-li k dispozici dostatečně výkonný zdroj, lze tento problém vyřešit pomocí propojení několika počítačů, které se navenek budou tvářit jako celek. Programy pro práci s multimediálním obsahem, jejichž hlavní výhodou je možnost rozdělit jednu úlohu mezi několik zdrojů, nachází uplatnění ve vzdělávacích institucích (video display wall), na výstavách, v reklamě (indoorTV), ve vizuálním umění a mnoha dalších oblastech.

Výběr vhodného nástroje úzce souvisí s požadavky na daný systém. Dostupná řešení můžeme rozdělit do dvou skupin: do první skupiny spadají nástroje zaměřené na distribuované zobrazování, jedním z nich je například SAGE, do druhé skupiny řadíme programy primárně určené k přehrávání videa na jednom zdroji, ovšem s možností synchronizace videa mezi několika instancemi programu. Problémem obou skupin je to, že nejsou implicitně přizpůsobené k distribuovanému přehrávání videí. Proto je hlavní myšlenkou této práce přizpůsobit videopřehrávač k práci v distribuovaném prostředí.

V teoretické části (kapitola 2) bude popsána problematika distribuovaných systémů a příslušných omezení. Analýzu problému a existujících řešení provedeme v kapitole 3. Na základě získaných poznatků z předešlé kapitoly bude v kapitole 4 navržena architektura celého nástroje. V kapitole 6 bude popsána implementace dílčích vrstev nástroje pomocí frameworku libyuri. Poté podrobíme distribuovaný videopřehrávač v kapitole 7 testům, které ověří funkčnost a spolehlivost jak celého systému, tak jednotlivých komponent. Výsledky budou diskutovány v poslední kapitole 8.

Kapitola 2
Problematika distribuovaných systémů

Problematika probíraná v této práci užívá několika pojmů, jejichž význam nemusí být jednoznačný a které vyžadují upřesnění.

- **Synchronizace** je stav, kdy více dějů probíhá současně a zároveň koordinovaně. Systémům pracujícím koordinovaně a synchronně říkáme synchronizované.
- **Uzly** jsou jednotlivé prvky systému. Typicky si pod uzly představujeme počítač či jinou řídící jednotku.
- **Distribuované systémy (DS)** jsou takové systémy, ve kterých mezi sebou jednotlivé uzly komunikují pouze pomocí zpráv.
- **Zpoždění (delay)** uvádí, jak dlouho trvá předání dat z jednoho uzlu do druhého.
- **Kolísání (jitter)** je dle [1] odchylka mezi zpožděními jednotlivých paketů.
- **Video display wall** je systém zobrazovacích zařízení, která jsou navzájem propojena za účelem dosažení jednolitého obrazu.

Dle [2] jsou hlavními rysy DS absence společné paměti a globálních fyzických hodin a nedeterministické provádění procesů. Nutno podotknout, že žádný z uzlů nemá úplnou informaci o celkovém stavu systému.

Při návrhu budeme počítat se dvěma způsoby rozložení dat mezi uzly, viz [3]:

- **Lokální rozložení (LR)** je takové rozložení, kdy všechny uzly obsahují stejná data, není je tedy zapotřebí sdílet.
- **Distribuované rozložení (DR)** je takové rozložení, kdy data jsou uložena v jednom až několika uzlech a ostatní uzly si o ně mohou požádat.

Každá z metod distribuce dat má své klady a zápory, například LR zaručuje rychlý přístup k datům v každém uzlu, avšak vyžaduje zajištění kopie na všech zdrojích a klade větší nároky na paměťovou kapacitu, naopak DR neklade důraz na paměťovou kapacitu, ale na propustnost sítě mezi uzly.

Při implementaci systému s LR stačí vyřešit problém synchronizace obrazů. Oproti tomu DR vyžaduje porozumění procesu výměny dat, což mimo jiné zahrnuje řízení provozu paketů v síti, snahu o zkrácení doby zpoždění mezi pakety a použití vyrovnávací paměti pro dočasné ukládaní dat.

2.1 Způsoby identifikace pořadí procesů

Vykonání jednotlivých procesů v daných uzlech ve správném pořadí je v DS těžko dosažitelné, neboť zde neexistuje žádný globální časový mechanismus (viz [2]), přesto existují takové identifikační metody, které tento problém řeší. Následuje výčet identifikačních metod uplatnitelných v DS.

2.1.1 Globální fyzické hodiny

Globální fyzické hodiny jsou synchronizovány s reálným časem. V DS se obtížně implementují, obzvlášť kvůli neomezenému množství uzlů a kvůli nutnosti použití externího časového zdroje.

2.1.2 Globální logické hodiny

Při hlubším prozkoumání problematiky identifikace pořadí procesů zjistíme, že k určení, zda nějaká událost byla nebo bude provedena dříve než jiná, není nutné použití globálních fyzických hodin. Dle [3] „není důležité, aby se procesy shodly na přesném čase, ale aby se shodly na pořadí, v jakém se staly jednotlivé události“. Hovoříme-li tedy o globálních logických hodinách, nemáme na mysli hodiny v pravém slova smyslu, ale metodu číslování procesů, ze které lze určit jejich nezaměnitelné pořadí.

2.2 Metody komunikace v distribuovaných systémech

Docílit synchronizace procesů v každém uzlu DS lze prostřednictvím komunikace mezi nimi. Výměnou zpráv mezi uzly se snažíme docílit stavu, v němž každý z uzlů má informaci o celkovém stavu systému. Jednotlivé metody musí zohlednit vlastnosti komunikačních sítí, jako jsou třeba nespolehlivost a ztrátovost dat. Není-li uvedeno jinak, předpokládáme, že:

- je definován jednoznačný způsob identifikace požadavků či procesů (možné identifikátory jsou popsány v podkapitole 2.1),
- jeden z uzlů má roli koordinátora komunikace (dále jen koordinátor) a obvykle řídí proces globální synchronizace mezi vykonavateli požadavků
- a žádný z uzlů nemá úplnou informaci o stavu systému.

2.2.1 Jednofázová komunikace s jednosměrným potvrzením

Komunikace mezi koordinátorem a vykonavatelem probíhá pouze v jednom směru, tj. od koordinátora k vykonavateli, a je realizována v jedné fázi, kdy koordinátor posílá vykonavateli zprávu s požadavkem PERFORM, který vykonavateli nařizuje přehrát multimediální obsah. Nevýhodou tohoto způsobu komunikace je, že při ztrátě či zpoždění paketu dochází k narušení plynulosti přehrávání. Obecně platí, že jednofázová komunikace není spolehlivá v komplexních systémech.

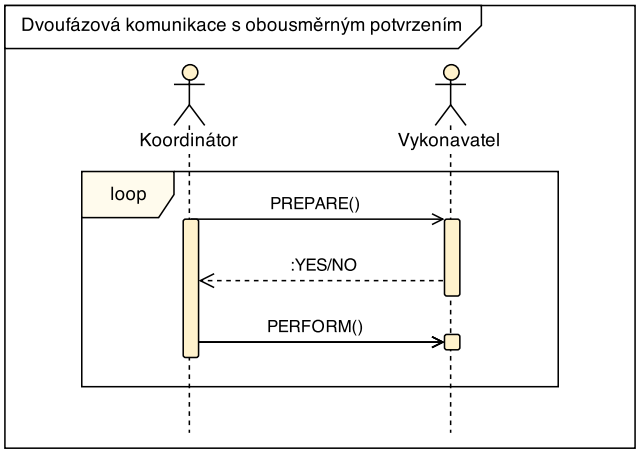
2.2.2 Jednofázová komunikace s obousměrným potvrzením

Na rozdíl od jednofázové komunikace tato metoda místo nespolehlivého jednosměrného potvrzení zavádí obousměrné, tj. od koordinátora k vykonavateli a zpět. Vykonavatelé po obdržení zprávy PERFORM přehrají multimediální obsah a poté odešlou odpověď koordinátorovi (YES/NO), zdali se podařilo tento obsah přehrát. Jednou z výhod této metody je dle [4] možnost vyhodnocení stavu systému na straně koordinátora.

2.2.3 Dvoufázová komunikace s obousměrným potvrzením

Jak uvádí [5] a [6], komunikace mezi koordinátorem a vykonavatelem probíhá v obou směrech, tj. od koordinátora k vykonavateli a zpět. Z obrázku 2.1 vyplývá, že komunikace je realizována ve dvou fázích. Nejprve koordinátor odešle vykonavatelům zprávu PREPARE, která dává vykonavatelům na srozuměnou, že si mají připravit data k přehrání, poté čeká na zprávu od vykonavatelů (YES/NO), zdali jsou požadovaná data dostupná. Neobdrží-li v časovém limitu T odpověď, zruší celou transakci, jinak započne druhou fázi. V té na základě obdržených odpovědí vyhodnotí připravenost celého systému k přehrání dat. Pokud situaci vyhodnotí kladně (READY, tj. od jistého počtu vykonavatelů obdrží zprávu YES), odešle vykonavatelům zprávu PERFORM, která jim nařídí přehrát data, a následně ukončí danou transakci, v opačném případě rovnou ukončí transakci.

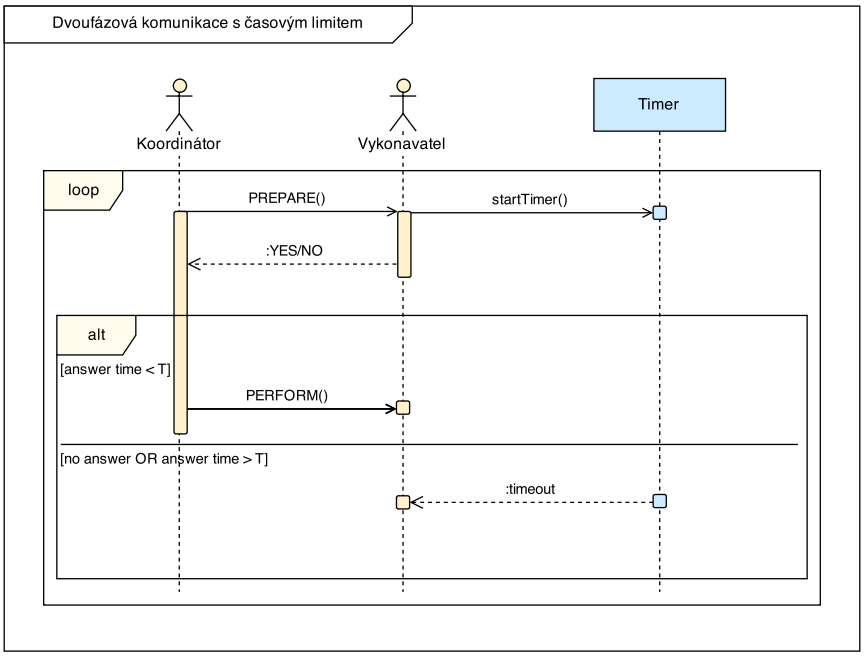
Hlavní nevýhodou dané metody je blokování komunikačního kanálu, ke kterému dojde jak na straně koordinátora při čekání na odpovědi YES/NO od vykonavatelů, tak na straně vykonavatelů při čekání na zprávu PERFORM od koordinátora. Tento problém řeší metody popsané níže.



Obrázek 2.1. Sekvenční diagram popisující průběh dvoufázové komunikace s obousměrnou komunikací. Všechny procesy uvnitř oblasti loop probíhají opakovaně.

2.2.4 Dvoufázová komunikace s časovým limitem

Metoda znázorněná na obrázku 2.2 vylepšuje dvoufázovou komunikaci zavedením časového limitu. První fáze probíhá obdobně jako v dvoustupňové synchronizaci s tím rozdílem, že vykonavatel, jakmile odešle odpověď YES/NO, čeká na zprávu PERFORM jen po určitý čas. Když zpráva PERFORM nedorazí do uplynutí této doby, dojde k automatickému přehrání dat.



Obrázek 2.2. Sekvenční diagram popisující průběh dvoufázové komunikace s časovým limitem. Kombinovaný fragment alt uvnitř diagramu uvádí za jakých podmínek dojde k provedení operace. Všechny procesy uvnitř oblasti loop probíhají opakovaně.

2.2.5 Dvoufázová komunikace s částečným potvrzením

Dvoufázová komunikace s částečným potvrzením předpokládá, že je-li libovolná, dostatečně velká podmnožina vykonavatelů připravená k přehrání dat, pak jsou připraveni všichni. To znamená, že pro vyhodnocení stavu systému jako READY stačí obdržet potvrzení YES/NO od omezeného počtu vykonavatelů. Stanovení tohoto počtu je kritickým parametrem daného algoritmu.

Kapitola 3
Analýza problému

Protože jsme k analýze problému již značně přispěli v předchozí kapitole popisem dostupných algoritmů, budeme se nyní věnovat existujícím nástrojům (přehrávačům) a jejich přístupu k naší problematice.

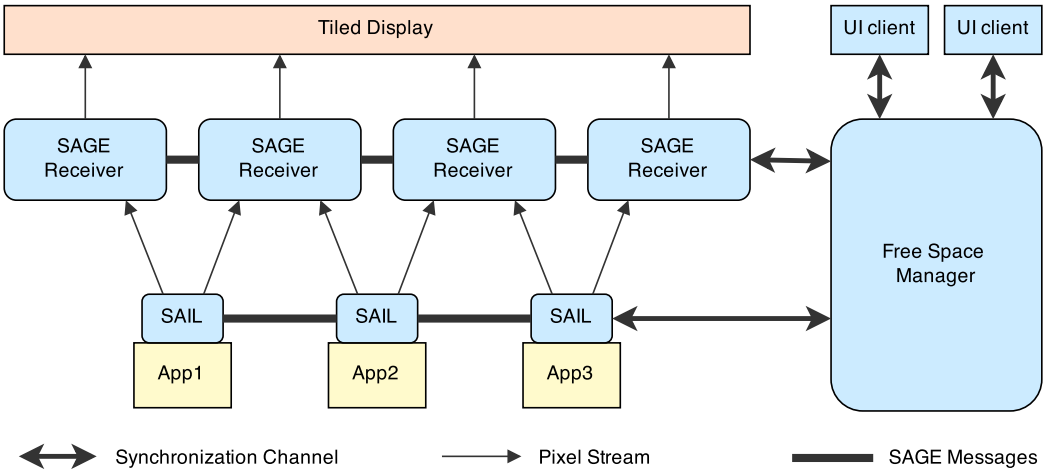
Distribuovány videopřehrávač (DV) je program běžící v prostředí distribuovaného systému (DS). DS se skládá z množiny uzlů, které mezi sebou komunikují pomocí zpráv. Cílem DV je synchronizovat lokální videopřehrávače (LV) v jednotlivých uzlech. Požadavky na DV v sobě zahrnují jednak požadavky na DS jako takové a jednak požadavky na LV. Při použití nástroje daného typu je kladen důraz na vzájemnou synchronizaci jednotlivých uzlů. Ta musí proběhnout v řádu milisekund.

3.1 Popis existujících řešení

Dostupná řešení můžeme rozdělit do dvou skupin: do první skupiny spadají nástroje zaměřené na distribuované zobrazování, jedním z nich je například SAGE. Do druhé skupiny řadíme programy primárně určené k přehrávání videa na jednom zdroji, ovšem s možností synchronizace videa mezi několika instancemi programu.

3.1.1 Scalable Adaptive Graphics Environment (SAGE)

Jak se uvádí v [7], SAGE[1]) je multiplatformní nástroj, který umožňuje přenos a zobrazování velkého objemu multimediálních dat v reálném čase ve video wall systémech z jednoho či více vzdálených zdrojů.



Obrázek 3.1. Architektura systému SAGE. Převzato z [8].

[1]) `http://www.sagecommons.org`

Architektura systému SAGE, jak je vidět na obrázku 3.1, se skládá z:

- **UI klient (User Interface Client)** představuje program s grafickým uživatelským rozhraním, pomocí kterého lze ovládat zobrazení dat na displejích. UI klient zobrazuje informaci o stavu systému a sbírá pokyny od uživatele, např. o změně velikosti okna aplikace či o jeho posunutí.
- Data nasbíraná UI klienty se přeposílají **FS manažerovi (Free Space Manager)**, který, jak je uvedeno v [9], tato data zpracuje (a následně udržuje) a vytvoří příslušné uspořádání jednotlivých programů na displeji. Na základě vytvořeného uspořádání přiřazuje FS manažer data odpovídajícím SAGE přijímačům (displejům) a uzlům s aplikační vrstvou (SAIL).
- **SAGE aplikační vrstva (SAIL)** rozděluje obraz na příslušné části, které odesílá příslušným SAGE přijímačům. SAIL může být jednoduše upravena uživatelem, třeba z důvodu vyhovění určitým požadavkům.
- **SAGE přijímače (SAGE Receivers)** přijímají data z vrstvy SAIL a řídí zobrazování příslušných dat na displejích.
- Pro lepší synchronizaci jsou v SAGE implementovány dva **synchronizační kanály (synchronization channels)**. Jeden na úrovni SAGE přijímačů, který zajišťuje synchronizaci obrazu a je realizován při použití komunikačního algoritmu, popsaného v podkapitole 2.2.3, a druhý na úrovni vrstvy SAIL. Synchronizace ve vrstvě SAIL je zapotřebí, pokud paralelně běžící procesy nemají synchronizovaný výstup. Jinak řečeno, pokud se doby jejich odezev výrazně liší. V takovém případě totiž hrozí ztráta obrazové informace.

3.1.2 VLC media player

Jeden z přehrávačů nejméně závislých na platformě je VLC[1]). Podporuje nejrůznější audio a video formáty (MPEG-1 , MP3 , AVI a další). Jednou z jeho funkcí je synchronizace video stop na uzlech v síti, která je ovšem podmíněna stejnou délkou těchto stop. Implementována je metoda jednofázové komunikace, popsané v podkapitole 2.2.1. Jako identifikátor slouží číslo snímku příslušné video stopy.

3.1.3 MPlayer

MPlayer[2]) je multimediální přehrávač pro Linux. Podporuje velké množství vstupních formátů (MPEG, AVI, VIVO a další). Napříč sítí lze synchronizovat více instancí tohoto přehrávače, což je možné využít pro vytvoření video wall systému [10]. Na rozdíl od VLC může každá instance MPlayeru přehrávat různě dlouhou video stopu. Jednotlivé instance se přitom budou snažit minimalizovat odchylku mezi svou video stopou a video stopou hlavní řídící instance (koordinátora). Komunikace mezi uzly se provádí pomocí metody jednofázové komunikace. Jako globální identifikátor opět slouží číslo snímku příslušné video stopy. Ovládání toku přehrávání se provádí na straně koordinátora, ostatní uzly se mu přizpůsobují.

[1]) `https://www.videolan.org/vlc`
[2]) `http://www.mplayerhq.hu/design7/news.html`

Kapitola 4
Návrh řešení

4.1 Specifikace požadavků

Ještě před samotným návrhem dílčích komponent videopřehrávače je zapotřebí definovat požadavky kladené na výsledný nástroj. Následující výčet požadavků vychází ze zadání práce a nově nabytých poznatků z předchozích dvou kapitol. Funkční požadavky popisují chování, které se od systému očekává. Obecné požadavky zahrnují omezení kladená na systém a specifikují další jeho vlastnosti.

4.1.1 Funkční požadavky

- **Synchronizované videopřehrávače na více než dvou uzlech.** Distribuovaný videopřehrávač umožní synchronizaci videopřehrávačů na více než dvou uzlech.
- **Přehrávání celého videa nebo jen jeho částí.** Každý z uzlů umožní přehrávání jak celého videa, tak i jeho výřezů.
- **Distribuce videa od jednoho uzlu ostatním.** Jeden z uzlů systému umožní distribuci videa zbylým uzlům.
- **Přehrávání v případě lokálního a distribuovaného rozložení dat.** Videopřehrávač umožní distribuované přehrávání jak lokální kopie videa, tak video streamu (viz kapitolu 2).
- **Grafické uživatelské rozhraní.** Jeden z uzlů bude poskytovat uživateli jednoduché uživatelské rozhraní, které umožní snadné ovládání a kontrolu stavu nástroje.
- **Ovládání toku přehrávání.** Nástroj umožní ovládání videopřehrávače na jednom z uzlů. Daný uzel také zajistí propagaci uživatelských příkazů ke zbylým uzlům systému.
- **Podpora resynchronizace.** Uzly se budou moci resynchronizovat. Vypadne-li během přehrávání jeden z uzlů, bude se schopen zotavit a pokračovat v přehrávání.
- **Snadná konfigurace přehrávače pomocí konfiguračních souborů.** Daný způsob konfigurace zaručí větší flexibilitu nástroje a dovolí uživatelům přizpůsobit přehrávač specifickému prostředí.
- **Konfigurace uzlů z příkazové řádky.** Konfigurace z příkazové řádky dovolí uživatelům měnit nastavení nástroje dle svých potřeb, aniž by bylo nutné vytvářet nový konfigurační soubor.

4.1.2 Obecné požadavky

- **Použití frameworku libyuri[1]).** Libyuri je framework poskytující prostředky k vytváření vícevláknových aplikací zpracovávajících audio a video soubory. Podrobněji bude popsán v následující kapitole 5.
- **Kompatibilita videopřehrávače s operačním systémem Linux.** Tento požadavek vychází z použití frameworku libyuri.

[1]) `http://projects.iim.cz/yuri`

- **Orientace na použití v lokální síti.** Nástroje daného typu se běžně užívají v lokálních sítích, např. v systému SAGE.
- **Použití jazyka C++.** Použití jazyka C++ je odvozeno především z požadavku na kompatibilitu s frameworkem libyuri.

4.2 Architektura distribuovaného videopřehrávače

V předchozí kapitole jsme zmínili, že daný videopřehrávač je program běžící v prostředí distribuovaného systému. Tento program se skládá z množiny spolupracujících uzlů. Obecně lze architekturu klasického multimediálního přehrávače rozdělit do několika nezávislých vrstev uspořádaných do úrovní. Nejnižší datová vrstva odpovídá za zpřístupnění dat a jejich konzistenci. Je řízena střední, ovládací vrstvou, která je zodpovědná za ovládání toku přehrávání. Konečně nejvyšší prezentační vrstva představuje grafické uživatelské rozhraní, které má na starosti komunikaci s uživatelem a zobrazení multimediálního obsahu. Z této obecné architektury můžeme odvodit architekturu pro distribuovaný přehrávač. Stačí přizpůsobit každou z vrstev lokálního přehrávače charakteristickým rysům distribuovaného systému, popsaného v kapitole 2, a přidat vrstvu zodpovědnou za synchronizaci mezi jednotlivými uzly.

4.2.1 Datová vrstva

Při návrhu datové vrstvy přehrávače musíme počítat se dvěma způsoby rozložení dat v distribuovaném systému, které již byly stručně popsány v kapitole 2.

Prvním, méně populárním, je lokální rozložení, při němž každý z uzlů disponuje vlastní kopií všech dat. Při použití daného způsobu rozložení dat musí uživatel zaručit, že všechny uzly v síti mají stejná data.

V případě distribuovaného rozložení dat budou uzly schopny přijímat nebo vysílat video stream do sítě. Nutno podotknout, že uzel poskytující data nemusí být zodpovědný za jejich přehrávání. Vysílač video streamu bude moci být umístěn na jakémkoliv zařízení, i bez zobrazovací jednotky.

Požadované chování datové vrstvy bude možné nastavit před spuštěním nástroje pomocí příkazové řádky nebo v konfiguračním souboru.

4.2.2 Synchronizační vrstva

Synchronizační vrstva bude zodpovědná za synchronizaci video stopy mezi jednotlivými uzly. Pro řízení uzlů bude využit centralizovaný model, ve kterém vždy jeden z uzlů hraje roli koordinátora komunikace a obvykle řídí celý proces globální synchronizace. Díky koordinačnímu uzlu s téměř úplnou informací o globálním stavu systému splňuje daný model, i přes svou jednoduchost, důležitý požadavek na rozdělení zodpovědnosti mezi uzly. Ovšem hlavní nevýhodou tohoto přístupu je hrozící vytížení sítě a koordinačního uzlu. Zatímco vytížení koordinátora se dá vyřešit pouze jeho přesunutím na výkonnější stroj, problém zahlcení sítě se budeme snažit odstranit omezením počtu odesílaných paket.

Pro implementaci centralizovaného modelu řízení uzlů a zaručení větší spolehlivosti nástroje musí mít každý z uzlů unikátní jméno. K označení se použije náhodně vygenerovaný identifikátor, který se přiřadí každému uzlu ihned po spuštění systému.

Chceme-li zaručit synchronizované přehrávání videí v distribuovaném systému, musíme navrhnout všem uzlům společný způsob identifikace video snímků. Jak již bylo zmíněno v kapitole 2, globální fyzické hodiny v distribuovaných systémech neexistují

a jejich případná implementace by byla náročná. Při hlubší analýze problému ale zjistíme, že podmínkou pro synchronizaci jednotlivých uzlů není existence fyzických hodin, nýbrž stanovení globální metody číslování procesů. Pro tyto účely navrhneme metodu globálních logických hodin. Lokální hodiny budou reprezentovány proměnnou, do které se bude zapisovat číslo posledního video snímku, resp. počet již zpracovaných snímků. Různé způsoby reprezentace jsou zapříčiněny zejména různorodostí video dat, např. u některých videí chybí číslování snímků nebo je poškozené. V takovém případě bude vhodné použít čítač, který video snímky očísluje. Hlavní nevýhodou čítače video snímků je absence detekce prohození video snímků, ke kterému může dojít v případě distribuovaného rozložení dat. Z tohoto důvodu bude v případě streamování dat vhodnější použít implicitní číslování snímků, které nám poslouží jako způsob identifikace jednotlivých snímků (budeme tak schopni detekovat prohození snímků), zároveň je tak nutné použít takové video, ve kterém je toto číslování již zahrnuté. Lokálními hodinami budou disponovat všechny uzly. Zvolení způsobu reprezentace hodin proběhne jak na straně koordinátora, tak na straně vykonavatelů.

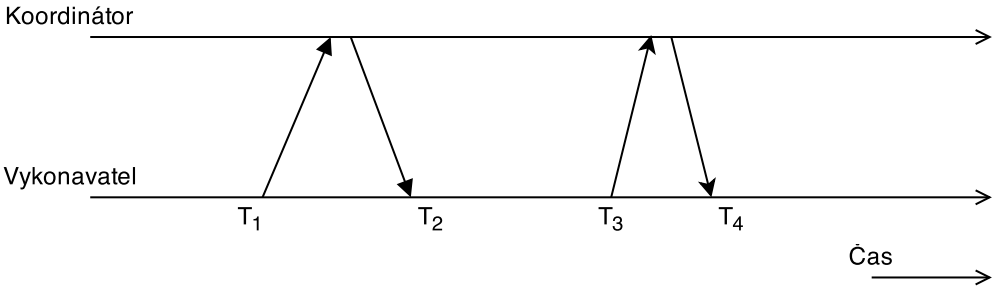
V rámci synchronizační vrstvy budou implementovány tři způsoby komunikace mezi koordinátorem a vykonavateli. Implementace několika metod zaručí větší flexibilitu výsledného nástroje a dovolí uživateli přizpůsobit videopřehrávač vlastním potřebám. Před výběrem komunikačních metod si shrneme typické problémy distribuovaných systémů, které bude nutné vyřešit v každém ze způsobů komunikace:

- **Ztráta jednoho z paketů.** Přestože provoz v lokální síti je zpravidla bezproblémový, dochází i v ní ke ztrátě datových paketů. Každá z vybraných metod komunikace musí být schopna si s výpadkem paketu poradit.
- **Proměnlivá doba zpoždění paketů.** Doba zpoždění paketu v síti není konstantní a závisí na vzdálenosti mezi odesílatelem a příjemcem.
- **Různý stupeň výkonnosti uzlů.** Rozdíl mezi výkony jednotlivých uzlů může způsobit zpomalení procesu synchronizace. Při návrhu způsobu komunikace musíme mít na paměti reakci ostatních uzlů na pomalejší stroj.
- **Výpadek uzlů.** Vzhledem k tomu, že při provozu v distribuovaném systému může dojít k částečnému nebo úplnému výpadku některého z uzlů, budeme při návrhu nástroje počítat s proměnlivým počtem konečných vykonavatelů.
- **Zotavení uzlů po výpadku.** V případě částečného výpadku musí být uzel schopen navázat spojení s koordinátorem a vyrovnat své zpoždění vůči ostatním uzlům.

4.2.2.1 Jednofázové metody komunikace

Ze dvou nejčastěji používaných metod jednofázové komunikace (viz podkapitolu 2.2) bude implementována metoda s jednosměrným potvrzováním. Tato metoda je jednodušší a nezahlcuje síť, na rozdíl od druhé metody, ve které je počet odeslaných zpráv skoro stejně velký jako u spolehlivějších dvoufázových metod.

V dané metodě bude koordinátor vysílat pakety s aktuální hodnotou logických hodin, dokud se jednotlivé uzly nezesynchronizují s koordinátorem a nezobrazí příslušný video snímek. Vzhledem k tomu, že vykonavatelé v dané metodě hrají roli pasivních posluchačů a nemají přehled o stavu systému, není možné při použití daného algoritmu zaručit synchronizaci hodin ve všech uzlech. Hlavním problémem tedy bude různý stupeň výkonnosti uzlů (jeden z uzlů může zpracovávat předchozí pakety déle než druhý) a také proměnlivá doba zpoždění paketů v síti. Vyřešit daný problém lze použitím Cristianova algoritmu. Daný algoritmus nám umožní stanovit přibližné zpoždění paketů při průchodu sítí za předpokladu, že doba přeposílání paketů tam a zpátky se neliší, viz [2] a [11].



Obrázek 4.1. Cristianův algoritmus. T_1 a T_3 znázorňuje čas odeslání paketů. T_2 a T_4 čas obdržení odpovědi. Doba odezvy T_{trans} se pak vypočítá jako $T_{trans} = (T_2 - T_1)/2$ nebo jako $T_{trans} = (T_4 - T_3)/2$. Převzato z [2].

Vykonavatel bude čas od času posílat koordinátorovi testovací paket, který mu koordinátor pošle zpět ihned po jeho obdržení. Doba trvání odesílání paketu tam a zpátky bude změřena a z ní se vypočte doba odezvy (viz obrázek 4.1). Danou hodnotu převedenou do mikrosekund pak vykonavatel použije pro odhad svého zpoždění ve video snímcích. Výpočet zpoždění d_{frame} ve video snímcích se provede jako $d_{frame} = T_{trans} * f/1000$, kde f je snímková frekvence (uvádí počet zobrazených video snímků za vteřinu). Pro upřesnění odhadu a redukování chyb ovlivněných metodikou měření bude možné výsledné zpoždění vypočítat na základě již naměřených hodnot. Způsob výpočtu si uživatel bude moci vybrat ze tři metod. První metoda nebude provádět zprůměrování hodnoty, druhá metoda použije nejčastěji se vyskytující zpoždění a konečně třetí metoda vynásobí počet výskytů jednotlivých zpoždění s jejich velikostí ve snímcích a obdrženou hodnotu vydělí jejich celkovým počtem.

4.2.2.2 Dvoufázové metody komunikace

Dvoufázové metody komunikace budou ve výsledném nástroji reprezentovány dvěma zástupci, s částečným potvrzením a s časovým limitem (viz podkapitolu 2.2). Dané metody jsou vylepšením běžné dvoufázové synchronizace (viz podkapitolu 2.2.3), na rozdíl od které disponují mechanismem odstranění uvíznutí.

Metoda s časovým limitem bude vycházet z předpokladu, že k synchronizaci všech uzlů musí dojít za maximální dobu T, kde $T = 1000/f$, kde f je snímková frekvence. Časový limit bude obsahovat každý uzel. Po uplynutí časového limitu dojde k zobrazení video snímku v každém uzlu. Pro použití dané metody je nutné uvést snímkovou frekvenci daného videa. Pokud snímkovou frekvenci nelze určit přímo z videa, nelze danou metodu použít.

Metoda s částečným potvrzením není na rozdíl od právě zmíněné metody závislá na znalosti snímkové frekvence, ale na počtu kladných potvrzení od vykonavatelů transakce. Abychom zabránili uvíznutí, ke kterému může dojít při ztrátě paketu, musíme počet očekávaných potvrzení nastavit na hodnotu nižší, než je počet konečných uzlů. Stanovení očekávaného počtu potvrzení rozdělíme na dva případy:

- **Počet vykonavatelů je pevně stanoven (nebude vyšší).** V daném případě předpokládáme, že počet vykonavatelů nebude vyšší než počet definovaný uživatelem, ale může být nižší. To jinými slovy znamená, že očekávaný počet nutný k potvrzení transakce lze vypočítat z celkového počtu uzlů.
- **Počet vykonavatelů se může měnit, uzly můžou přibývat.** Tato varianta je o něco složitější než varianta první, neboť číslo vykonavatelů není stanoveno. Z toho vyplývá,

že počet očekávaných potvrzení se bude měnit v závislosti na množství aktuálně připojených uzlů.

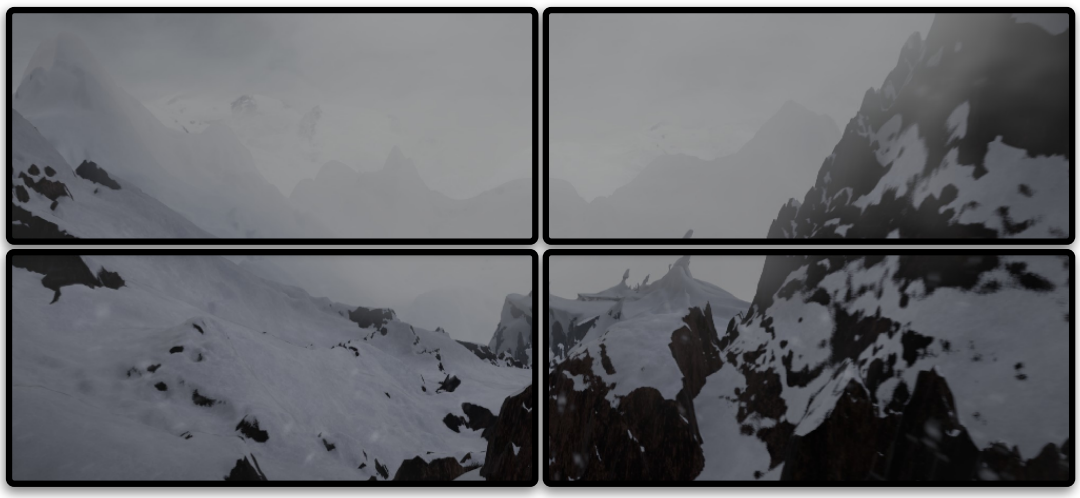
Další vlastností této metody je reakce na záporná potvrzení, ke kterým může dojít, pokud video snímek nebyl připraven k zobrazení. Abychom tento problém vyřešili, přidáme parametr `strict`. Nastavíme-li daný parametr, dojde k zobrazení pouze v případě, kdy bude obdržen očekávaný počet potvrzení a zároveň žádné nebude záporné.

Při implementaci obou metod použijeme paradigma automata-based programming (programování založené na automatech), při jehož použití program nebo jeho část je interpretován jako model formálního stroje (v našem případě konečného automatu). Použití daného programovacího paradigmatu zlepší čitelnost, udržitelnost a znovupoužitelnost našeho kódu.

4.2.3 Ovládací a prezentační vrstva

Jak již bylo zmíněno v požadavcích výše, ovládání přehrávače bude možné pouze na jednom stroji, který zajistí propagaci instrukcí k ostatním uzlům. Z poznatků získaných v podkapitole 4.2.2 víme, že k řízení uzlů bude použit centralizovaný model. Z toho lze odvodit, že ovládání bude možné pouze na straně koordinátora. Vykonavatelé pak budou mít pouze vrstvu prezentační a budou závislé na příkazech od koordinátora.

Prezentační vrstva na vykonavatelích bude představena jedním až několika zobrazovacími okny. V každém okně bude možné zobrazit jak celé video, tak i jeho výřez. Možnost zobrazení výřezu videí bude uplatněna ve video wall systémech, kde výsledný obraz je zobrazen na velkoplošném zařízení tvořeném několika displeji, viz obrázek 4.2. Nastavení plochy k výřezu bude probíhat přes příkazovou řádku nebo prostřednictvím konfiguračních souborů.



Obrázek 4.2. Ukázka možného rozdělení video snímků v potenciálním video wall systému se čtyřmi displeji.

Grafické uživatelské rozhraní (GUI) bude v případě koordinátora představovat webové rozhraní s náhledem na video a základními ovládacími prvky (spustit, zastavit, přetočit). Webové rozhraní bude poskytováno webovým serverem na straně koordinátora. Dány přístup je nejvhodnější zejména proto, že ovládání nástroje bude možné ze všech zařízení v rámci dané sítě. Zároveň dovolí použití přehrávače v systémech, kde jeden z uzlů je vzdálený a ovládání z něj by bylo komplikované. Uživatelské rozhraní bude mít responzivní design, což znamená, že bude schopné se přizpůsobit zařízením s různým rozlišením.

Kapitola 5
Použité knihovny a frameworky

5.1 Framework libyuri

Framework libyuri vyvíjený v Institutu intermédií na Fakultě elektrotechnické ČVUT v Praze poskytuje prostředky pro vytváření vícevláknových nástrojů v jazyce C++ pro práci s multimediálními daty. Podle [12], nástroje využívající frameworku libyuri vytváří orientovaný graf, ve kterém jsou uzly (pro přehlednost jim budeme říkat třídy) odpovědné za jednotlivé fáze zpracování propojené hranami, které přenáší datové rámce.

Propojení těchto tříd se většinou provádí prostřednictvím konfiguračních XML souborů. Průběh spuštění aplikace pak vypadá následovně:

```
1  Načíst data z konfiguračního souboru
2  IF Příkazová řádka obsahuje argumenty THEN
3      Zpracovat argumenty z příkazové řádky
4  ENDIF
5  Definovat vytvoření třídy
6  Vytvořit hrany grafu
7  Propojit všechny třídy prostřednictvím hran
8  Spustit třídy a hrany v samostatném vlákně
```

Nejdůležitější část frameworku tvoří jádro, které obsahuje komponenty potřebné pro vytváření a správu tříd. Dále budou popsány ty části frameworku, které budou použity při implementací našeho nástroje.

- **Základní datové typy pro práci s multimediálními soubory.** Podle [12] datové rámce představují základní jednotku dat, kterou si prostřednictvím hran mezi sebou vyměňují třídy.
- **Jádro systému.** Obsahuje komponenty potřebné pro řízení a správu tříd a třídy pro vytváření a konfiguraci grafu i jeho komponent prostřednictvím XML souborů a příkazové řádky.
- **Třídy.** Třída je součástí grafu a v závislosti na její implementaci může mít několik vstupních a výstupních hran. Všechny procesy prováděné v rámci dané třídy běží v samostatném vlákně, které je za ni zodpovědné.
- **Moduly.** Moduly se mohou skládat z několika tříd, obvykle poskytují jednu konkrétní funkcionalitu.
- **Hrany.** Propojují jednotlivé třídy a řídí přenos datových rámců mezi nimi.
- **Události.** Možnost posílání událostí mezi třídy zaručuje slabou provázanost mezi moduly systému. Ke specifikaci jednotlivých událostí dochází v konfiguračních souborech. Třídy schopné generovat události musí být předkem třídy `BasicEventProducer` a přijímače těchto událostí musí být předkem třídy `BasicEventConsumer`.

5.2 Knihovna ultragrid

Knihovna Ultragrid[1]) je zaměřena na usnadnění procesu vysílání multimediálních dat ve vysokém rozlišení. Jak uvádí [13], knihovna umožňuje vysílání video streamu s nízkým stupněm komprimace nebo úplně bez ní a zaručuje nízký stupeň latence. Po přidání modulu `ultragrid` do frameworku libyuri je možné zacházet s libyuri jako s frameworkem podporujícím proces streamování dát do sítě.

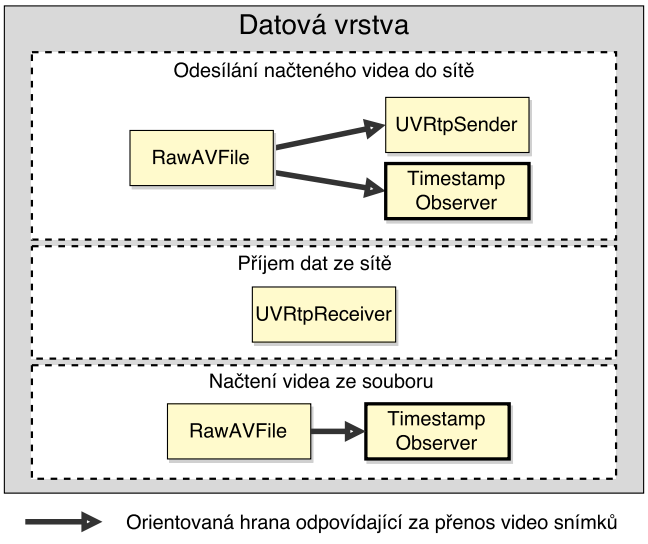
[1]) `http://www.ultragrid.cz/en`

Kapitola 6
Implementace

Distribuovaný videopřehrávač byl napsán v jazyce C++ s využitím jeho nejnovějšího standardu C++11. Při návrhu videopřehrávače jsem se snažila dodržet modulární architekturu frameworku libyuri. Třídy poskytující jednu funkcionalitu jsou shrnuté do příslušných modulů, jejichž implementace je popsána níže. Komunikace mezi třídami probíhá prostřednictvím mechanismu událostí popsaného v podkapitole 5.1. Použití daného přístupu mi umožnilo zachovat jednoúčelovost každé třídy, např. součástí synchronizačních modulů není implementace transportního protokolu odpovídajícího za posílání dat mezi uzly, neboť daný problém řeší jiný modul. Propojení vrstev a specifikace směrů přeposílání událostí bude podrobně popsáno v podkapitole 6.5. Každá vrstva obsahuje několik zaměnitelných komponent, které poskytují různou funkcionalitu. Každá komponenta je tvořena určitým propojením tříd. Nyní se zaměřím na popis implementace každé z vrstev popsaných v kapitole 4.

6.1 Implementace datové vrstvy

Datová vrstva je zodpovědná za načtení, dekódování a distribuci video dat. Komponenty, ze kterých se skládá, jsou zobrazeny na obrázku 6.1.



Obrázek 6.1. Propojení tříd v datové vrstvě za předpokladu distribuovaného a lokálního rozložení dat. Každá z komponent (na obrázku jsou znázorněny čárkovanou čárou) zastupuje určité propojení tříd (na obrázku jsou znázorněny plnou čárou) a je zodpovědná za jednu funkcionalitu. V tučném rámečku jsou znázorněny třídy přidané do libyuri.

6.1.1 Poskytování dat

V případě LR je za poskytování dat zodpovědná třída `RawAVFile`, která vytváří datové rámce a přidává do nich nezbytné informace (např. dobu trvání jednoho snímku, pokud

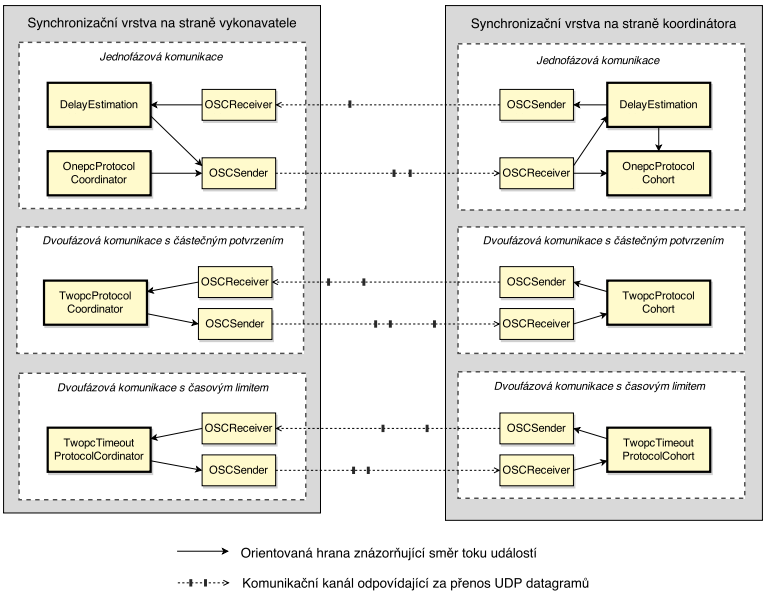
je zapnuté dodržování snímkové frekvence). Pokud jeden z uzlů disponuje lokálními daty a zároveň vysílá data do sítě (DR), jedna z výstupních hran z `RawAVFile` vede do třídy `UVRtpSender`. Ta datové rámce zabalí do paketu a odešle příslušnému uzlu. Za příjem dat ze sítě je zodpovědná třída `UVRtpReceiver`.

6.1.2 Dodržování snímkové frekvence

Dodržování snímkové frekvence má implicitně na starosti třída `RawAVFile`, která vkládá datové rámce do výstupní hrany se zpožděním rovným době zobrazení jednoho snímku. V daných podmínkách by se doba, za kterou bylo dané video přetočeno, rovnala době přetočení ve videopřehrávači. Tento problém řeší třída `TimestampObserver`, pomocí které je zodpovědnost za dodržování snímkové frekvence přesunuta o vrstvu výše. `TimestampObserver` dědí ze třídy `BasicEventConsumer`, což znamená, že přijímá události od ostatních tříd. Ve smyčce dochází ke kontrole fronty událostí, pokud je dodržování snímkové frekvence zapnuté, dochází ke zpoždění snímků, jinak je obdržený snímek rovnou odeslán do výstupní hrany. Při použití daného modulu je nutné ve třídě `RawAVFile` zakázat nastavení časových značek a dodržování snímkové frekvence.

6.2 Implementace synchronizační vrstvy

V rámci synchronizační vrstvy byly implementovány tři metody komunikace mezi uzly distribuovaného systému (viz podkapitolu 4.2.2). Každá z metod bude ve výsledném grafu představovat jednotlivou komponentu, jenž bude zodpovědná za synchronizaci přehrávání. Každá komponenta se bude skládat z několika tříd, například komponenta odpovídající za implementaci jednofázové metody komunikace na straně koordinátora bude tvořena třídami `OnepcProtocolCoordinator`, `DelayEstimation`, `OSCSender` a `OSCReceiver`. Komponenty reprezentující dvoufázové metody komunikace budou tvořené třídou zastupující dvoufázovou komunikaci a třídami `OSCSender` a `OSCReceiver`.



Obrázek 6.2. Propojení tříd v synchronizační vrstvě. Každá z komponent (na obrázku jsou znázorněny čárkovanou čárou) zastupuje určité propojení tříd (na obrázku jsou znázorněny plnou čárou) a je zodpovědná za jednu funkcionalitu. V tučném rámečku jsou znázorněny třídy přidané do libyuri.

Jak je vidět na obrázku 6.2, odesílání dat do sítě budou mít na starosti třídy `OSCSender` a `OSCReceiver`. Dané třídy umožňují přenos dat pomocí TCP nebo UDP protokolů. Data, která mají být odeslána přes síť, třídy obdrží prostřednictvím mechanizmu události, tzn. že pro odesílání dat do sítě stačí definovat dané třídy a specifikovat události, které budou přijímat od ostatních uzlů. V případě synchronizační vrstvy se bude odesílání zpráv provádět prostřednictvím UDP protokolu, pakety se budou odesílat pouze skupině uzlů (multicast).

Dále jsou popsány třídy a moduly, které byly přidány do frameworku libyuri.

6.2.1 Cristianův algoritmus

Modul `delay_estimation` obsahuje implementaci Cristianova algoritmu (viz obrázek 4.1). Chování třídy `DelayEstimation` závisí na postavení uzlů v systému. Vykonavatel periodicky odesílá koordinátorovi zprávu `connection_test` a pomocí stopek si zaznamenává čas (v mikrosekundách), kdy došlo k odeslání. Při úspěšném obdržení odpovědi je změřená doba vydělena dvěma (potřebujeme zjistit dobu trvání odesílání paketu v jednom směru) a převedena do milisekund. Odhadovaná doba zpoždění v síti je pak pomocí metody `emit_event` odeslána do fronty událostí, odkud ji mohou získat ostatní uzly.

6.2.2 Jednofázová metoda komunikace

Modul `onepc_protocol` poskytuje implementaci metody jednofázové komunikace popsané v podkapitole 2.2.1 a skládá se ze dvou tříd. Třída `OnepcProtocolCoordinator` představuje koordinátora komunikace mezi vykonavateli, které zastupuje třída `OnepcProtocolCohort`. Obě třídy jsou součástí nezávislých komponent (viz podkapitolu 6.2), které navzájem komunikují prostřednictvím zasílání událostí. Základní funkcí každé třídy je metoda `run`, která se spouští po spuštění instance dané třídy. Průběh metody `run` na straně koordinátora pak vypadá následovně:

```
1  WHILE Vlákno neobdrželo příkaz o ukončení
2     Vyzvednout video snímek ze vstupní hrany
3     IF Použít čítač snímků THEN
4        Vytvořit zprávu PERFORM s aktuální hodnotou čítače
5     ELSE
6        Vytvořit zprávu PERFORM s implicitním číslem aktuálního snímku
7     ENDIF
8     Odeslat zprávu do fronty událostí
9     Vložit snímek do výstupní hrany
10 ENDWHILE
```

Obdobně probíhá i průběh algoritmů na straně vykonavatelů transakce:

```
1  WHILE Vlákno neobdrželo příkaz o ukončení
2     Vyzvednout video snímek ze vstupní hrany
3     IF Počet snímků za vteřinu nebyl nastaven THEN
4        Vypočítat počet snímků za vteřinu z hodnot získaných ze snímku
5     ENDIF
6     Obdržet požadavek PERFORM s číslem snímku od koordinátora
7     IF Obdržená událost DELAY THEN
8        Vypočítat a zaznamenat zpoždění ve snímcích
9        Zvýšit číslo snímku obdržené od koordinátora o zpoždění
10    ENDIF
11    IF Číslo aktuálního snímku < Obdržené číslo video snímku THEN
```

```
12          Vyrovnat čísla video snímku
13       ENDIF
14       Vložit snímek do výstupní hrany
15    ENDWHILE
```

V krocích 1 až 5 probíhá vyzvednutí snímků ze vstupní hrany a nastavení aktuálního čísla snímku dle čítače, nebo, při použití implicitního číslování, rovnou ze snímku. Následuje výpočet snímkové frekvence, pokud tato nebyla nastavena. Hodnota snímkové frekvence je zapotřebí při výpočtu doby zpoždění ve snímcích (krok 8). Krok 6 reprezentuje obdržení události od koordinátora. Dále je zkontrolována fronta událostí, zdali neobsahuje zprávu typu DELAY. Je-li událost obdržena, dojde k výpočtu zpoždění ve snímcích a jeho zaznamenání. Výsledné zpoždění se vypočte z naměřených dat dle způsobu výpočtu průměrné hodnoty. Dále dojde k navýšení čísla snímku obdrženého od koordinátora o vypočítanou hodnotu zpoždění. V krocích 11 až 13 bude vyrovnáno zpoždění ve snímcích. Na konci bude aktuální snímek vložen do výstupní hrany.

6.2.3 Deterministický konečný automat

Do jádra frameworku byla přidána generická abstraktní třída `StateTransitionTable`, která implementuje chování deterministického konečného automatu. Přechodová tabulka v dané třídě je reprezentována asociativním neuspořádaným kontejnerem. Hodnoty uložené v kontejneru představují dvojici klíč-hodnota. Klíč se vypočítá z hodnoty aktuálního stavu a vstupní události. Hodnotou je objekt typu `Transition`, jehož parametry jsou konečný stav a odkaz na přechodovou funkci. Daný způsob implementace zaručí deterministické chování automatu, neboť použití asociativního kontejneru zaručuje jednoznačnost klíčů, což znamená, že přechod z jednoho stavu do druhého je jasně určen.

Každý potomek třídy `StateTransitionTable` musí definovat vlastní přechodovou tabulku pomocí metody `define_transition_table`. Také každá třída může přetížit funkci `do_default_action`, k jejímuž vykonání dojde, pokud požadovaný přechod nebude nalezen v tabulce.

6.2.4 Implementace dvoufázových metod komunikace

Modul `twopc_protocol` poskytuje dvě metody komunikace. Implementace každé z metod je rozdělena do dvou tříd, první třída reprezentuje koordinátora komunikace, druhá vykonavatele. Jak již bylo zmíněno v návrhové části v podkapitole 4.2.3, na třídy se pohlíží jako na konečný automat, jehož chování je odvozeno od aktuálního stavu a vstupní události. Konečný automat se dá implementovat mnoha způsoby, v daném případě byla vybrána implementace pomocí přechodové tabulky realizované v `StateTransitionTable` (viz předchozí podkapitolu). Každá třída modulu `twopc_protocol` je potomkem `StateTransitionTable`, a proto musí inicializovat vlastní přechodovou tabulku, do které je převedena logika algoritmu. Průběh operací po spuštění instance třídy pak vypadá následovně:

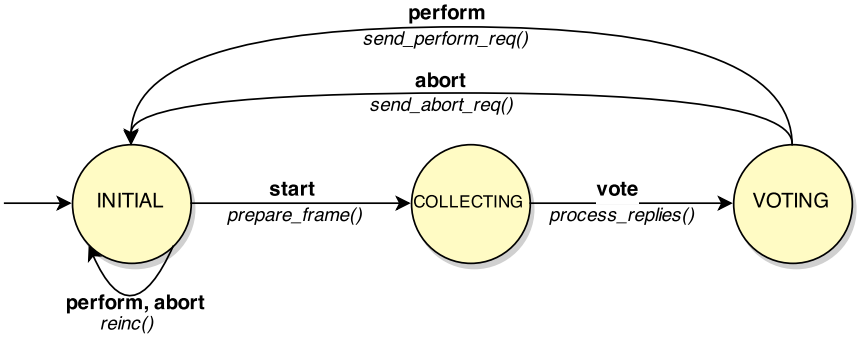
```
1  Nastavit parametry
2  Nastavit hodnoty počátečního stavu a události
3  Inicializovat přechodovou tabulku
4  WHILE Vlákno neobdrželo příkaz o ukončení
5     Najít přechod definovány aktuálním stavem a událostí
6     Vykonat přechodovou funkcí
7     Změnit aktuální stav na hodnotu koncového stavu
8  ENDWHILE
```

V prvních dvou krocích dochází k nastavení hodnot parametrů získaných od uživatele a k inicializaci přechodové tabulky. Dále následuje cyklus, který je vykonáván, dokud aktuální vlákno neobdrží požadavek o ukončení. V krocích 4 až 6 dochází k přechodu z aktuálního stavu do koncového. Při přechodu je zavolána přechodová funkce (krok 5), která na základě vnějších podnětů (obdržených zpráv, získaných snímků) mění stav aktuální události. Tato přednastavená událost je použita při dalším průchodu cyklem.

Dvoufázovou komunikaci s částečným potvrzením reprezentují tyto dvě třídy: `TwopcProtocolCoordinator` a `TwopcProtocolCohort`. Jak již bylo zmíněno výše, každá třída definuje vlastní přechodovou tabulku popisující pravidla přechodů mezi stavy.

Průběh komunikace na straně koordinátora je graficky znázorněn na obrázku 6.3. Počáteční stav INITIAL, v němž se automat nachází, pokud je připraven k vykonání další fáze, vykoná při přechodu do stavu COLLECTING proceduru `prepare_frame`, která načte video snímek ze vstupní hrany a odešle zprávu PREPARE. Při přechodu ze stavu COLLECTING do VOTING bude zavolána procedura `process_replies`, která zpracuje požadavky od vykonavatelů a na jejich základě změní stav aktuální události. Událost bude změněna na PERFORM, pokud počet očekávaných potvrzení bude vyšší nebo roven počtu obdržených potvrzení, a pokud je nastavena proměnná `strict`, bude splněna podmínka striktního zobrazení popsána v podkapitole 4.2.3.

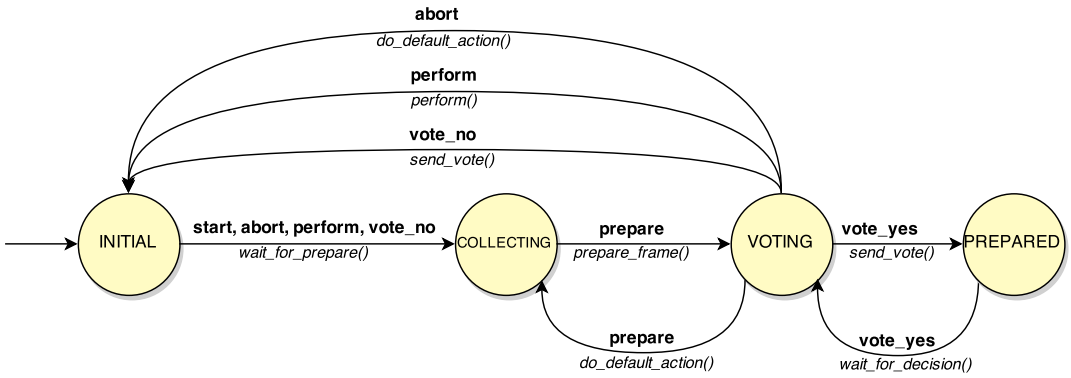
Z následujícího stavu VOTING při obdržení události PERFORM bude zavolána procedura `send_perform_req`, která odešle událost PERFORM do fronty událostí a video snímek do výstupní hrany. V případě událostí ABORT dojde k odeslání paketu se zprávou ABORT a k zahození video snímku. Ve stavu INITIAL dojde k přenastavení hodnot a kontrole množiny vykonavatelů, pokud je jejich počet proměnlivý. Vykonavatelé, od kterých v poslední době nebylo obdrženo žádné potvrzení, budou vymazány. Na konci dojde ke změně události na hodnotu START.



Obrázek 6.3. Stavový automat na straně koordinátora definující stavy a přechody mezi nimi při dvoufázové komunikaci s částečným potvrzením.

Na straně vykonavatele (viz obrázek 6.4) při přechodu z počátečního stavu INITIAL do COLLECTING je zavolána procedura `wait_for_prepare` čekající na obdržení požadavku PREPARE od koordinátora. Po obdržení požadavku dochází k provedení procedury `prepare_frame` zodpovědné za přípravu video snímku. Na základě události, ke které v proceduře dojde, se odvíjí další kroky. V prvním případě nastane událost VOTE_NO, jež znamená, že snímek nebyl připraven a má být zahozen. Nastane-li tato událost, dojde k přechodu do stavu INITIAL, odkud bude odesláno negativní potvrzení koordinátorovi a fáze bude zahájena od začátku. Pokud k vyzvednutí snímku dojde, odešle vykonavatel zprávu YES a přejde do stavu PREPARED. Ve stavu PREPARED bude čekat na rozhodnutí od koordinátora. Po obdržení rozhodnutí dojde k zavolání

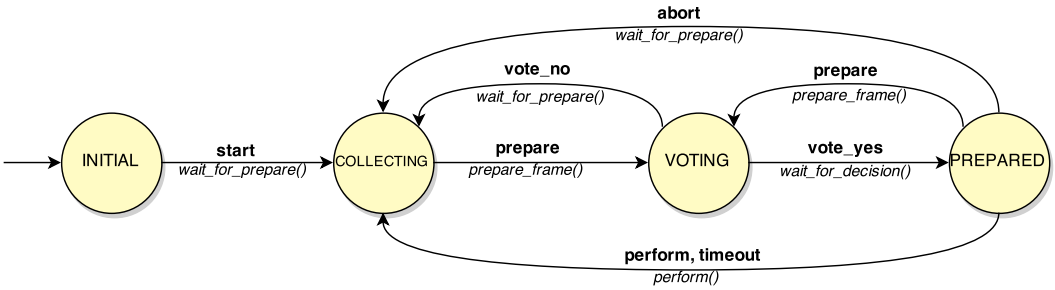
procedury `wait_for_decision` a přechodu do stavu VOTING, v němž na základě obdržených událostí dojde buď k zobrazení, nebo k zahození snímku. Tady je nutné poznamenat, že pokud v průběhu procedury `wait_for_decision` nedojde k obdržení události PREPARE, vykonavatel zahodí aktuální snímek a přijde do stavu COLLECTING. Daný bod zaručuje, že na straně vykonavatele nedojde k uvíznutí.



Obrázek 6.4. Stavový automat na straně vykonavatele definující stavy a přechody mezi nimi při dvoufázové komunikaci s částečným potvrzením.

Třídy `TwopcTimeoutProtocolCoordinator` a `TwopcTimeoutProtocolCohort` reprezentují dvoufázovou komunikaci s časovým limitem. Jak již bylo zmíněno v podkapitole 4.2.3, tato metoda je závislá na hodnotě snímkové frekvence. Nastavení snímkové frekvence může být provedeno pomocí konfiguračního souboru, pokud tomu tak není, bude vypočítána z informací získaných z video snímku (pokud jsou k dispozici).

Stavový diagram na straně koordinátora vypadá obdobně jako v předchozím případě (viz obrázek 6.3). Posloupnost vykonání je stejná, až na to, že v metodě `process_replies` dochází k vyhodnocování stavů na základě uplynulé doby a potvrzení od jiných uzlů. K poslání zprávy ABORT dojde pouze v případě, pokud vykonavatelé nejsou připraveni. Na rozdíl od předchozí metody rozhodnutí o zobrazení video snímku po uplynutí doby může udělat každý z uzlů.



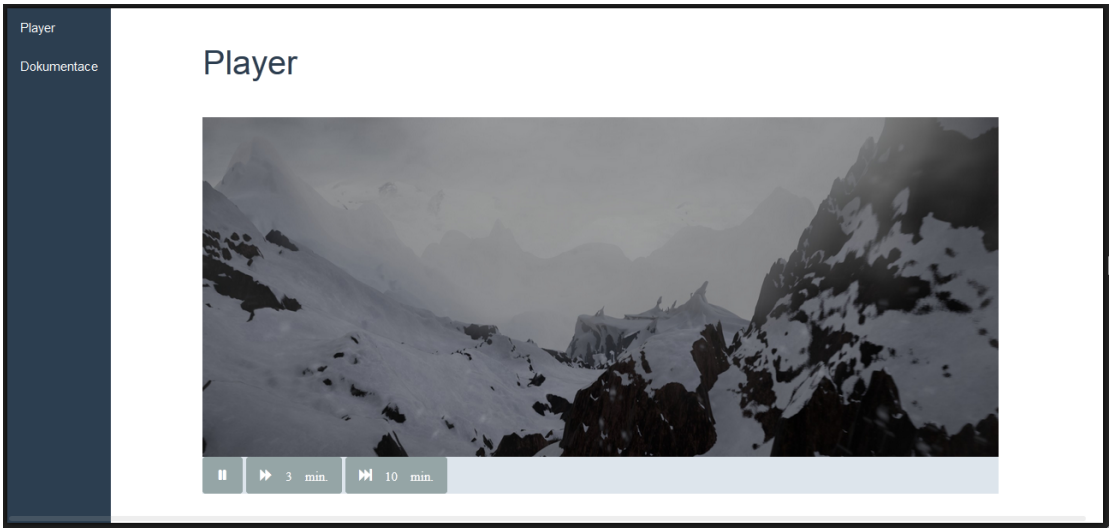
Obrázek 6.5. Stavový automat na straně vykonavatele definující stavy a přechody mezi nimi při dvoufázové komunikaci s časovým limitem.

V případě vykonavatele, obrázek 6.5, k přechodu z počátečního stavu INITIAL do COLLECTING dojde pouze po obdržení zprávy PREPARE. V následujícím kroku se v proceduře `prepare_frame` zaznamená doba počátku fáze a spustí se cyklus, který se pokusí načíst snímek ze vstupní hrany v daném časovém limitu. Po ukončení procedury bude událost nastavena buď na hodnotu VOTE_NO, což znamená zahájení fáze od začátku zavoláním procedury `wait_for_prepare`, nebo na VOTE_YES, v případě

podařeného vyzvednutí požadovaného snímku koordinátorem. Při přechodu do stavu PREPARED vykonavatel čeká na rozhodnutí od koordinátora. V příštím přechodu dojde k zobrazení snímku pouze při obdržení kladného rozhodnutí (PERFORM) nebo uplynutí časové doby (TIMEOUT), v jiných případech bude snímek vyhozen a fáze zahájena od začátku.

6.3 Implementace ovládací vrstvy

Ovládací vrstva je reprezentována webovým rozhraním, k jehož vytvoření byl použit modul `webserver`. Nejdůležitější roli hraje třída `WebServer`, jenž odpovídá za zprovoznění webového serveru na straně koordinátora a přeposílání požadavků od uživatele dalším třídám. V případě žádosti o stránku je daný požadavek předán třídě `WebDirectoryResource`, která vrátí požadovanou stránku, pokud bude nalezena v kořenovém adresáři. Kořenový adresář obsahuje soubory potřebné pro vytvoření webového rozhraní implementovaného v jazyce HTML s využitím javascriptové knihovny jQuery[1]) a frameworku Bootstrap[2]).



Obrázek 6.6. Webové rozhraní k ovládání videopřehrávače.

Jak je vidět na obrázku 6.6, na hlavní stránce se nachází náhled na přehrávané video a ovládací lišta. Náhled se tvoří z průběžně se měnících obrázků, které generuje třída `WebImageResource`. Ovládací lišta obsahuje jednoduché ovládání. Při stisknutí tlačítka je na server odeslán GET požadavek, který obsahuje název události a hodnotu. Třída `WebControlResource` danou hodnotu zpracuje a zařadí do fronty událostí, kde si ji následně vyzvedne `PlaybackController`. Druhá stránka obsahuje krátký popis synchronizačních metod a jejich vstupní parametry. Vytvořené rozhraní je responzivní, což znamená, že se přizpůsobí zařízením s různým rozlišením.

6.3.1 Třída PlaybackController

Hlavní myšlenka třídy `PlaybackController`, odpovídající za řízení toku přehrávání, je jednoduchá a odvozena z mechanizmů předávání datových rámců mezi třídami. Jak jsme již zmínili výše, datové rámce si třídy předávají pomocí orientovaných hran, což ve výsledku znamená, že jsou na sobě závislé. Daný poznatek je uplatněn ve třídě

[1]) `https://jquery.com/`
[2]) `http://getbootstrap.com/`

`PlaybackController`, která je na základě příchozích událostí schopna zablokovat předávání datových rámců nebo, v případě posouvání, zahodit přebytečné rámce.

```
WHILE Vlákno neobdrželo příkaz o ukončení
   Zkontrolovat frontu událostí
   IF Obdržená událost PAUSE THEN
      Zastavit přehrávání
   ENDIF
   IF Obdržená událost MOVE THEN
      Spočítat posun ve video snímcích
      Odeslat událost s žádostí o zákaz dodržování snímkové frekvence
      WHILE Číslo aktuálního snímku se nerovná očekávanému
         Vyzvednout video snímek ze vstupní hrany
         Vložit snímek do výstupní hrany
      ENDWHILE
      Odeslat událost s žádostí o dodržování snímkové frekvence
   ELSE
      Vložit snímek do výstupní hrany
   ENDIF
ENDWHILE
```

Komponenta obsahující danou třídu musí být umístěna pouze na straně koordinátora mezi datovou a synchronizační vrstvou (zaručí zastavení procesu synchronizace). Umístění pouze na straně koordinátora je nutné kvůli architektuře synchronizační vrstvy, neboť přehrávání na straně vykonavatelů se zastaví, jakmile dojde k zastavení komunikace s koordinátorem.

6.4 Implementace prezentační vrstvy

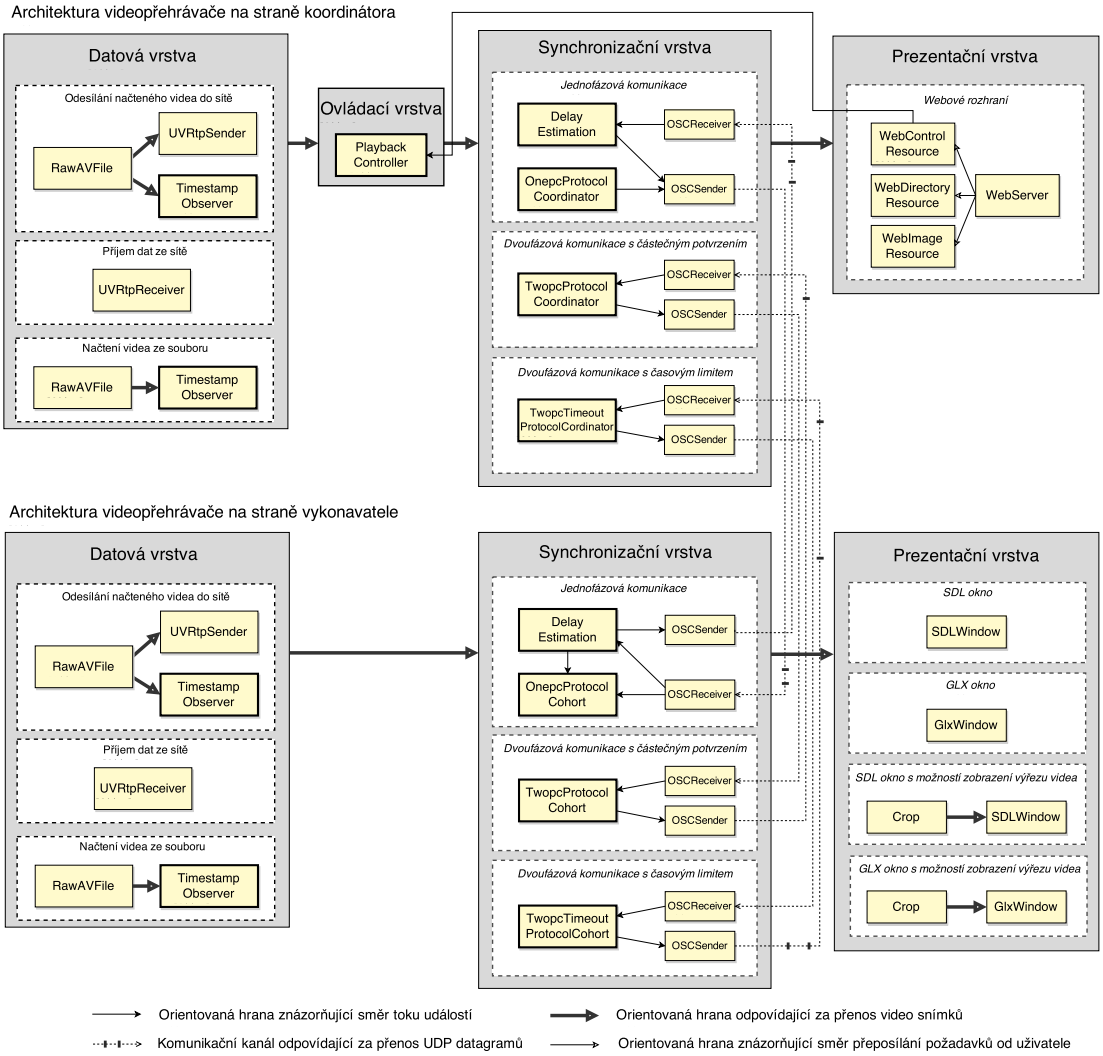
Prezentační vrstva na straně koordinátora představuje webové rozhraní popsané výše. Na straně vykonavatele je pak reprezentována SDL nebo GLX zobrazovacími okny dle volby uživatele. Každé okno může, ale nemusí, zobrazovat část videa. K těmto účelům byla použita třída `Crop`, která ořízne obraz dle parametrů obdržených od uživatele. Okna obsahují minimální ovládací prvky jako zavřít, minimalizovat a maximalizovat.

Jeden z uzlů také může mít několik zobrazovacích zařízení. Kvůli tomu byla implementována podpora zobrazení ve více oknech v rámci jednoho nástroje.

6.5 Propojení vrstev videopřehrávače

Jak již bylo zmíněno v podkapitole 5.1, vytvoření aplikace využívající moduly z libyuri se většinou provádí prostřednictvím třídy `XmlBuilder`, která na základě konfiguračního XML souboru vytváří orientovaný graf. Konfigurační soubor tak musí definovat propojení mezi jednotlivými vrstvami videopřehrávače. Každá vrstva obsahuje několik zaměnitelných komponent, které poskytují různou funkcionalitu. Komponenty v rámci jedné vrstvy jsou snadno zaměnitelné a výběr příslušné komponenty je možné provést před spuštěním nástroje pomocí příkazové řádky.

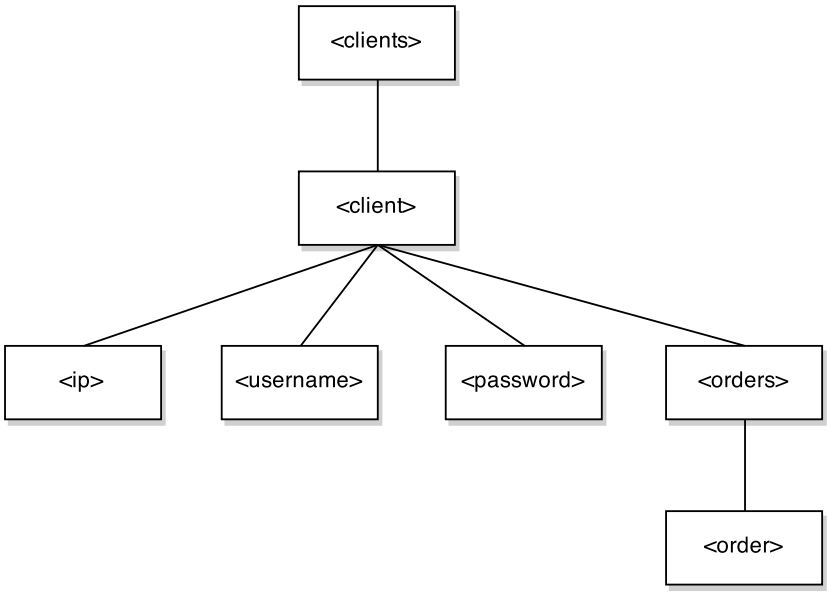
Jednotlivé komponenty byly definovány v různých souborech a jejich připojení probíhá až po spuštění nástroje. Výsledné propojení vrstev a komponent v nich obsažených jsou znázorněné na obrázku 6.7.



Obrázek 6.7. Obecné schéma znázorňující propojení datové, synchronizační, ovládácí a prezentační vrstvy (na obrázku jsou vyplněny šedou barvou) na straně koordinátora a vykonavatele. Každá z komponent (na obrázku jsou znázorněny čárkovanou čárou) zastupuje určité propojení tříd (na obrázku jsou znázorněny plnou čárou) a je zodpovědná za jednu funkcionalitu. V tučném rámečku jsou znázorněny třídy přidané do libyuri.

6.6 Implementace vzdáleného spuštění

V jazyce Python byl napsán skript, který zjednoduší spuštění nástroje ve všech uzlech. Skript po spuštění načte speciální XML soubor, v němž jsou uvedena data potřebná pro vzdálené připojení ke všem uzlům systému. Typy definovaných elementů v XML souboru a jejich pořadí je znázorněno na obrázku 6.8.



Obrázek 6.8. Pořadí a typy elementů obsažených v XML souboru.

Při načtení hodnot ke každému elementu typu client z XML souboru je vytvořeno vlákno, které obsahuje data potřebná pro spuštění videopřehrávače na vzdáleném stroji. Dále dochází ke spuštění vláken, která se pomocí modulu Pexpect[1]), jenž využívá SSH, připojují ke vzdálenému uzlu a začínají v něm spouštět příkazy uvedené v elementu typu order. Po spuštění všech příkazů čeká každé z vláken na požadavek k ukončení, který získá od hlavního vlákna. K ukončení hlavního vlákna a odesílání požadavků dojde po stisknutí `Ctrl+C` nebo při zadání příkazu `exit` do terminálu.

Ze svého adresáře se skript spouští takto:

```
player.py konfiguracni_soubor.xml [--resol_x --resol_y]
```

Pokud při spuštění skriptu bude uvedeno horizontální (`resol_x`) a vertikální (`resol_y`) rozlišení, dojde k výpočtu zobrazované plochy na daných uzlech dle jejich pořadí. Vypočítaná hodnota se doplní místo složených závorek (`geometry={ }`), které uvedeme v příkazu ke spuštění.

Poslední uzel uvedený v konfiguračním souboru vždy představuje koordinátora.

[1]) `https://pexpect.readthedocs.org/en/latest/`

Kapitola 7
Testování

7.1 Metodiky testování

Jelikož hlavním cílem nástroje je synchronizace fyzického výstupu, nebude snadné otestovat jeho celkovou funkčnost. Průběh testování tedy provedeme v několika krocích, které nám umožní ověřit správné fungování jednotlivých komponent videopřehrávače.

7.1.1 Unit testy

Testování pomocí tzv. unit testů spočívá v ověřování funkcionality každé z dílčích jednotek nástroje dle předem stanovených pravidel a je nezbytnou součástí vývoje aplikace. Jednotkou v našem případě chápeme část modulu, třídu nebo funkci se složitější logikou, jejichž funkčnost lze ověřit nezávisle na jiných jednotkách.

7.1.2 Integrační a systémové testování

Zatímco integrační testování ověřuje spolupráci jednotlivých komponent, systémové testování se zabývá funkčností nástroje jako takového. Oba způsoby jsou zaměřené na vyhodnocení výstupu testované jednotky. V našem případě ale není vyhodnocení takového výstupu jednoduché, neboť, jak jsme již zmínili výše, jedná se o výstup fyzický. Vyhodnocení výstupu, které je nutné pro zhodnocení dosažených cílů a ověření funkčnosti nástroje, je tedy možné provést buď vizuálně, tj. od oka, nebo použitím záznamového zařízení. Vizuální způsob testování je nutně nepřesný. Lidské oko není totiž schopné při hodnotě snímkové frekvence vyšší jak 30 snímků za sekundu rozeznat jednotlivé snímky, natož počátek překreslovacího procesu. Naštěstí můžeme použít záznamové zařízení. Tím může být jak kamera, tak snímač umístěný před monitory. Použití kamery pouze vyžaduje, aby tato disponovala vyšší snímkovou frekvencí, než je snímková frekvence přehrávaného videa. Za výhodu tohoto přístupu lze považovat, že na rozdíl od snímače neklade omezení na počet zobrazovacích zařízení. Snímač je závislý na počtu zobrazovacích zařízení, umístění i testovacím, demonstračním videu. Takové video by muselo být vytvořeno ze snímků vyplněných kontrastními barvami a frekvence, se kterou by se tyto snímky střídaly, by musela být měnitelná za běhu, abychom se vyvarovali nezaznamenatelnému posunu.

7.2 Průběh testování a dosažené výsledky

7.2.1 Unit testy

K implementaci unit testů byl použit framework Catch[1]), který je součástí libyuri. Každý testovací soubor představuje spustitelný modul, který ověřuje funkcionalitu pouze jedné jednotky.

[1]) `https://github.com/philsquared/Catch`

- `module_onepc_protocol_test` ověřuje správnost výpočtu průměrné doby zpoždění v modulu `onepc_protocol`.
- `module_twopc_protocol_test` ověřuje funkčnost pomocných funkcí obsažených v modulu `twopc_protocol`.
- `module_state_table_test` ověřuje funkčnost přechodů mezi stavy v přechodové tabulce a zpětné volání procedury při daných přechodech. Dále testuje, zdali automat vždy zachovává deterministický stav.

Spuštění všech testů se provádí pomocí zadání příkazu `make tests` do příkazové řádky. Po vykonání celé množiny unit testů lze konstatovat, že všechny testované komponenty pracují správně.

7.2.2 Integrační testy

V rámci integračních testů byla ověřována vzájemná spolupráce datové, synchronizační a prezentační vrstvy. Testování bylo prováděno jak v prostředí distribuovaného systému, tak i na jednom počítači s využitím testovacích hran. Třída `UnreliableSingleFramePolicy` implementuje hranu, ve které dochází k náhodnému ztrácení paketů. V průběhu testování byla tato hrana použita pro simulaci nespolehlivého spojení, např. streamování. Třída `DelayedSingleFramePolicy` pro změnu implementuje hranu spolehlivou, avšak napodobující zpomalený přenos. Dané hrany umožnily věrohodnou simulaci poruch bežných při streamování videí.

Testování ukázalo, že modul `twopc_protocol` je odolný vůči ztrátám paketů i jejich mírnému zpoždění. V modulu `onepc_protocol` způsobí ztráty paketů trhání obrazu, jejich zpoždění pak vede ke ztrátě synchronizace.

7.2.3 Systémové testy

Naším cílem bylo pořídit videozáznamy demonstračních videí a na jejich základě vyhodnotit výsledné zobrazování jednotlivých video snímků a průběh jejich překreslování. Vyhodnocování výstupů systémových testů se provádělo jak vizuálně, tak pomocí záznamových zařízení. Těmi byly v našem případě dvě videokamery. První kamera GoPro[1]) byla vybrána pro zaznamenání průběhu překreslování a zobrazování, neboť umožňuje natáčení o vysoké snímkové frekvenci. Druhá kamera byla schopna zaznamenávat video ve vysokém rozlišení. Videím pořízenými těmito kamerami a zaznamenávajících průběh přehrávání demonstračních videí budeme říkat dokumentační videa. Použití snímače bylo zamítnuto jednak z časových důvodů a jednak kvůli jeho závislosti na počtu zobrazovacích zařízení.

Při analýze dokumentačních videí byl kladen důraz na stanovení doby potřebné k synchronizaci vykreslování na zobrazovacích zařízeních. Jedná se o dobu od začátku vykreslování v jednom z uzlů do začátku vykreslování ve všech uzlech. Tato hodnota je podstatná, na jejím základě lze totiž stanovit, jak bude systém vnímán uživateli.

Testování videopřehrávače bylo provedeno v laboratořích Institutu intermédií[2]) v prostředí distribuovaného systému se čtyřmi uzly, z nichž jeden posloužil jako koordinátor. Je také nutné zmínit, že dané uzly měly různý stupeň výkonnosti a odlišné grafické karty. Všechny uzly se nacházely v lokální síti a disponovaly vlastními daty.

Na obrázku 7.1 je zobrazen průběh testování videopřehrávače v laboratořích IIM. Data byla pořízena na kameru s hodnotou snímkové frekvence dvakrát vyšší (50 fps), než byla snímková frekvence demonstračního videa (25 fps). To znamená, že dva snímky

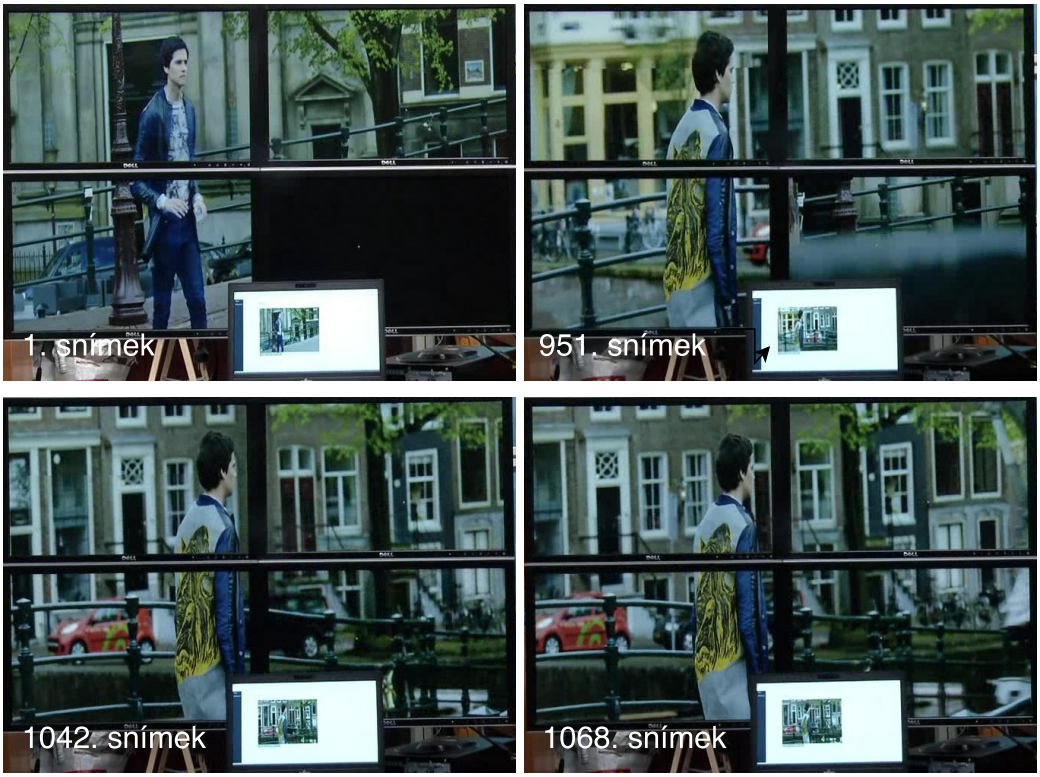
[1]) `http://gopro.com/`
[2]) `https://www.iim.cz/`

pořízené kamerou odpovídají jednomu snímku demonstračního videa. Chyba měření tak činila 10 ms. Na druhém snímku je vidět, že levý horní displej zahájil překreslování obrazovky jako první. Na třetím a čtvrtém snímku je patrné zpoždění pravého horního uzlu. Na pátém a následujícím (vynechaném) šestém snímku pozorujeme výsledný obraz. Poslední, sedmý snímek zobrazil již nový snímek demonstračního videa (patrná změna polohy oka zvířete). K úplnému vykreslení obrazu ve většině uzlů (všechny kromě horního pravého) došlo za 40 ms. Uzel, který odpovídá za zobrazení v pravé horní obrazovce, se opozdil o 10 ms. Doba potřebná k synchronizaci vykreslování na zobrazovacích zařízeních se rovnala 80 ms.

Obrázek 7.1. Ukázka průběhu zobrazování snímků demonstračního videa v prostředí distribuovaného videopřehrávače při použití jednofázové metody komunikace. Video snímky uvedené na obrázku pochází z dokumentačního videa, které bylo natočeno profesionální kamerou se snímkovou frekvenci 50 fps. Snímky na obrázku zachycují pro člověka skoro nepostřehnutelné překreslování obrazu, ke kterému dochází v průběhu zobrazování. Popisky byly k snímkům přidány dodatečně a představují číslování snímků dokumentačního videa. Jako demonstrační video byl použit animovaný film Gran Dillama[1]).

Zotavení uzlu po výpadku bylo také otestováno v prostředí laboratoří IIM. Obrázek 7.2 znázorňuje průběh zotavení po výpadku, od nového spuštění uzlu po jeho opětovnou resynchronizaci. V daném případě byl výpadek uzlu započat po čase 00:01:25. K opětovnému spuštění uzlu došlo po 30 s (tedy přibližně v čase 00:01:55). Prvních 951 snímků pořízených z dokumentačního videa od místa výpadku obsahuje černou obrazovku, což znamená, že v daný okamžik se uzel snažil získat snímek od koordinátora. Dobu, kterou trvalo získání snímku, nejde určit přesně, avšak lze stanovit dobu, která uplynula od zapnutí nástroje do zobrazení prvního video snímku. Daná doba se rovná 19 s. Na 951. snímku je znázorněna snaha vykonavatele o resynchronizaci. Na dalším snímku vykonavatel již dosáhl synchronizovaného stavu. Na 1068. snímku, který byl pořízen několik sekund po dokončení zotavení, je vidět, že zotavení po výpadku proběhlo úspěšně a přehrávání ve všech uzlech je opět synchronizované.

[1]) `http://www.caminandes.com/`



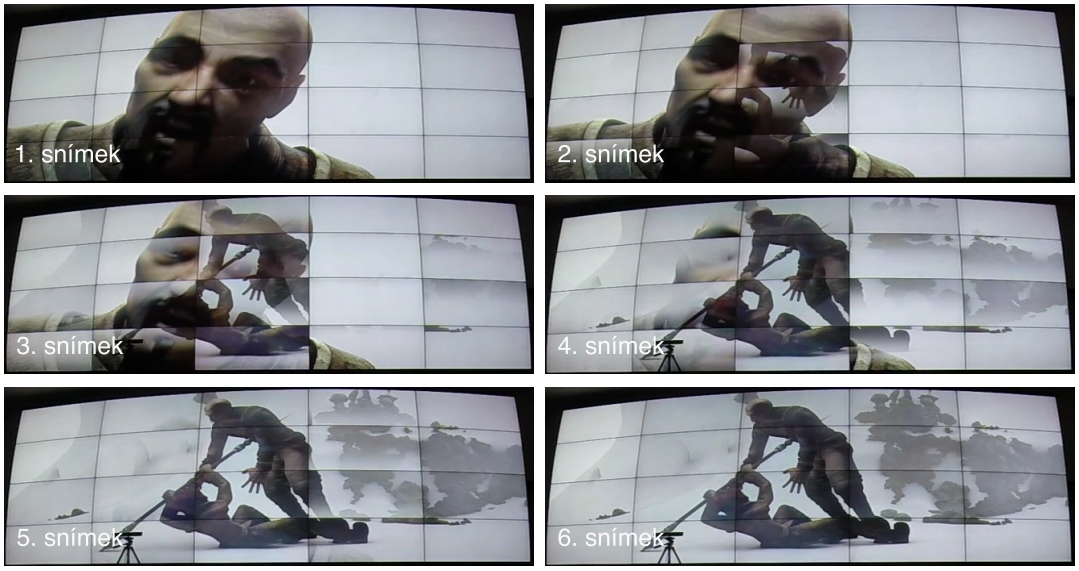
Obrázek 7.2. Ukázka zotavení uzlu po výpadku v prostředí distribuovaného videopřehrávače při použití dvoufázové metody komunikace s částečným potvrzením. Video snímky uvedené na obrázku pochází z dokumentačního videa, které bylo natočeno profesionální kamerou se snímkovou frekvenci 50 fps. Snímky na obrázku zachycují pro člověka skoro nepostřehnutelné překreslování obrazu, ke kterému dochází v průběhu zobrazování. Popisky byly k snímkům přidány dodatečně a představují číslování snímků dokumentačního videa. Jako demonstrační video byl použit animovaný film Tears of Steel[1]).

Videopřehrávač byl také otestován v SAGElab[2]) laboratoři, která je vybavena velkoplošným vizualizačním zařízením. Zařízení se skládá z šesti serverů Dell PowerEdge T620 s grafickými kartami NVIDIA GeForce GTX 680 propojenými vysokorychlostní sítí. Pět uzlů disponuje zobrazovacími zařízeními, které tvoří vždy čtveřice displejů uspořádaných do sloupce. Poslední, šestý uzel touto zobrazovací jednotkou nedisponuje. V prostředí SAGElab byl koordinátor umístěn právě na tento uzel. Ovládání toku přehrávání se provádělo přes webové rozhraní. Všechny uzly disponovaly vlastními daty.

Obrázek 7.3 tvoří posloupnost snímků, která byla získána z dokumentačního videa. Při testování dvoufázové metody s částečným potvrzením byl přes příkazovou řádku nastaven počet vykonavatelů na pět a počet požadovaných potvrzení na 60 % (potvrzení se tak očekávala pouze od třech vykonavatelů). Ostatní parametry byly nastaveny na implicitní hodnoty. Dokumentační video bylo natočeno kamerou se snímkovou frekvencí 100 fps, demonstrační video mělo snímkovou frekvenci 25 fps. Chyba měření v daném případě byla pouhých 5 ms. V případě striktního dodržení snímkové frekvence na straně videopřehrávače můžeme prohlásit, že počet snímků, který dokumentační video zaznamenalo za dobu zobrazení jednoho snímku demonstračního videa, je roven čtyřem.

[1]) `https://mango.blender.org/`
[2]) `http://sagelab.cesnet.cz/o-laboratori/`

Na druhém snímku dokumentačního videa je zaznamenán začátek překreslování, což znamená, že v daném bodě již došlo k synchronizaci. Přestože dobu potřebnou k synchronizaci nejde stanovit přesně, lze vypočítat dobu, která uběhla od ukončení vykreslování předchozího snímku na nejpomalejším uzlu po první náznaky začátku vykreslování dalšího snímku na nejrychlejším uzlu. Tento uzel by totiž nezačal vykreslovat další snímek, pokud by tímto snímkem již nedisponovaly i ostatní uzly. Lze tedy prohlásit, že v okamžiku, kdy nejrychlejší uzel začne s vykreslováním, je systém synchronní (to ale ještě neznamená, že je synchronní proces vykreslování). Tato doba představuje čas potřebný na přípravu snímku, synchronizaci uzlů a částečné vykreslení. V našem případě se rovná 10 ms a je reprezentována prvním a druhým snímkem. Doba synchronizace vykreslování na zobrazovacích zařízeních se rovná 20 ms (druhý až čtvrtý snímek). K úplnému překreslení obrazu ve všech uzlech (druhý až pátý snímek, jeden snímek demonstračního videa) dochází přibližně za 30 ms, což je doba lidským okem nepostřehnutelná.



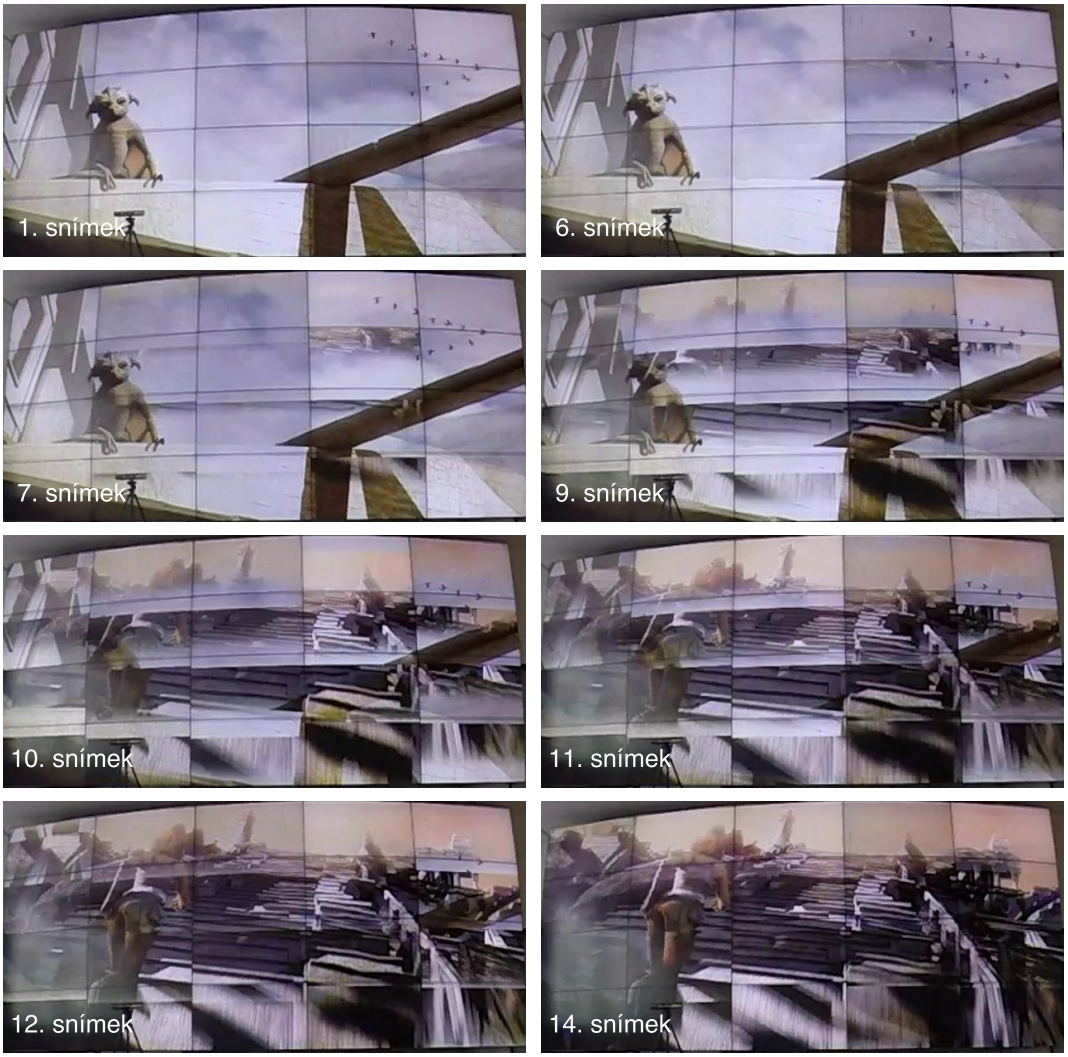
Obrázek 7.3. Ukázka průběhu zobrazování snímků demonstračního videa v prostředí distribuovaného videopřehrávače při použití dvoufázové metody komunikace s částečným potvrzením. Video snímky uvedené na obrázku pochází z dokumentačního videa, které bylo natočeno kamerou GoPro se snímkovou frekvenci 100 fps. Snímky na obrázku zachycují pro člověka skoro nepostřehnutelné překreslování obrazu, ke kterému dochází v průběhu zobrazování. Popisky byly k snímkům přidány dodatečně a představují číslování snímků dokumentačního videa. Jako demonstrační video byl použit animovaný film Sintel[1]).

Zobrazování video snímků v distribuovaném videopřehrávači při použití dvoufázové metody s časovým limitem je znázorněno na obrázku 7.4. Demonstrační video Sintel bylo přehráno se snímkovou frekvenci 25 fps, dokumentační video bylo natočeno kamerou GoPro se snímkovou frekvencí 240 fps. Chyba v daném případě byla proměnlivá, nicméně její maximální hodnota se rovnala 2,08 ms. Za předpokladu, že videopřehrávač striktně dodržoval snímkovou frekvenci, můžeme prohlásit, že jeden snímek demonstračního videa byl zaznamenán v dokumentačním videu přibližně desetkrát.

První snímek na obrázku zobrazuje nový, právě vykreslený snímek demonstračního videa. Na šestém snímku je vidět zahájení procesu vykreslování nového snímku. To znamená, že v daný okamžik již došlo k synchronizaci mezi uzly. Dobu synchronizace

[1]) `https://durian.blender.org/`

nelze určit s jistotou, jelikož synchronizační algoritmus dostává snímky s určitou prodlevou, lze ale říct, že k přípravě snímků na všech uzlech, jejich následné synchronizaci a začátku vykreslování aspoň v jednom z uzlů došlo za 21 ms. Šestý až dvanáctý snímek znázorňují průběh vykreslování. Na šestém snímku je vidět začátek překreslování jen v jediném, čtvrtém uzlu zleva, na devátém snímku je už patrné, že k překreslování dochází ve většině uzlů. To znamená, že proces překreslování snímků je synchronizovaný po 12 ms. Poslední snímek pak zobrazuje nově vykreslovaný snímek. Samotné vykreslení nového snímku bylo provedeno za 25 ms.



Obrázek 7.4. Ukázka průběhu zobrazování snímků demonstračního videa v prostředí distribuovaného videopřehrávače při použití dvoufázové metody komunikace s časovým limitem. Video snímky uvedené na obrázku pochází z dokumentačního videa, které bylo natočeno kamerou GoPro se snímkovou frekvenci 240 fps. Snímky na obrázku zachycují pro člověka skoro nepostřehnutelné překreslování obrazu, ke kterému dochází v průběhu zobrazování. Popisky byly k snímkům přidány dodatečně a představují číslování snímků dokumentačního videa. Jako demonstrační video byl opět použit animovaný film Sintel.

Kapitola 8
Diskuse

8.1 Vyhodnocení výsledků testů

V průběhu systémových testů jsme se zaměřili na stanovení doby, po které dojde k synchronizaci vykreslování na zobrazovacích zařízeních. Tento údaj udává, po jakou dobu byl obraz mezi uzly nekonzistentní. Čím vyšší byla tato hodnota, tím zřetelnější byla nekonzistence obrazu.

Jednofázová metoda komunikace se i přes svou jednoduchost prokázala jako efektivní způsob synchronizace v systémech se stejně výkonnými uzly. V případě uzlů s odlišnou výkonností může být doba procesu synchronizace vyšší než doba zobrazení snímku (viz obrázek 7.1), což vede k trhání obrazu. Avšak v našem případě bývá čas potřebný k synchronizaci proměnlivý a obvykle není vyšší než čas na zobrazení jednoho snímku (viz video na přiloženém DVD). Z toho lze odvodit, že chování metody je ve mnohém závislé na výkonnosti uzlů. Nicméně tato vlastnost byla očekávána, neboť použitá metoda komunikace není spolehlivá a nebere v potaz výkonnost uzlů.

Obě dvě dvoufázové metody komunikace prokázaly mnohem lepší výsledky než metoda předchozí. Oba způsoby pracovaly korektně a byly schopny reagovat i na pomalejší uzly.

Metoda s částečným potvrzením se ukázala jako velmi spolehlivá a efektivní v distribuovaných systémech s konečným počtem uzlů (viz obrázek 7.3). K synchronizaci vykreslování docházelo přibližně po 20 ms, což znamená, že nesynchronizované stavy se objevovaly po velmi krátkou dobu a lidské oko je nebylo schopné zaznamenat. Tento způsob komunikace, na rozdíl od metody s časovým limitem, přenáší zodpovědnost za zobrazování snímků na stranu koordinátora. Toho můžeme využít v případě systémů, kde výsledný obraz má být zobrazen buď na všech uzlech, nebo na žádném z nich (za to je zodpovědný parametr `strict`). Příkladem takového systému může být systém s různě výkonnými uzly.

Metoda s časovým limitem se ukázala jako velmi spolehlivá v případě distribuovaných systémů s proměnlivým počtem uzlů (viz obrázek 7.4). Při testování v průběhu dané metody nebyly zaznamenány žádné nesynchronizované stavy. Analýza dokumentárních videí prokázala, že k synchronizaci procesu vykreslování došlo ve všech uzlech po 12 ms. To znamená, že stavy, v nichž vykreslování není synchronní, nejsou rozpoznatelné člověkem. Metodu s časovým limitem lze použít v systémech, kde musí být obraz neustále zobrazen, i v případě výpadku nebo absence dat v jednom z uzlů.

Zotavení uzlu po výpadku také fungovalo v každé metodě korektně. Uzel po výpadku byl schopný navázat spojení s koordinátorem a sesynchronizovat s ním svůj stav (viz obrázek 7.2 a video na přiloženém DVD). Proces zotavení je ale bohužel pomalý. Důvodem je, že vykonavatel musí zpracovat všechny snímky od začátku, neboť jiný přístup k datům v rámci datové vrstvy nebyl ve frameworku libyuri implementován.

Všechny metody komunikace mezi uzly, pomocí nichž bylo docíleno synchronizace přehrávaní, prokázaly dobré výsledky, a to i přesto, že jsou zodpovědné pouze za synchronizaci snímků před samotným vykreslováním. Kdyby se nám podařilo synchronizo-

vat samotné grafické karty, mohly bychom dobu synchronizace při vykreslování zmenšit přibližně o polovinu.

8.2 Zhodnocení dosažených cílů

8.2.1 Synchronizované videopřehrávače na více než dvou uzlech

Daný požadavek se podařilo splnit. Výsledný nástroj dovoluje synchronizované přehrávání videí v distribuovaných systémech. Synchronizační vrstva navíc poskytuje více způsobů synchronizace a dovoluje tak přizpůsobit nástroj požadavkům klienta (více v podkapitole 6.2).

8.2.2 Přehrávání celého videa nebo jen jeho částí

Daný požadavek se podařilo splnit. Výběr zobrazovací plochy lze nastavit pomocí příkazové řádky nebo v konfiguračním souboru. Výpočet plochy k oříznutí videa může proběhnout automaticky a při použití skriptu určeného ke vzdálenému spuštění (více v podkapitole 6.4).

8.2.3 Distribuce videa od jednoho uzlu ostatním

Daný požadavek se podařilo splnit. Využita byla knihovna ultragrid. Vysílač se může nacházet i mimo systém. Podpora distribuovaného rozložení se zapne pomocí parametru `data_distribution` (více v podkapitole 6.1).

8.2.4 Přehrávání s lokálním a distribuovaným rozložením dat

Daný požadavek se podařilo splnit. Videopřehrávač podporuje jak lokální, tak distribuované rozložení dat. V případě distribuovaného rozložení dat musí v systému existovat uzel, který data poskytne.

8.2.5 Grafické uživatelské rozhraní a ovládání toku přehrávání

Daný požadavek se podařilo splnit. Grafické uživatelské rozhraní poskytuje webový server na straně koordinátora (více v podkapitole 6.3). Rozhraní představuje webovou aplikaci, která uživateli nabízí náhled na video a základní ovládací prvky (spustit, zastavit, přetočit). Webové rozhraní je přístupno i mimo uzel koordinátora, což umožňuje ovládání z jakéhokoliv zařízení v dané síti.

8.2.6 Podpora resynchronizace

Daný požadavek se podařilo splnit. Mechanismus resynchronizace byl implementován v rámci synchronizační vrstvy. Dovoluje zapojení uzlů do distribuovaného přehrávání. To znamená, že spuštění všech uzlů nemusí proběhnout před začátkem přehrávání na straně koordinátora (kromě případu dvoufázové metody s částečným potvrzováním s nastaveným konstantním počtem vykonavatelů). Videopřehrávač si poradí s proměnlivým počtem uzlů, v případě připojení dalšího uzlu do sítě ho přidá do množiny svých podřízených uzlů.

8.2.7 Konfigurace pomocí souborů a příkazové řádky

Daný požadavek se podařilo splnit. Díky frameworku libyuri je konfigurace nástroje možná jak pomocí konfiguračních souborů, tak příkazové řádky. Nicméně, konfigurační soubory ve složce *configs/player* byly navrženy tak, aby uživatel nemusel vytvářet žádné

nové soubory a vystačil si s ovládáním přes příkazovou řádku. Konfigurační soubor, případně argumenty do příkazové řádky, se uvádí při spuštění videopřehrávače. Lze je také uvést v každém uzlu zvlášť, a nakonfigurovat tak dané uzly dle svých potřeb.

Kapitola 9
Závěr

Tato bakalářská práce měla za cíl vytvořit distribuovanou verzi videopřehrávače. Hlavním problémem v přizpůsobení videopřehrávače k práci v distribuovaném prostředí bylo zaručení synchronizace přehrávání videí. Po podrobném prozkoumání problematiky distribuovaných systémů a jejich charakteristických rysů byly navrženy a implementovány tři způsoby komunikace mezi uzly systému. Každý z těchto způsobů má své výhody a nevýhody, společně však pokrývají všechny možné případy užití.

Součástí přehrávače je také grafické rozhraní a nástroj pro vzdálené ovládání všech uzlů v síti.

Na základě testování, které proběhlo jak v prostředí SAGElab, tak i v laboratořích IIM, můžeme prohlásit, že vytyčené cíle byly splněny. Nástroj umožňuje synchronní distribuované přehrávání videí. Nedochází k trhání, zasekávání ani jiným obrazovým neduhům.

Literatura

[1] Understanding Jitter in Packet Voice Networks (Cisco IOS Platforms). In: *Cisco Systems, Inc* [online]. 2006 [cit. 2014-11-15]. Dostupné z: http://www.cisco.com/c/en/us/support/docs/voice/voice-quality/18902-jitter-packet-voice.html

[2] KSHEMKALYANI, Ajay D a Mukesh SINGHAL. *Distributed computing: principles, algorithms, and systems*. Cambridge: Cambridge University Press, 2008, 1 - 239. ISBN 978-0-521-87634-6.

[3] KLIMEŠ, Cyril. *Distribuované systémy: texty pro distanční studium* [online]. Ostravská univerzita v Ostravě, Přírodovědecká fakulta Katedra informatiky a počítačů, [2004][cit. 2015-01-03]. Dostupné z: Server Ostravské univerzity.

[4] CHAUHAN, Ganpat Singh, Mukesh Kumar GUPTA a Ajay KHUNTETA. IOCP: Implementation of Optimized Commit Protocol. In: *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)* [online]. IEEE, 2014, s. 1-6 [cit. 2015-01-03]. ISBN 978-1-4799-4040-0. DOI: 10.1109/ICRAIE.2014.6909141. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6909141

[5] SKEEN, D. a M. STONEBRAKER. A Formal Model of Crash Recovery in a Distributed System. In: *IEEE Transactions on Software Engineering* [online]. 1983, s. 219-228 [cit. 2015-11-10]. ISSN 0098-5589. DOI: 10.1109/TSE.1983.236608. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1703048

[6] SKUPA, Jindřich. *KIVFS - Synchronizace a trasování požadavků*. Plzeň, 2012. Dostupné z: https://otik.uk.zcu.cz/bitstream/handle/11025/3030/diplomova_prace.pdf?sequence=1. Diplomová práce. Západočeská univerzita v Plzni.

[7] About: Scalable Adaptive Graphics Environment. In: *SAGE™ Scalable Adaptive Graphics Environment* [online]. University of Illinois Board of Trustees, 2013 [cit. 2015-01-03]. Dostupné z: http://sage.sagecommons.org/project/about/

[8] JEONG, Byungil, Luc RENAMBOT, Ratko JAGODIC, Rajvikram SINGH, Julieta AGUILERA, Andrew JOHNSON a Jason LEIGH. High-Performance Dynamic Graphics Streaming for Scalable Adaptive Graphics Environment. In: *ACM/IEEE SC 2006 Conference (SC'06)* [online]. IEEE, 2006 [cit. 2014-11-12]. ISBN 0-7695-2700-0. DOI: 10.1109/SC.2006.35. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4090198

[9] NAM, Sungwon, Sachin DESHPANDE, Venkatram VISHWANATH, Byungil JEONG, Luc RENAMBOT a Jason LEIGH. Multi-application inter-tile synchronization on ultra-high-resolution display walls. In: *Proceedings of the first annual ACM SIGMM conference on Multimedia systems - MMSys '10* [online]. New York, USA: ACM Press, 2010 [cit. 2014-12-06]. ISBN 9781605589145. DOI: 10.1145/1730836.1730854. Dostupné z: http://portal.acm.org/citation.cfm?doid=1730836.1730854

[10] Synchronized playback over a network. In: *MPlayer Documentation* [online]. © 2000-2011 [cit. 2015-01-03]. Dostupné z: http://www.mplayerhq.hu/DOCS/HTML/en/networksync.html

[11] LEDVINA, Jiří. *Přednášky z distribuovaných systémů*. [2008], 122 s. Dostupné z: http://zcu.arcao.com/kiv/ds/ds.pdf

[12] TRÁVNÍČEK, Zdeněk. *Libyuri: Popis knihovny libyuri*. Praha, 2015. [cit. 2015-05-13].

[13] About. In: *UltraGrid: Open-source software for high bandwidth, low latency network transmissions*. [online]. 2015 [cit. 2015-05-14]. Dostupné z: https://www.sitola.cz/igrid/index.php/About

Příloha A
Použité zkratky

AVI	Audio Video Interleave
GLX	OpenGL Extension to the X Window System
GUI	Graphical User Interface
HTML	HyperText Markup Language
MPEG-1	Motion Pictures Experts Group
MP3	MPEG Audio Layer III
SAGE	Scalable Adaptive Graphics Environment
SDL	Simple DirectMedia Layer
SSH	Secure Shell
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VIVO	Vivo Software's video format
XML	Extensible Markup Language

Příloha B
Uživatelská příručka

Distribuovány videopřehrávač je program běžící v prostředí distribuovaného systému, který se skládá z množiny uzlů, které mezi sebou komunikují pomocí zpráv. K řízení uzlů se používá centralizovaný model, což znamená, že jeden uzel hraje roli koordinátora a stručně řečeno řídí proces přehrávání videa v rámci celého systému. Uzlům podřízeným koordinátorovi se říká vykonavatelé.

B.1 Spuštění

Videopřehrávač se spouští v každém uzlu (zdroji) zvlášť z příkazové řádky. Při spuštění je nutné uvést konfigurační soubor (soubor *player.xml* ve složce *configs/player*) a údaje o způsobu rozložení dat mezi uzly (viz kapitolu 2). Konfigurační soubor poskytuje velké množství nastavitelných parametrů (argumentů), jejichž nastavení lze změnit přes příkazovou řádku. Všechny argumenty musí být uvedeny při spuštění a nelze je v žádném případě měnit za běhu programu.

Příkaz ke spuštění videopřehrávače na straně vykonavatele pak může vypadat následovně:

```
1  yuri2 player.xml path=video.avi
```

Vzhledem k tomu, že spouštění nástroje postupně ve všech uzlech by bylo náročné (zvlášť v případě velkého množství vzdálených zdrojů), je možné spuštění automatizovat použitím speciálního skriptu *player_remote_control.py*, jenž se nachází ve složce *remote_control*. Vzdálené spuštění nástroje je možné provést z jakéhokoliv zdroje, který obsahuje skript *player_remote_control.py*. Při spuštění je nutné uvést cestu k XML souboru, který obsahuje informace o vzdálených zdrojích (IP adresu, login a heslo pro připojení) a také příkazy, které chceme spustit na vzdáleném zdroji. Pokud při spuštění skriptu bude uvedeno horizontální (`resol_x`) a vertikální (`resol_y`) rozlišení, dojde k výpočtu zobrazené plochy dle pořadí uvedených klientů. Vypočítaná hodnota se doplní místo dvojitých složených závorek (`geometry={ }`), které uvedeme v příkazu ke spuštění.

Nejjednodušší XML soubor, jenž obsahuje informace ke vzdálenému spuštění videopřehrávače na třech zdrojích (jeden koordinátor a dva vykonavatelé) může vypadat následovně:

```
1  <!-- iim_sage.xml -->
2  <clients>
3      <client>
4          <ip>192.168.23.90</ip>
5          <username>worker</username>
6          <password>123456</password>
7          <orders>
8              <order>
9               DISPLAY=:0 yuri2 player.xml path=Sintel.mp4
```

```
                synch_method=1 fps=25 geometry={}
               </order>
           </orders>
       </client>
       <client>
           <ip>192.168.23.169</ip>
           <username>worker</username>
           <password>123456</password>
           <orders>
               <order>
                DISPLAY=:0 yuri2 player.xml path=Sintel.mp4
                synch_method=1 fps=25 geometry={}
               </order>
           </orders>
       </client>
       <client>
           <ip>192.168.23.190</ip>
           <username>worker</username>
           <password>123456</password>
           <orders>
               <order>
                DISPLAY=:0 yuri2 player.xml path=Sintel.mp4
                coordinator=true synch_method=1 fps=25
               </order>
           </orders>
       </client>
   </clients>
```

Tento konfigurační soubor zajistí spuštění videopřehrávače na třech zdrojích. Zdroj s IP adresou 192.168.23.190 bude hrát roli koordinátora (na to poukazuje argument `coordinator`). K synchronizaci přehrávání bude použita metoda jednofázové komunikace s jednosměrným potvrzováním (argument `synch_method`). Všechny uzly obsahují video Sintel.mp4, které na vykonavatelích bude oříznuto na základě hodnot, které skript doplní místo složených závorek.

Spuštění skriptu bude vypadat následovně:

```
player_remote_control.py iim_sage.xml --resol_x=1920 --resol_y=1080
```

B.2 Ovládání přehrávání

Videopřehrávač se ovládá přes webové rozhraní, které je přístupné na IP adrese koordinátora, výchozí port je 8080 (např. v případě předchozího konfiguračního souboru bude webové rozhraní přístupné na http://192.168.23.190:8080). Rozhraní poskytuje základní ovládací prvky (spustit, zastavit, přetočit). Ke spuštění přehrávání na všech zdrojích nedojde po spuštění videopřehrávače, ale až po stisknutí tlačítka se symbolem `spustit` ve webovém rozhraní.

Video lze také přetáčet. Během přetáčení se rychlost přehrávání zvýší, na konci přetáčení se rychlost vrátí na svou počáteční hodnotu (viz videa na přiloženém DVD).

Způsob vypnutí videopřehrávače je závislý na způsobu jeho spuštění. V případě vzdáleného spuštění proběhne vypnutí na všech zdrojích automaticky po stisknutí `Ctrl+C` nebo při zadání příkazu `exit` do příkazové řádky na zdroji, který byl za toto spuštění zodpovědný. Jinak musí být vypnutí provedeno manuálně na každém zdroji zvlášť.

B.3 Synchronizované přehrávání

Synchronizovaného přehrávání ve videopřehrávači bylo docíleno použitím tří metod komunikace mezi uzly. Výběr správné metody komunikace je důležitý, neboť každá z metod je závislá na prostředí (např. na počtu zdrojů), v němž bude použita.

Metody komunikace rozdělujeme na:

- **Jednofázová komunikace** je vhodná pro distribuované systémy s menším počtem uzlů. Její hlavní výhodou je rychlost a absence vytížení sítě.
- **Dvoufázová komunikace s časovým limitem** se dá použít, pouze pokud je známa snímková frekvence daného videa a daná hodnota není proměnlivá. Komunikace s časovým limitem je vhodná jak pro systémy s variabilním počtem vykonavatelů, tak pro systémy s konstantním počtem vykonavatelů.
- **Dvoufázová komunikace s částečným potvrzením** je vhodná pro systémy, ve kterých je počet vykonavatelů konstantní nebo kde zodpovědnost za zobrazení je pouze na straně koordinátora.

Výběr způsobu komunikace probíhá přes příkazovou řádku (viz podkapitolu B.6) a musí být stejný pro každý zdroj.

B.4 Rozložení dat

Nástroj podporuje distribuované a lokální rozložení dat. Pokud zdroj bude přijímat data ze sítě (distribuované rozložení), je dodatečně nutné určit, zdali bude také odesilatelem (vysílačem), nebo jen pouhým příjemcem. V případě lokálního rozložení dat je zapotřebí uvést cestu k lokálnímu souboru s videem. Nastavení způsobu rozložení dat a cesty k souboru je popsáno v podkapitole B.6.

B.5 Zobrazení

Zobrazení na straně koordinátora je definováno webovým rozhraním, jak již bylo zmíněno v podkapitole B.2. Na straně vykonavatele dochází po spuštění k zobrazení SDL nebo GLX oken. Pomocí příslušných argumentů je možné definovat plochu k zobrazení (viz podkapitolu B.6).

B.6 Argumenty příkazové řádky

Aplikace `yuri2` spustí videopřehrávač na základě konfiguračního souboru *player.xml*. Přizpůsobení nástroje uživatelským požadavkům probíhá pomocí argumentů. Všechny argumenty mají přiřazenou hodnotu určitého typu. Typy hodnot rozdělujeme na následující skupiny:

- `BOOL_VAL` může nabývat hodnot true, nebo false.
- `INT_VAL` může nabývat celočíselných hodnot.
- `RESOL` musí být uvedena ve tvaru `resol_X x resol_Y + x + y`, kde `resol_X` a `resol_Y` představují rozlišení videa a `x` a `y` jsou souřadnice.
- `VAL` může nabývat jakékoliv hodnoty.

Dále jsou popsány nejdůležitější argumenty, které je možné definovat před spuštěním videopřehrávače. Předchází vzorový zápis:

- `nazev_parametru=[datovy_typ]` popis argumentu. *Výčet metod, pro které lze argument použít. Nejsou-li žádné uvedeny, lze ho použít vždy.*
  - `hodnota` výčet možných hodnot argumentu
- `path=[VAL]` uvádí cestu k souboru.
- `data_distribution=[INT_VAL]` definuje rozložení dat mezi uzly, podrobněji popsáno v podkapitole B.4.
  - `1` (implicitně) zdroj disponuje vlastními daty.
  - `2` zdroj poskytuje data ostatním prostřednictvím sítě (vysílač).
  - `3` zdroj odebírá data ze sítě (přijímač).
- `coordinator=[BOOL_VAL]` (implicitně false) určuje, zdali bude daný uzel koordinátorem.
- `synch_method=[INT_VAL]` výběr způsobu komunikace mezi zdroji (viz podkapitolu B.3).
  - `1` (implicitně) jednofázová metoda.
  - `2` dvoufázová metoda s částečným potvrzením.
  - `3` dvoufázová metoda s časovým limitem.
- `frame_index=[BOOL_VAL]` (impicitně true) způsob číslování video snímků. Pokud je nastaven na true, synchronizace snímku probíhá na základě indexu, jenž byl obsažen v snímku. Jinak čísla video snímkům přiřazuje čítač.
- `timeout=[INT_VAL]` definuje maximální dobu čekání na odpověď (v milisekundách) na odeslanou zprávu. Daná hodnota může být vypočítána z hodnoty snímkové frekvence. Po uplynutí této doby dojde k zobrazení video snímku. *Delay estimation nebo Dvoufázová komunikace s časovým limitem*
- `period=[INT_VAL]` doba (v milisekundách), po které dojde k opakování měření doby zpoždění mezi koordinátorem a vykonavatelem. Čím je tato doba menší, tím je odhad přesnější, ale systém vytíženější. *Jednofázová komunikace*
- `central_tendency=[VAL]` nastaví způsob, dle kterého budou zprůměrována naměřená zpoždění při průchodu sítí. *Jednofázová komunikace*
  - `none` (implicitně) zpoždění nebude zprůměrováno.
  - `average` zpoždění bude vynásobeno počtem jeho výskytů a součet daných hodnot bude vydělen počtem všech měření.
  - `mode` zpoždění bude rovno hodnotě, která se mezi zpožděními vyskytuje nejčastěji.
- `strict=[BOOL_VAL]` zapne striktní režim zobrazení, což znamená, že pokud jeden z uzlů nebude schopný přehrát snímek, nedojde k jeho přehrávání nikde. *Dvoufázová komunikace s časovým limitem*
- `missing_confirmation=[INT_VAL]` uvádí maximální počet snímků, které vykonavatel může nepotvrdit, aniž by byl vyloučen ze systému. Pokud číslo nepotvrzených snímků bude vyšší, dojde k odstranění uzlů s daným id ze seznamu vykonavatelů. *Dvoufázová komunikace s časovým limitem a Dvoufázová komunikace s částečným potvrzením*
- `fps=[INT_VAL]` nastaví snímkovou frekvenci. *Dvoufázová komunikace s časovým limitem*
- `cohorts=[INT_VAL]` nastaví počet vykonavatelů. Je-li tato hodnota nastavena, je automaticky vypnuta podpora variabilního počtu vykonavatelů. *Dvoufázová komunikace s částečným potvrzením*

- `variable_cohorts=[BOOL_VAL]` uvádí, zda počet vykonavatelů může být variabilní. *Dvoufázová komunikace s částečným potvrzením*
- `confirmation=[INT_VAL]` uvádí minimální počet vykonavatelů (v procentech), kteří musí být připraveni, aby došlo k přehrávání videa. *Dvoufázová komunikace s částečným potvrzením*
- `crop=[RESOL]` ořízne zobrazovací plochu na požadovanou velikost.
- `use_sdl_window=[BOOL_VAL]` (implicitně nastaveno na true) výběr typu okna (viz podkapitolu B.5).

B.7 Scénáře použití

B.7.1 Scénář: Vytížená síť

V případě vytížené sítě bude nejlepší volbou výběr jednofázové synchronizace, neboť tolik nezahlcuje síť. Budeme předpokládat, že se přehrávání zúčastní tři uzly, ze kterých bude jeden koordinátorem. Spuštění nástroje proběhne v každém uzlu zvlášť. Příkaz, který spustíme na stráně koordinátora, bude vypadat následovně:

```
yuri2 player.xml synch_method=1 coordinator=true path=Sintel.avi
```

Po spuštění bude danému zdroji přiřazena role koordinátora a na jeho adrese bude spuštěno webové ovládací rozhraní.

Dále přejdeme ke spuštění přehrávačů na vykonavatelích (příkaz je stejný pro všechny tři vykonavatele):

```
yuri2 player.xml synch_method=1 path=Sintel.avi
central_tendency=average period=26
```

Daný příkaz definuje vykonavatele, naměřená zpoždění v odesílání paketů mezi uzly budou zprůměrována pomocí metody average, k měření zpoždění bude docházet pravidelně každých 26 milisekund.

Přehrávání spustíme přes webové rozhraní, po spuštění dojde k distribuovanému přehrávání neořízntého videa ve všech uzlech.

B.7.2 Scénář: Přehrávání videa na systému SAGE

K přehrání videa v systému SAGE použijeme dvoufázovou metodu komunikace s časovým limitem, ale není to v daném případě nutné, mohli bychom použít i metodu s částečným potvrzením. Předpokládejme distribuovaný systém složený ze čtyř zdrojů. Abychom mohli spustit videopřehrávač vzdáleně, vytvoříme XML soubor, který bude definovat všechny zdroje. Zjednodušená ukázka daného souboru je uvedena níže:

```
<clients>
    <client>
        <ip>192.168.23.190</ip>
        <username>worker</username>
        <password>123456</password>
        <orders>
            <order>cd ./Projects/yuribuild/</order>
            <order>
             DISPLAY=:0 yuri2 player.xml path=Tears_of_Steel.mp4
             coordinator=true synch_method=3 fps=25
            </order>
```

```
12          </orders>
13      </client>
14      <client>
15          <ip>192.168.23.190</ip>
16          <username>worker</username>
17          <password>654321</password>
18          <orders>
19              <order>cd ./Projects/yuribuild/</order>
20              <order>
21               DISPLAY=:0 yuri2 player.xml path=Tears_of_Steel.mp4
22               synch_method=3 fps=25 resolution=1920x1200
23               geometry=960x540+0+0
24              </order>
25          </orders>
26      </client>
27      ....
28  </clients>
```

Z daného XML souboru vyplývá, že pro synchronizaci bude použita komunikace s časovým limitem (`synch_method=3`). Také je uvedeno, že video má snímkovou frekvenci 25 fps a každý z uzlů bude zobrazovat jen výřez videa (`geometry=960x540+0+0`). Argument `resolution` uvádí rozlišení zobrazovacího zařízení na straně vykonavatele. Celý konfigurační soubor se nachází na přiloženém DVD.

Spouštění provedeme vzdáleně z koordinátora přes příkazovou řádku spuštěním *player_remote_control.py*. Po spuštění otevřeme webové rozhraní, které se nachází na stránce http://192.168.23.190:8080 a klikneme na tlačítko se symbolem `spustit`. Po několika vteřinách se spustí distribuované přehrávání.

Příloha C
Pokyny pro kompilaci

Složky `yuri` a `ultragrid` na přiloženém DVD je zapotřebí umístit do stejného adresáře. Pro kompilaci knihovny ultragrid je nutné vykonat následující kroky:

- `./autogen.sh --disable-libavcodec`
- `make`

Daný příkaz provede kompilaci knihovny. Následně je nutné vytvořit složku `yuri_build` a přejít do ní pomocí příkazové řádky. V této složce spustíme následující příkazy:

- `cmake [cesta k frameworku yuri] -DYURI_DISABLE_ULTRAGRID=OFF`
- `make`

Po kompilaci se spustitelné soubory nachází ve složce `yuri_build`.

Příloha D
Obsah přiloženého CD

/
- `bp` ..... Zdrojové soubory k textu bakalářské práce.
- `components` ..... Komponenty přidané do frameworku libyuri.
- `ultragrid` ..... Knihovna ultragrid.
- `videos` ..... Dokumentační videa pořízená během testování.
- `yuri` ..... Framework libyuri.
- `kuzneana_2015bach.pdf` ..... Text bakalářské práce.