Technická univerzita v Liberci

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2646 – Informační technologie

Studijní obor: 1802R007 – Informační technologie

Software pro přenos video-streamu mezi počítači v reálném čase

The Software for Real Time Video Streaming between Computers

Bakalářská práce

Autor:	**Vladimír Starec**
Vedoucí práce:	Ing. Miroslav Holada, Ph.D.
Konzultant:	Ing. Josef Chaloupka, Ph.D.

V Liberci dne 18. 5. 2011

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum 18. 5. 2011

Podpis

Poděkování

Děkuji panu Ing. Miroslavu Holadovi, Ph.D., vedoucímu mé bakalářské práce, za trpělivost a odborné vedení při jejím vypracování.

V Liberci dne 18. 5. 2011 .....

Vladimír Starec - autor

Název: **Software pro přenos video-streamu mezi počítači v reálném čase**

Autor: Vladimír Starec

Vedoucí: Ing. Miroslav Holada, Ph.D.

Abstrakt: Cílem práce je navázat na bakalářský projekt a realizovat v programovacím jazyce C++/CLI software pro přenos video-streamu mezi počítači v modelu klient - server. Hlavními kritérii pro software je rychlost přenosu v reálném čase, datová nenáročnost přenosu, možnost ovládat kameru z klientské strany a dále navržení a implementace přidělování práv jednotlivým klientům. V textové zprávě jsou shrnuty teoretické informace o jednotlivých použitých technologiích, popis architektury programu, jeho důležitých datových struktur a algoritmů. Součástí jsou také výsledky praktického testování softwaru a měření datové náročnosti.

Klíčová slova: video streaming, directshow, windows driver model, WDM, komprese, jpeg, TCP

Title: **The Software for Real Time Video Streaming between Computers**

Author: Vladimír Starec

Thesis leader: Ing. Miroslav Holada, Ph.D.

Abstract: The goal of the thesis is to follow up the bachelor's project and to build software for client-server video-streaming using C++/CLI programming language. The main software criteria are 1) real time transfer, 2) low data volume demand, 3) possibility to control the camera from the client's side, 4) creating and implementing rules for distribution of the camera control rights. The theory part of the thesis describes the used technologies. The practical part of the thesis contains information about software architecture, used data structures and algorithms as well as software testing and data transfer measuring.

Keywords: video streaming, directshow, windows driver model, WDM, compression, jpeg, TCP

Obsah

Seznam obrázků

1 Úvod

Před započetím práce a vlastním výběrem tématu bylo nutné odpovědět si na následující otázky: „Je zapotřebí vytvářet další program pro streamování videa? Má smysl vytvářet další kopii programu VLC, Skype apod., když dnes existuje mnoho alternativ pro přenos videa?“ Odpověď na otázky je jednoduchá. Tyto program jsou funkční a velmi rozšířené, ale nehodí se k účelu, pro který je tento software určen. Software je určen pro použití v oblasti robotiky, například v projektu ponorky Technické univerzity v Liberci. Od programu se požaduje minimální datová náročnost a odezva v reálném čase. Především rychlost je hlavním problémem výše zmiňovaných programů. Při ovládání robota jsou nutné okamžité reakce. Pokyny obsluhy nemohou být zpožděné o desetiny až celé sekundy. Odpověď na položené otázky byla teda zřejmá.

Výsledkem bakalářského projektu byl program napsaný v C++/CLI, který je schopen posílat videostream ve formě jednotlivých snímků komprimovaných metodou JPEG mezi jedním serverem a klientem. V bakalářské práci bylo hlavním úkolem doplnit program o možnost připojení více klientů k serveru a naprogramovat mechanismus pro kompletní ovládání kamery z klientské strany. Dále navrhnout a implementovat přidělování práv k ovládání kamery mezi jednotlivými klienty.

V této zprávě se snažím o shrnutí teoretických znalostí z oblasti použitých technologií a úplný popis architektury a funkčnosti programu včetně výsledného testování efektivity při různých rozlišeních a stupních komprese.

2 Použité technologie

V této části bakalářské práce jsou popsány jednotlivé technologie a jejich význam pro vytvořený software.

2.1. DirectShow

2.1.1 Úvod do DirectShow

DirectShow je architektura pro zpracování multimédií na platformě Microsoft Windows. Účelem architektury je zjednodušení a vyřešení následujících problémů, které ze zpracování multimédií vyplývají: [1]

- Zpracování multimédií je datově velice náročné a často je potřeba zpracovávat data v reálném čase.
- Ačkoliv se audio a video zpracovává jako oddělené datové toky, je potřeba, aby byly přesně synchronizované.
- Je potřeba pracovat s různými zdroji dat, např. s lokálními soubory, počítačovou sítí, televizním vysíláním nebo stejně jako v případě této práce, s webkamerami.
- Data mohou mít různý formát.
- Programátor předem neví, jaké zařízení bude dostupné na systému cílového uživatele.

DirectShow umožňuje ve vysoké kvalitě snímat a přehrávat video i zvuk v různých formátech, například MPEG, AVI, MP3 nebo WAV. Její hlavní úlohou je zjednodušit vývoj aplikací určených pro operační systém Windows, oddělením problémů se snímáním obrazu a zvuku z rozdílného hardwaru.

Architektura je založena na objektovém modelu COM[1] (Component Object Model) a je vytvořena v C++. DirectShow poskytuje většinu potřebných

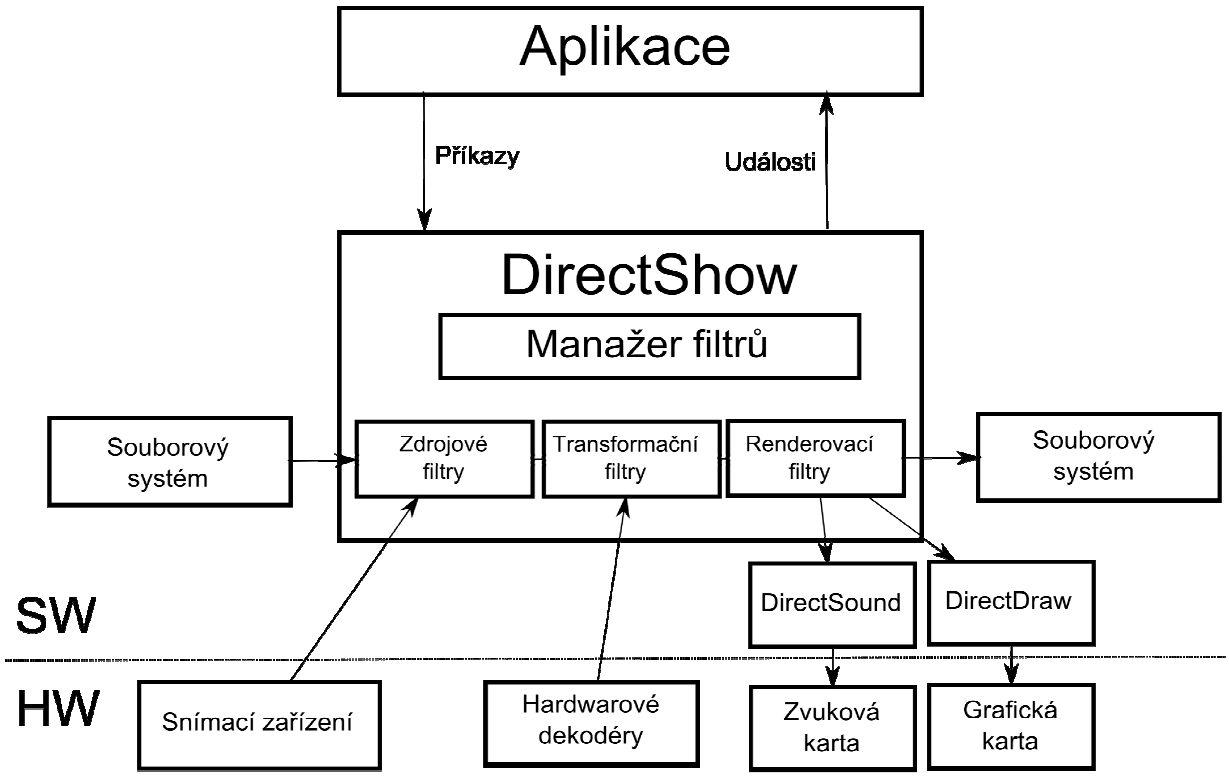
[1] „COM není jazyk ani soubor objektů, ale pouze standard či programovací technika - nezávislá na programovacím jazyku a operačním systému. COM určuje základní vlastnosti objektů a pravidla pro práci s nimi. Objekty v COM (tzv. *komponenty*) mezi sebou komunikují dle pevně stanoveného protokolu. COM je *binární standard* - knihovny psané dle COM mají i po zkompilování pevně danou strukturu, proto je lze použít i v jiném programovacím jazyce, než byly napsány.“ Citováno z [11].

komponent pro tvorbu aplikace, nicméně v případě nutnosti rozšíření programu je nutné implementovat nové komponenty jako COM objekty. Díky komponentám, které poskytuje, je možné vytvořit aplikace, jako jsou přehrávače videa, programy pro editaci videa, konvertory nebo obrazové procesory.

V tomto programu je DirectShow použita jako rozhraní pro snímání obrazu z webkamery. Umožňuje přístup ke každému jednotlivému snímku videostreamu jako k bitmapě uložené v podobě pole znaků (datový typ char) s barevnými RGB kanály v pořadí B, G, R, B, G, R...

2.1.2 Popis architektury

Architekturu a funkčnost DirectShow ilustruje následující obrázek.



Obrázek 1 - schéma komunikace architektury DirectShow

Aby bylo rozhraní co nejvíce variabilní a zvládalo zpracovat co největší množství různých zařízení a formátů, je složené z jednotlivých modulů

implementovaných jako COM objekty. Těmto modulům se zde říká filtry a jsou základním stavebním prvkem DirectShow.

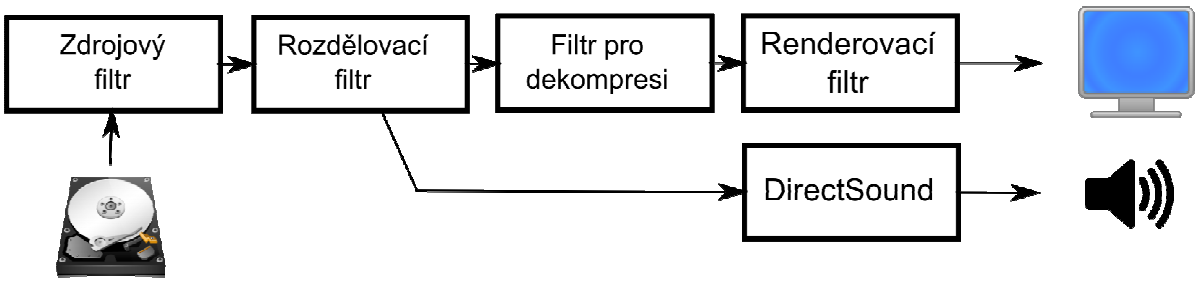
2.1.3 DirectShow Filtry

Rozhraní DirectShow poskytuje celou řadu standardních filtrů pro zpracování multimédií. Každý filtr je reprezentován COM objektem a stará se o jednu specifickou funkci. Filtr musí implementovat určité rozhraní a metody tak, aby s ním ostatní filtry mohly nezávisle spolupracovat. V rámci různých verzí se navíc tato rozhraní nemění nebo se pouze rozšiřují. Díky tomu se může programátor na funkce spolehnout, protože má jistotu, že budou funkce přítomné i v budoucích verzích. Vzhledem k tomu, že jsou filtry nezávislé, lze je za sebe skládat dle libosti a vytvořit tak celý řetězec pro zpracování obrazu.

Na následujícím příkladu lze ilustrovat, jaké filtry je potřeba použít pro přehrání videa ve vlastní aplikaci:

- Zdrojový filtr (File source filtr) – načtení souboru jako streamu bytů
- Rozdělovač streamů (AVI splitter) – rozdělení dat na video (jednotlivé snímky) a audio
- AVI dekompresor – použití různých dekompresních filtrů pro různé typy komprese
- Renderovací filtr – filtr pro vykreslení obrazu na monitor
- DirectSound filtr – přehrání zvuku pomocí zvukové karty

Řetězení filtrů ilustruje následující obrázek:



Obrázek 2 - ilustrace řetězení filtrů v aplikaci

Spojení mezi jednotlivými filtry obstarávají COM objekty, které se nazývají piny. Filtry si mezi sebou předávají data pomocí těchto pinů. Celé této sadě filtrů se v DirectShow říká diagram filtrů.

Jednotlivé filtry můžeme rozdělit podle jejich činnosti do několika kategorií:

- Zdrojové filtry: filtry přes které vstupují veškerá data do diagramu. Data mohou pocházet z různých zdrojů, např. kamery, sítě, souboru apod. Každý zdrojový filtr ovládá zpracování jednoho typu zdroje.
- Transformační filtry: zpracovávají vstupní data a převádí je na výstupní streamy. Patří sem například kodéry a dekodéry.
- Renderovací filtry: jsou na konci diagramu a v aplikaci se starají o prezentaci dat uživateli. Videostream mohou vykreslovat na obrazovku. Zvuková data přehrají pomocí zvukové karty.
- Rozdělovací filtry: rozdělí složitější datové streamy pro jejich oddělené zpracování. Například při zpracování souboru AVI rozdělí data do oddělených streamů videa a zvuku.
- Spojovací filtry: provádí opačnou operaci než rozdělovací filtry. Spojují více streamů do jednoho hotového formátu.

2.1.4 Manažer filtrů

Manažer filtrů je COM objekt, který kontroluje jednotlivé filtry v diagramu filtrů. Je prostředníkem mezi vlastní aplikací a jednotlivými DirectShow filtry. Jeho hlavní funkce jsou následující:

Změny stavů a jejich řízení

Filtry procházejí jednotlivými stavy v určitém pořadí. Při práci s filtry nepředáváte příkazy ke změně stavu přímo jednotlivým filtrům. Místo toho odešlete jeden příkaz manažeru a ten se postará o správnou distribuci příkazů mezi filtry.

Referenční hodiny

Všechny filtry používají jedny referenční hodiny pro přesnou synchronizaci streamů. Manažer vybere jako referenční hodiny hodinové pulsy zvukové karty nebo systémové hodiny.

Události diagramu

Manažer používá frontu událostí pro informování aplikace o událostech, které nastaly v jednotlivých filtrech. Tento mechanismus je podobný jako mechanismus zpracování zpráv v systémech Microsoft Windows (Windows Message Loop).

Metody pro stavbu diagramu

Manažer poskytuje řadu metod pro přidávání, odebírání a propojování filtrů v diagramu. Diagram tak lze s pomocí manažeru za běhu upravovat.

Manažer filtrů naproti tomu neřídí přenos dat mezi filtry. Filtry si je předávají samy pomocí propojení v podobě pinů. Veškeré procesy zpracovávání – filtry, manažer – běží vždy asynchronně v odděleném vlákně. Při potřebě synchronizace určitých činností je nutné použít událostí nebo nastavování uživatelských příznaků.

2.1.5 Windows Driver Model a jeho využití v DirectShow

Windows Driver Model je framework pro tvorbu ovladačů zařízení v systémech Microsoft Windows od verzí Windows 98 a 2000. Ovladače WDM jsou rozvrstveny do komplexní hierarchie, kde jednotlivé vrstvy komunikují pomocí struktur IRPs (I/0 request packets). WDM bylo vyvinuto za účelem zjednodušení tvorby ovladačů pro systém Microsoft Windows.

Základem architektury je sada na hardwaru nezávislých ovladačů, které se nazývají Class Drivers. Tyto ovladače poskytují základní funkcionalitu bez závislosti na konkrétním zařízení a umožňují komunikaci mezi jednotlivými vrstvami. Výrobce poskytne tzv. minidrivery – ovladače specifické ke konkrétnímu zařízení a jeho funkcím. Ve většině případů volá minidriver funkce Class Driveru.

V DirectShow jsou WDM ovladače použity pro přístup k zařízení. O zpracování video-zařízení se stará WDM Video Capture filtr. Software automaticky rozpozná všechna nainstalovaná zařízení pro snímání videa a dokáže jejich seznam enumerovat pro výběr zařízení. Díky využití ovladačů lze zpracovat jednotlivé módy kamery a její veškeré funkce a zobrazit je ve formuláři aplikace.

2.2. JPEG komprese

JPEG se řadí mezi ztrátové komprese a alespoň v oblasti fotografie je pravděpodobně nejpoužívanější kompresí obrazu na světě. Komprese využívá nedokonalostí lidského oka, které je citlivé především na jasovou složku obrázku a méně citlivé na změny v barvě. Lidské oko je navíc schopno rozeznat pouze omezené množství detailů. Při vhodném nastavení komprese tedy lidské oko nemusí vůbec rozpoznat změnu v kvalitě obrazu, protože došlo k odstranění pouze takových detailů, které by v nekomprimovaném obrázku stejně nevidělo. Díky těmto vlastnostem je komprese vhodná pro použití na reálné obrazy s velkým množství jemných detailů. Vzhledem k účinnosti komprese a dostupnosti open-source[2] knihoven byla komprese použita i v tomto softwaru.

JPEG komprese má několik základních kroků:

1. Převod barev z RGB do YC_bC_r
2. Podvzorkování chrominančních kanálů
3. Dopředná diskrétní kosinová transformace (DCT)
4. Kvantování koeficientů DCT pomocí kvantizačních tabulek
5. Kódování DCT koeficientů
6. Uložení do souboru

2.2.1 Převod barev z RGB do YC_bC_r

Prvním krokem, který se při kódování provede, je převedení obrazových dat z barevného prostoru RGB do barevného prostoru YC_bC_r. Zde se namísto

[2] Open-source = otevřený kód, programy nebo knihovny je dovoleno za dodržení určitých podmínek využívat ve vlastních programech

uchovávání jednotlivých barevných kanálů používá zachovávání jasové hodnoty Y a dvou chrominančních kanálů – červeného C_r a modrého C_b. Výhoda tohoto modelu spočívá ve vysoké citlivosti oka na jasovou hodnotu, ale nízké citlivosti na barevnou informaci. Díky tomu lze chrominanční kanály uchovávat v méně úrovních a zkomprimovat je s větší ztrátou, než kdybychom použili tři rovnocenné barevné kanály v modelu RGB. Chrominanční kanály jsou navíc v YC_bC_r uchovávány rozdílově. Hodnoty barevných kanálů jsou tedy rozdílem hodnoty Y a skutečné hodnoty C_b nebo C_r. Tento prostor byl primárně navržen pro přenos TV a video signálu.

Pro převod z RGB do YC_bC_r existují přesně dané vzorce, které se mohou lišit pouze drobnými změnami v koeficientech. V softwaru je použit následující vzorec:

$$Y = 0.29900 * R + 0.58700 * G + 0.11400 * B$$

$$C_b = -0.16874 * R - 0.33126 * G + 0.50000 * B + 128$$

$$C_r = 0.50000 * R - 0.41869 * G - 0.08131 * B + 128$$

2.2.2 Podvzorkování barevných kanálů

Jak již bylo zmíněno výše, lidské oko je méně citlivé na informace o barevnosti, proto si můžeme dovolit vyhradit více paměti na kanál Y a méně na barevné kanály. Na kanál Y je tedy vyhrazen celý jeden byte pro každý pixel. Naproti tomu jednotlivé barevné kanály jsou počítány jako průměr z bloku 2×2 pixely. Pro takový blok 4 pixelů se pak použije tento průměr uložený v jednom bytu a tím se ušetří za každý barevný kanál 3 byty. Pro celý blok 2×2 pixely, který by zabíral 12 bytů, už se tedy jedná o úsporu 50%. V této fázi začíná být celý proces ztrátový, neboť už neuchováváme přesnou informaci ke každému pixelu zvlášť.

2.2.3 Dopředná diskrétní kosinová transformace (DCT)

Dalším stěžejním krokem je diskrétní kosinová transformace, která slouží pro převod vzorků z časové oblasti do oblasti frekvenční a provádí se postupně na blocích 8×8 nebo 16×16 pixelů. Těmto blokům se v JPEG kompresi říká makrobloky. Výsledkem procesu je matice 8×8 (16×16) hodnot. První hodnota je stejnosměrná složka a ostatní hodnoty jsou

koeficienty kosinových složek signálu. Dále je třeba zmínit, že hodnoty v matici ještě před provedením transformace posunou odečtením hodnoty 128 do rovnoměrného rozložení okolo nuly.

Samotné převedení bloku pixelů pomocí DCT není nijak ztrátové, neboť se lze ke stejnému výsledku dostat beze ztráty informace pomocí zpětné kosinové transformace. Výhodou ale je, že v matici 8×8 hodnot se ty nejvýznamnější koeficienty (stejnosměrná složka a amplitudy malých frekvencí) nacházejí v levém horním rohu a jejich důležitost se směrem doprava a dolů snižuje. V matici se navíc s rostoucí vzdáleností od zmiňovaného rohu nacházejí velmi malá čísla, v mnoha případech i nuly. Proto tudíž je toto zakódování velmi vhodné pro účinnou komprimaci.

2.2.4 Kvantování koeficientů DCT

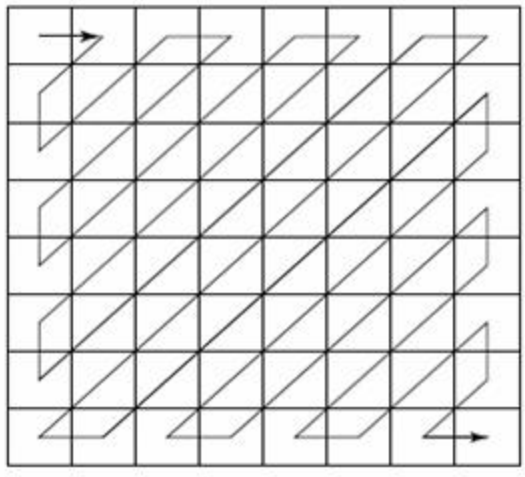
V následujícím kroku dojde k nejvýznamnější úspoře paměti. Matice hodnot získané v předchozím kroku se v tomto kroku vydělí tzv. kvantizační tabulkou. Ta má v levém horním rohu nízké hodnoty, které se směrem doprava a dolů zvyšují. Hodnoty závisí na konkrétním nastavení kvality komprese – čím vyšší čísla v tabulce, tím více koeficientů se vynuluje a sníží se kvalita obrazu. Po vydělení touto tabulkou se velká část koeficientů DCT (ty nejméně významné) změní na nulu (dělí se celočíselně). Z dat se tak odstraňují hodnoty od nejméně významných, až po ty nejdůležitější.

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Obrázek 3 - příklad výsledné matice hodnot po provedení DCT a vynásobení kvantizační maticí

2.2.5 Kódování DCT koeficientů

V předchozím kroku došlo k vynulování velké části hodnot. Matice je tak připravena pro efektivní zakódování. Hodnoty matice se zpracovávají jako jednorozměrný řetězec metodou „cik-cak“.



Obrázek 4 - ilustrace metody cik-cak (převzato z [5])

Díky tomu se nejprve zpracují hodnoty v levém horním rohu, které jsou nenulové, a nakonec zbude dlouhý řetězec nul. Tato část řetězce tvořená samými nulami se zakóduje speciálním kódem EOB. Výsledkem je řetězec bajtů, na který je uplatněna Huffmanova metoda.

Huffmanovo kódování je bezeztrátová metoda komprese založená na nahrazování původních hodnot sekvencemi bitů o různých délkách, přičemž nejčastěji opakující se hodnoty se zakódují nejkratší sekvencí bitů.

2.2.6 Uložení komprimovaných dat

Posledním krokem je uložení dat do streamu/souboru. Při uložení do souboru může být k datům přidána informační hlavička, která může obsahovat například i ve fotografii velmi používaný EXIF[3]. Pro tento software je ale jedním z kritérií také nízká datová náročnost přenosu. Do hlavičky se proto zapisuje pouze to nejdůležitější – rozměry obrazu a délka datové části.

[3] Jde o informační soubor připojitelný do obrazového souboru JPEG, obsahující informace o vzniku snímku. [4]

2.3. TCP

Protokol TCP (Transmission Control Protocol) je síťovým protokolem postaveným nad protokolem IP (Internet Protocol) a představuje v modelu OSI[4] transportní vrstvu. Tento protokol vytváří spojení mezi dvěma počítači a umožňuje mezi nimi předávat data. Navíc zaručuje doručení každého paketu i jejich správné pořadí. Ke každému paketu přidává pořadové číslo a kontrolní součet. Ten je vypočten i na straně příjemce a v případě, že se shoduje, odešle potvrzovací zprávu s číslem potvrzovaného paketu zpět k odesílateli. Pokud se odesílateli nevrátí do určitého času potvrzovací zpráva, je tento paket odeslán znovu. Jednotlivé běžící programy (i více instancí jednoho programu), pro které jsou data určena, jsou odlišeny přiděleným nebo zvoleným portem (v rozsahu 0-65535).

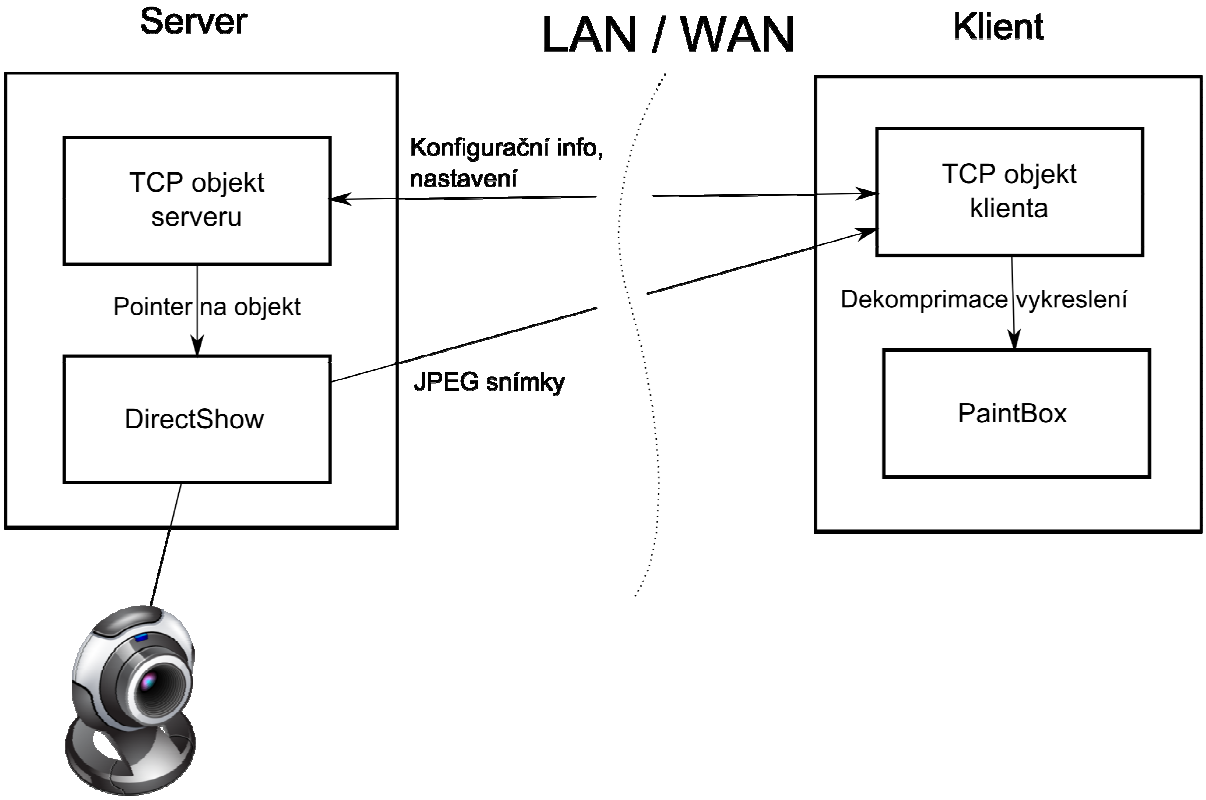
Kvůli potvrzování došlých dat se protokol v běžné praxi nepoužívá pro aplikace vyžadující rychlé zpracovávání dat, jako je přenos videa. Tyto programy používají většinou protokoly bez potvrzování, neboť není v zájmu rychlosti potřeba kontrolovat neporušenost dat jednotlivých snímků a data se posílají pouze jedním směrem ke klientovi. Na protokolu TCP byla vytvořena komponenta pro posílání dat po síti poskytnutá vedoucím práce. Daná komponenta byla použita i v tomto softwaru.

[4] Referenční model vypracovaný organizací ISO a schválený roku 1984. Jeho úlohou je standardizace norem v oblasti počítačových sítí. [7]

3 Popis vytvořeného softwaru

Software byl realizován v prostředí Microsoft Visual Studio v jazyce CLI/C++ a jeho základ byl vytvořen během bakalářského projektu. Tato kapitola popisuje kompletní architekturu a funkčnost aplikace, která byla nazvána CameraRobot.

Pro software byl zvolen model klient-server. Součástí zadání bylo i naprogramování možnosti připojení více klientů k jednomu serveru a také možnost ovládat z klientské strany kameru. Rozšířením zadání bylo navržení mechanismu pro přidělování práv klientům k nastavení kamery.



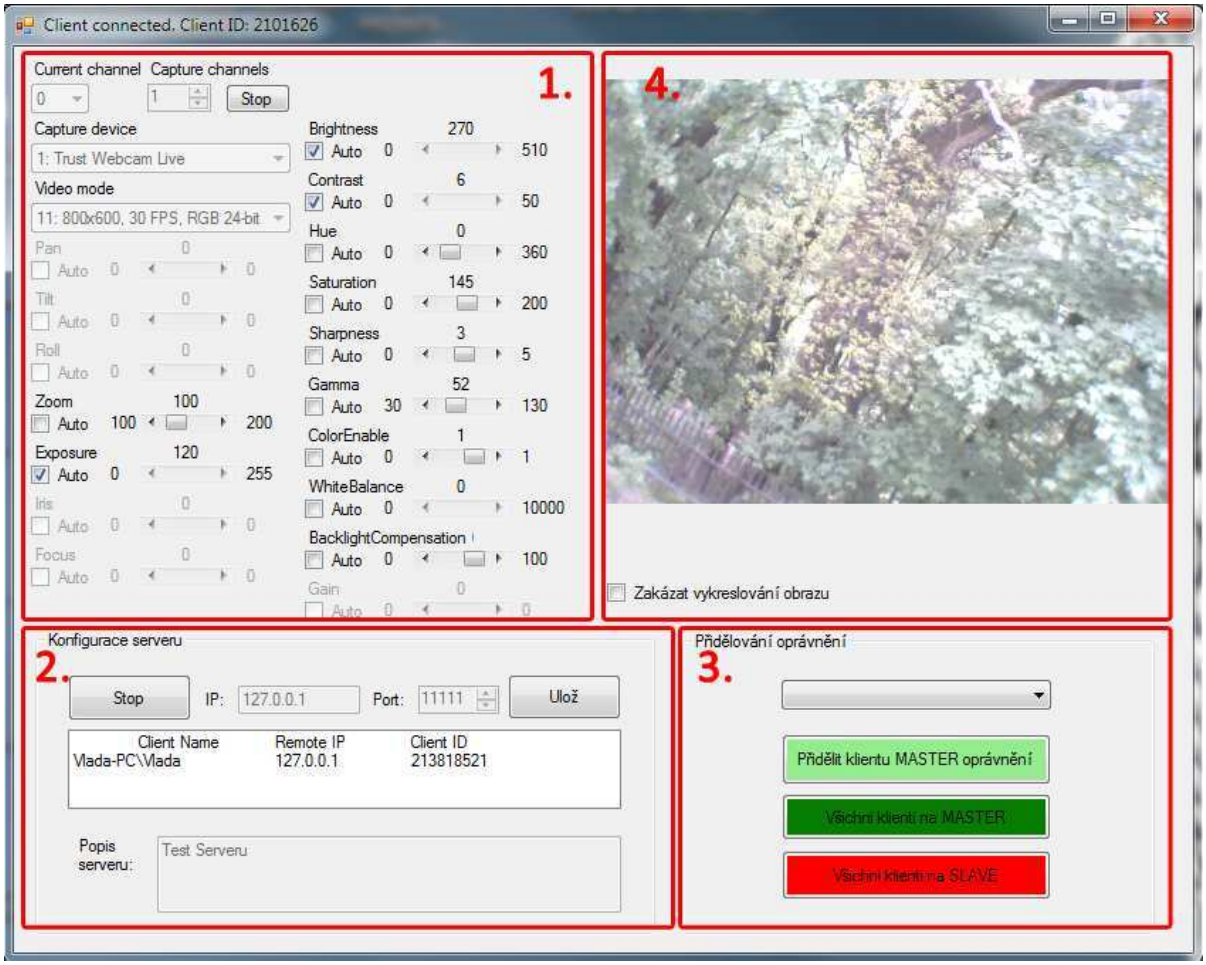
Obrázek 5 - základní schéma aplikace

Na PC se spuštěným serverem je připojená kamera. Obraz z kamery se zpracovává pomocí DirectShow a v případě vyžádání se pomocí vlastního protokolu odesílá příslušnému klientovi. V následujících částech budou podrobně popsány jednotlivé komponenty.

Jako server byla logicky zvolena PC stanice s připojenou kamerou, která bude odesílat video jednotlivým připojeným klientům. Server stačí tedy na PC spustit a vše lze ovládat z připojeného klienta.

3.1. Formulář serveru

Na následujícím obrázku je náhled formuláře serveru. Jednotlivé části formuláře a jejich funkce jsou popsány níže.



Obrázek 6 - náhled formuláře serveru

Panel ovládání kamery (oblast číslo 1) je zobrazen pomocí DirectShow. Umožňuje vybrat jednu z připojených kamer a vybrat jeden z jejích režimů snímání. Pokud kamera umožňuje nastavení expozice, jasu, světlosti a další, zobrazí se i ovládací prvky pro tyto parametry.

Panel s konfigurací serveru (oblast číslo 2) slouží pro nastavení konfigurace a spuštění serveru. Je třeba vyplnit vlastní IP adresu a port, na

kterém bude server naslouchat. Do pole popis serveru lze vypsat jakékoliv informace, například název serveru, popis umístění kamery apod. Tato zpráva se odešle ihned po připojení klienta a zobrazí se v jeho formuláři. V seznamu jsou vidět všichni připojení klienti včetně jejich ID a jmen.

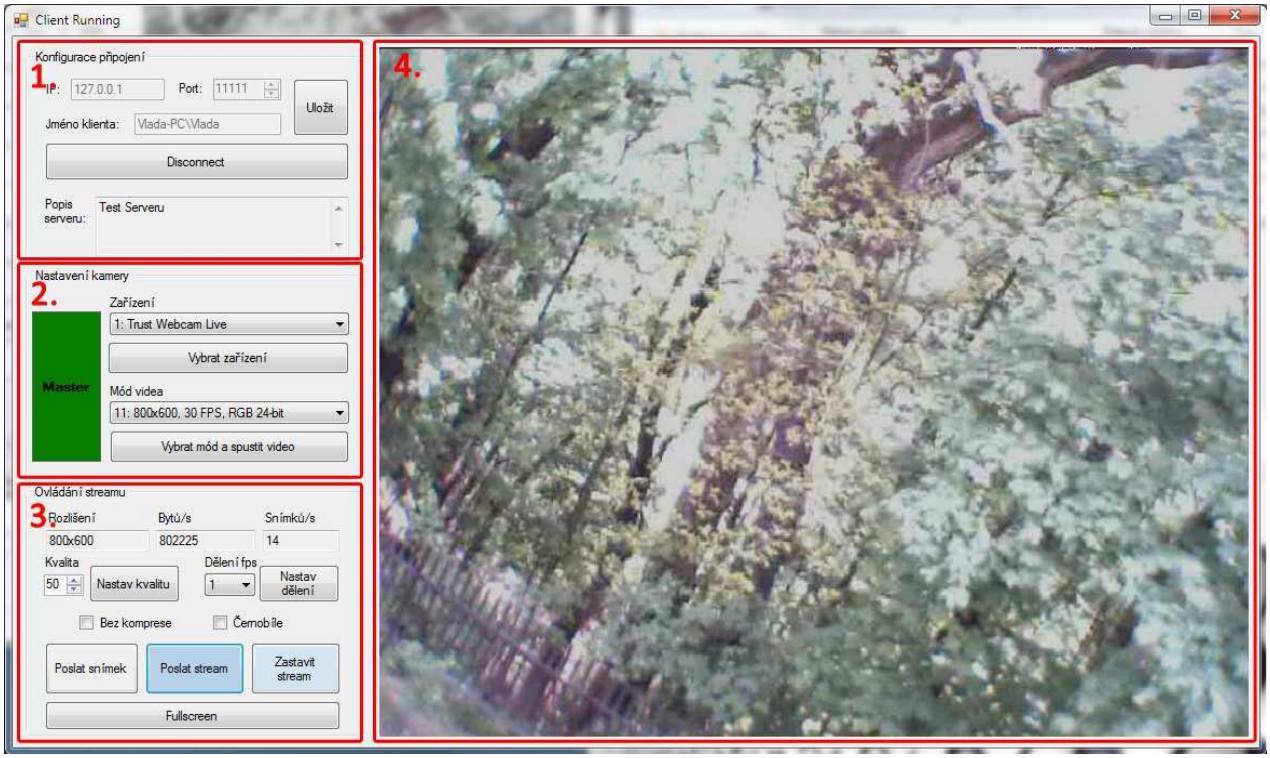
V panelu přidělování práv (oblast číslo 3) lze vybrat jednoho z klientů a tomu přidělit práva k ovládání kamery. Případně je možné odeslat práva všem připojeným klientům nebo všem práva odebrat.

V panelu s náhledem obrazu (oblast číslo 4) lze sledovat snímané video. Pro snížení datové náročnosti je možné vykreslování vypnout.

Všechna nastavení serveru včetně nastavení vykreslování nebo popisu serveru je možné uložit do souboru. Server si tento soubor po spuštění otevře a uložené informace načte.

3.2. Formulář klienta

Na následujícím obrázku je náhled formuláře klienta. Jednotlivé části formuláře a jejich funkce jsou popsány níže.



Obrázek 7 - náhled formuláře klienta

Panel konfigurace připojení (oblast číslo 1) slouží pro konfiguraci klienta a jeho připojení k serveru. Je potřeba nastavit IP adresu serveru a port, na kterém čeká na připojení. V poli Jméno klienta je možno nastavit název klienta, pod kterým se bude zobrazovat na serveru. Defaultně se načte ze souboru, jinak se nastaví jméno právě přihlášeného uživatele ve Windows. Po připojení k serveru se jeho popis zobrazí v poli Popis serveru. Nastavení lze taktéž uložit do souboru stejně jako u serveru.

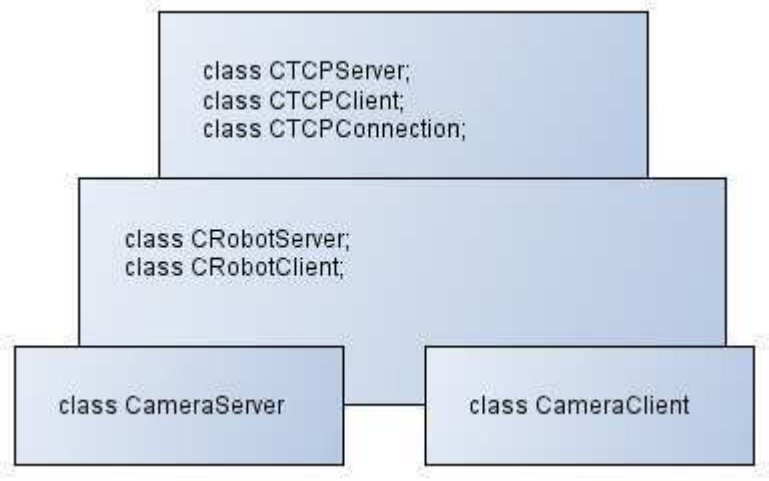
Panel nastavení kamery (oblast číslo 2) je aktivní pouze pokud má klient přidělená práva, jinak jsou prvky zakázané. V první nabídce lze vybrat kameru. Po nastavení kamery se objeví v druhé nabídce dostupné módy kamery. Po vybrání kamery se rovnou spustí video na serveru a je možné vyžádat posílání streamu.

Panel ovládání streamu (oblast číslo 3) slouží k nastavení streamu a jeho kvality. Klient si může nechat poslat jen jeden snímek nebo celý stream. V krocích po deseti lze nastavovat kvalitu videa. Při aktivaci volby černobílého obrazu se posílá jednokanálová černobílá bitmapa o velikosti 1/3 původní trojkanálové barevné bitmapy. Klient má dále možnost nechat si posílat nekomprimovaný videostream v plné kvalitě. V boxech se zobrazuje rozlišení aktuálně přijímaných snímků, aktuální datový tok a počet přijatých snímků za sekundu.

Největší část formuláře obsahuje vykreslovaný obraz (oblast číslo 4), který lze stisknutím tlačítka fullscreen nebo dvojklikem zvětšit na celou obrazovku.

3.3. Hierarchie tříd

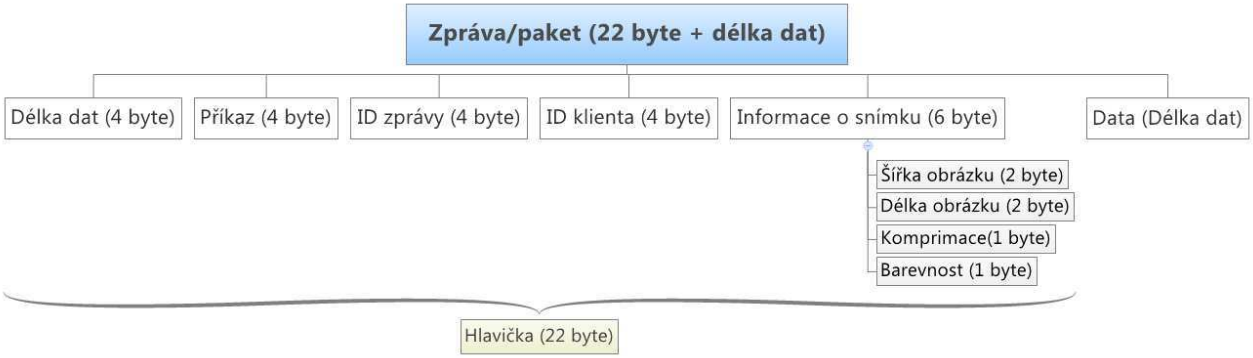
O základní TCP komunikaci se starají třídy `CTCPServer`, `CTCPClient`. Rozšířením těchto tříd jsou třídy `CRobotServer` a `CRobotClient`, které přidávají potvrzování zpráv a možnost jejich bufferování. O zpracování obrazu z webkamery se stará třída `CameraCustomDSVC`, která rozšiřuje základní funkce pro ovládání kamery třídy `DShowVCDotNet`.



Obrázek 8 - diagram hierarchie tříd

3.4. Protokol pro posílání zpráv

Datové zprávy, které si vyměňují klienti se serverem, se skládají z hlavičky a datového řetězce. Hlavička se kvůli nárůstu funkcionality zvětšila na 22 bytů. Velikost datové části je teoreticky omezená pouze rozsahem proměnné pro délku dat. Při použití datového typu integeru tedy 2,147,483,647 bytů.



Obrázek 9 - struktura datové zprávy

Význam jednotlivých hodnot částí je následující:

- Délka dat: Určuje délku datového řetězce. Informace nutná ke správnému přečtení všech dat. Pokud se posílá zpráva bez dat, je nastavena na 0.
- Příkaz: Označení příkazu nebo obsahu zprávy pomocí předdefinovaných konstant datového typu integer. (Například `VIDEO_FRAME` pro označení zprávy se snímkem apod. Kompletní seznam konstant je uveden v Příloze B)
- ID zprávy: jedinečné ID zprávy pro jejich potvrzování.
- ID klienta: Při spuštění klienta se vygeneruje jeho unikátní ID, které odešle serveru a od té doby se bude pomocí tohoto ID identifikovat.
- Informace o snímku: v JPEG kompresi se i při 100% kvalitě ztrácí určitá část informace. Proto bylo rozhodnuto, že má mít klient možnost nechat si poslat stream bez komprimace v plné kvalitě. Kvůli tomu bylo nutné přidat do hlavičky informace o snímku pro jeho správné rozbalení u klienta.
    - Šířka a délka snímku: pro správné vykreslení do PictureBoxu
    - Boolean hodnota komprimace: značí, zda se jedná o bitmapu nebo JPEG – podle toho se na klientské straně použije metoda pro dekomprimaci, nebo se snímek rovnou vykreslí.
    - Boolean hodnota barevnosti: při převedení do odstínů šedé se spočítá hodnota ze všech třech kanálů dle následujícího vzorce:

      ```
      PIX = 0.3*R + 0.59*G + 0.11*B
      ```

      Tato hodnota se běžně ukládá do všech třech kanálů. V tomto softwaru se ale uloží jen jednou a výsledkem je nekomprimovaný buffer o velikosti 1/3 původního barevného bufferu.

Zprávy, které si mezi sebou server a klienti vyměňují, mají strukturu formulářových zpráv systému Windows, kde je naše vlastní hlavička uložena v parametru `WParam` a data uložena v parametru `LParam`.

3.5. Nastavení kamery a přidělování práv

Prvním a nejdůležitějším bodem zadání bylo ovládání kamery ze strany klienta. Do této doby bylo nutné nastavit kameru na serveru a klient pouze pasivně přijímal data a vykresloval je. Bylo nutné navrhnout systém, jakým si budou jednotlivé stanice předávat informace. S nastavováním kamery souvisí i model přidělování oprávnění k nastavení kamery.

3.5.1 Mechanismus přidělování práv

Každý klient se spustí vždy bez práv k ovládání kamery, tedy jako SLAVE. Pokud je prvním klientem připojeným k serveru, odešlou se mu práva k nastavení. Každý další klient se spustí jako SLAVE a zůstane bez práv. Z formuláře serveru lze vybrat kteréhokoliv z připojených klientů a odeslat mu práva, stejně tak lze všem práva odebrat nebo je naopak všem přidělit. Pokaždé když se odpojí klient, kontroluje se, zda není jediným připojeným klientem. Pokud ano, odešlou se mu práva k nastavení kamery.

Vlastnictví práv je implementováno zakázáním nebo povolení ovládacích prvků (nabídek a tlačítek) na formuláři klienta. Vlastní odesílání práv je realizováno odesíláním zpráv ze serveru ke klientovi s příkazy `SET_CLIENT_MASTER` nebo `SET_CLIENT_SLAVE`. Každý klient se spouští jako SLAVE se zakázanými ovládacími prvky kamery. Odeslání práv k nastavení kamery ze serveru probíhá odesláním zprávy s příkazem `SET_CLIENT_MASTER`. Ve chvíli, kdy klient přijme takovou zprávu, povolí se ovládací prvky kamery a klient se stává MASTEREM. Pokud naopak od serveru obdrží zprávu s příkazem `SET_CLIENT_SLAVE`, ovládací prvky se mu zakážou a klient se stává SLAVEM.

3.5.2 Nastavení kamery

Vlastní přenesení obsahu nabídky výběru kamery a módu kamery probíhá hned po připojení klienta k serveru. Nejprve se odešle obsah nabídky výběru kamery. Teprve po potvrzení výběru kamery klientem lze přenést obsah nabídky pro výběr módu zvolené kamery. Po výběru módu už se na serveru spustí snímání obrazu. Pokud ale snímání probíhá už při

připojení klienta, přenese se obsah nabídek rovnou ke klientovi a nastaví se i skutečně vybrané volby. Klient tak ví, jaká kamera a jaký mód je aktuálně nastavený na serveru. Jakákoliv změna se vždy distribuuje mezi všechny klienty. Pokud tedy jeden změní nastavení, ostatní tuto změnu ihned zaregistrují. Změnu zaregistrují i přesto, že jsou právě SLAVE. Obsah nabídek a vybraná hodnota se aktualizují neustále, SLAVE pouze nemá možnost hodnoty měnit a odesílat je serveru.

Přenesení obsahu nabídek proběhne nejprve odeslání zprávy ze serveru ke klientovi s příkazem k vymazání obsahu (`DEVICE_LIST_CLEAR` pro vymazání položek výběru kamery, `DEVICE_LIST_MODES_CLEAR` pro vymazání položek dostupných módů). Poté se vždy jednotlivé položky příslušné nabídky zabalí do samostatné zprávy a odešlou se klientovi. Na straně klienta se zprávy postupně přijmou a ve stejném pořadí se po jedné vloží do nabídky.

Přenesení vybrané hodnoty z klienta na server nebo i obráceně probíhá jednoduše odesláním indexu vybrané položky. Obsah nabídek je na obou stranách stejný a položky jsou i ve stejném pořadí, mezi klienty a serverem lze tedy synchronizovat pouze vybraný index.

3.6. Server – algoritmy a struktury

3.6.1 Připojení více klientů

Na straně serveru bylo potřeba vyřešit hned několik problémů. Prvním z nich bylo připojení více klientů. Jednotlivá spojení se na serveru ukládají pomocí třídy `CTCPConnection` a odkazují na sebe pomocí cirkulárních pointerů `prev` a `next`. V původním modelu probíhala komunikace a odesílání zpráv pouze na klienta na prvním místě. Z tohoto důvodu bylo doprogramováno hned několik metod:

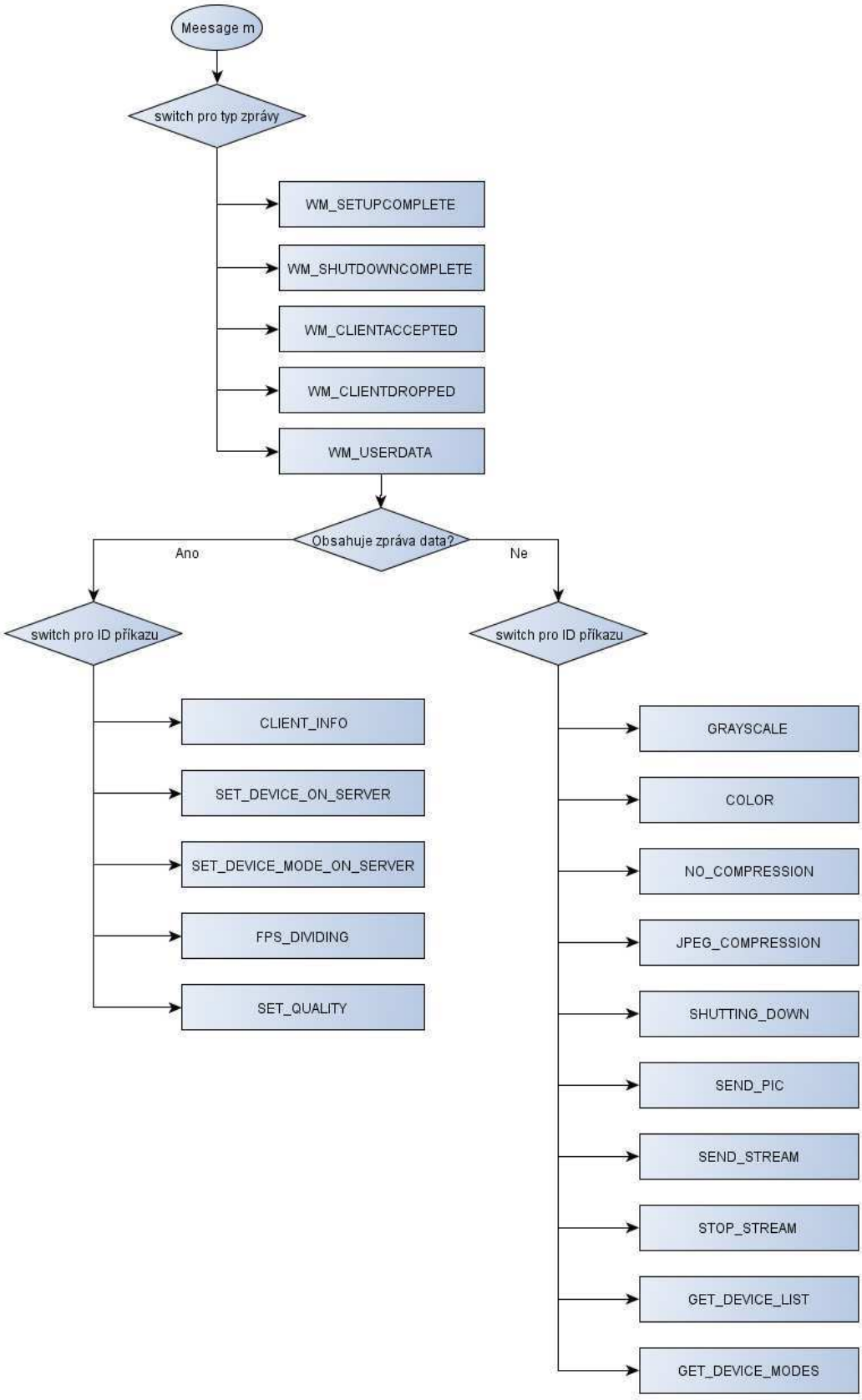
- **SendToAllClients** – projde postupně všechna TCP spojení a na každé odešle připravenou zprávu
- **SendToOneClient** – odešle zprávu na konkrétního klienta podle jeho pořadí v seznamu

- **SendToSpecificClient** – odešle zprávu na jedno konkrétní TCP spojení

3.6.2 Zpracování přijatých zpráv

Veškeré zprávy, které server přijme, zpracovává nejprve vlákno serveru ve vrstvě TCP komunikace. Zde se zpracovávají přijaté zprávy, které souvisí s TCP spojením, všechny ostatní zprávy se předávají nadřazenému vláknu – formuláři serveru. Mezi tyto zprávy patří veškeré uživatelské zprávy ohledně nastavení kamery, nastavení streamu atd. Zprávy se na formuláři zpracovávají ve virtuálně přepsané metodě `WndProc`.

Podle ID uložené v proměnné `msg` se nejprve rozliší typ formulářové zprávy. V případě, že dorazí zpráva s ID `WM_USERDATA`, jedná se o uživatelskou zprávu odeslanou některým z připojených klientů. V parametru `LParam` se nacházejí přijatá data a v parametru `WParam` je uložena naše vlastní hlavička zprávy (Více v kapitole 3.4. Protokol pro posílání zpráv). Podle ID zprávy v naší hlavičce a přítomnosti nebo absenci dat se zpracování dále větví. V případě, že nejsou data přítomna, vyhodnotí se zpráva pouze podle příkazu v hlavičce. Například `GRAYSCALE` znamená, že se má odesilateli zprávy posílat černobílý stream apod. Naopak u zpráv obsahujících data se datová část podle příkazu zpracuje. Zpráva s příkazem `SET_DEVICE_ON_SERVER` obsahuje v datové části integer s indexem vybrané kamery. Zpracování probíhá dle schématu uvedeného níže.



Obrázek 10 - schéma zpracování zpráv na formuláři serveru

3.6.3 Ukládání klientů – třída ClientInfo

Jelikož bylo rozhodnuto o vytvoření modelu, ve kterém si každý klient může nezávisle na ostatních zvolit, zda chce stream přijímat, v jaké kvalitě ho chce přijímat a v jaké barevnosti, bylo nutné vytvořit strukturu pro uchovávání klientů a jejich nastavení. Byla vytvořena nová třída – `ClientInfo`, která je v podstatě jedním prvkem spojového seznamu. Pomocí třídy se budou do seznamu ukládat informace o připojených klientech a jejich specifická nastavení ohledně přijímaného streamu jako je kvalita, barevnost apod. Třída obsahuje následující atributy:

- **Jméno klienta:** Zde se ukládá jméno, které zadává klient ve formuláři. Slouží pro snadnou orientaci a identifikaci klientů na serveru.
- **ID klienta:** Unikátní identifikační číslo klienta, které má klient vygenerované ihned po spuštění. ID je součástí hlavičky každé zprávy a slouží pro identifikaci jejího odesilatele.
- **Kvalita:** Kvalita JPEG komprese u odesílaného streamu pro daného klienta. Po spuštění je automaticky nastavena na hodnotu 50.
- **Objekt JPEG encoderu (třída CMiniJpegEncoder):** objekt pro JPEG kompresi daného klienta. Kvalita komprese se zadává jako parametr konstruktoru třídy. Aby mohli mít jednotliví klienti nastavenou vlastní kvalitu, musí mít každý vytvořen vlastní objekt pro kompresi, jinak by se před kompresí každého snímku musel encoder znovu instanciovat.
- **Dělení snímkovací frekvence:** Kvůli snížení snímkovací frekvence byla naprogramována možnost dělení snímkovací frekvence. Pomocí této hodnoty je dělena snímkovací frekvence kamery a klientovi se odesílají pouze vybrané snímky.
- **Boolean změny kvality:** indikuje změnu nastavení kvality. Hodnota se kontroluje před kompresí snímku. Pokud je nastavena na true, vytvoří se nový encoder s nově nastavenou kvalitou.
- **Boolean odesílání videa:** nastavení klienta, zda požaduje odesílání celého videa.

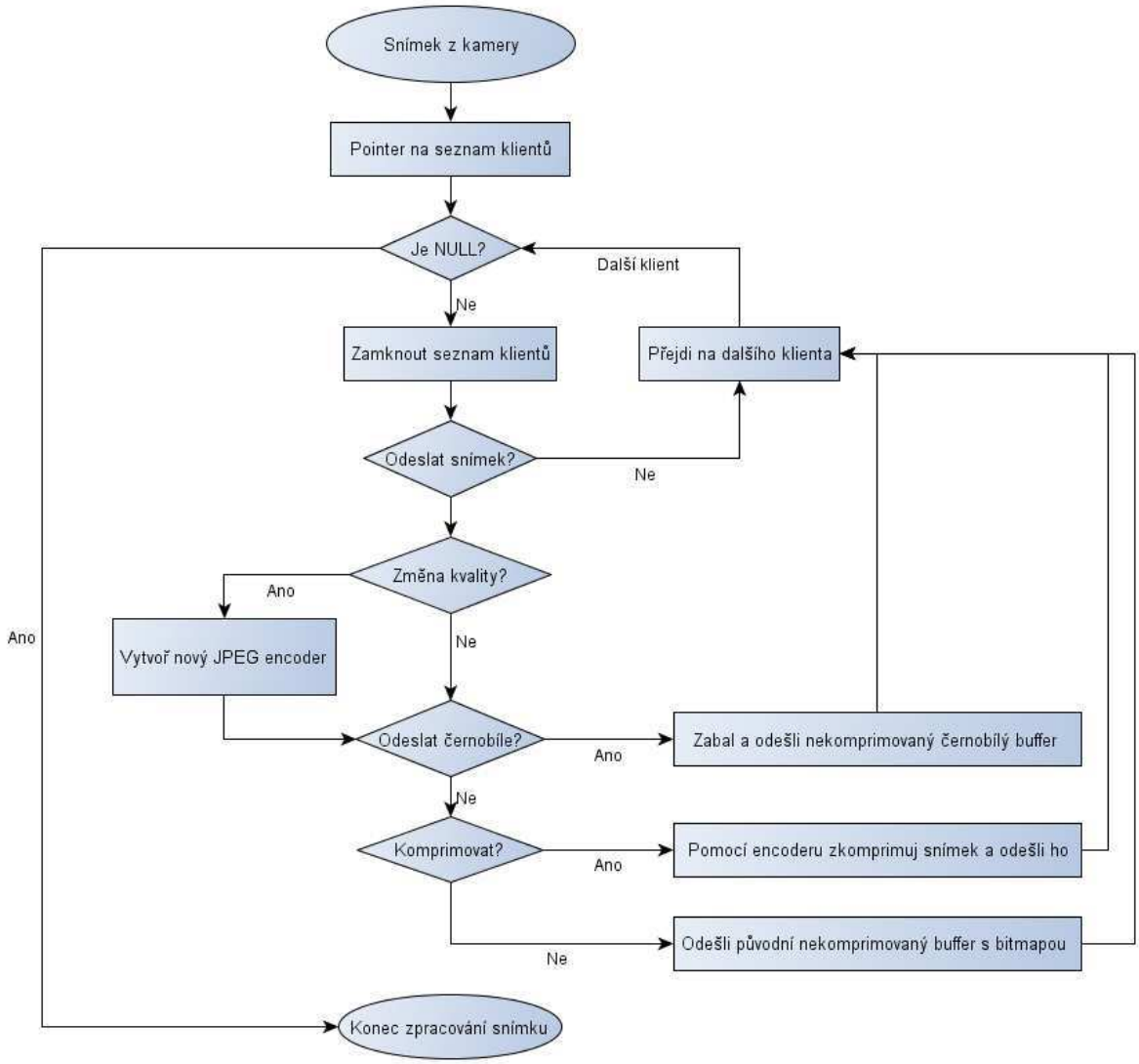
- **Boolean pro odeslání jednoho snímku:** Pokud chce klient odeslat jeden snímek, nastaví se tato hodnota na true. Po odeslání jednoho snímku se nastaví opět na false.
- **Boolean pro nastavení barevnosti:** indikuje, zda chce klient přijímat barevný nebo černobílý stream.
- **IP adresa:** IP adresa připojeného klienta. Slouží pouze pro výpisy v seznamech, není nutná pro vlastní běh aplikace.
- **Port:** Port, na kterém probíhá spojení s klientem. Slouží pouze pro výpisy v seznamech, není nutný pro vlastní běh aplikace.
- **Pointer na další instanci ClientInfo:** Pointer na další prvek v seznamu. Pokud jsme na konci seznamu, je nastaven na NULL.
- **Pointer na konkrétní TCP spojení s klientem (pointer na instanci CTCP Connection):** Každý klient má uložený přímo pointer na konkrétní spojení pro zajištění rychlého odesílání dat.

Uložení nového klienta do seznamu probíhá na serveru po přijetí zprávy s konstantou `CLIENT_INFO`. Obsahem této zprávy je ID číslo klienta, jméno, IP adresa a port v textovém řetězci, kde jsou položky odděleny řetězcem „`||`“. Ostatní položky se nemusí přenášet, ale nastaví se automaticky (kvalita je 50, neposílají se žádné snímky, černobílá škála vypnutá). Zprávu `CLIENT_INFO` odesílá klient ihned po obdržení zprávy od serveru s příkazem `WM_CONNECTED`.

3.6.4 Zpracování a odesílání snímků kamery

Základní snímání obrazu z webkamery je naprogramováno ve třídě `DShowVCDotNet`. Tato třída se také stará o zobrazení všech ovládacích prvků kamery – výběr kamery, módu, nastavení světlosti atd. Za účelem zpracování obrazu a rozšíření základní funkcionality byla vytvořena nová třída `CameraCustomDSVC`, která je potomkem výše zmíněné `DShowVCDoNet`. Zpracování probíhá v callback metodě `OnFrameCaptured`, která se jak název napovídá, volá při každém jednotlivém zpracovaném snímku.

Při sejmutém snímku probíhá zpracování podle následujícího diagramu:



Obrázek 11 - diagram zpracování snímku na serveru

Při každém sejmutém snímku je potřeba projít celý seznam klientů a u každého zvlášť se rozhodovat, jak se snímek zpracuje. Nejprve je třeba zjistit, zda chce klient snímek vůbec přijímat. Pokud ano, musí se zkontrolovat, jestli si klient nezměnil nastavení kvality komprese. V tom případě se vytvoří nový objekt encoderu se správnou kvalitou. Poté se zpracování větví dle kvality a barevnosti. Pokud chce klient odeslat černobílý snímek, zabalí se a odešle snímek z černobílého bufferu. Chce-li klient přijímat snímky bez komprese, odešlou se data přijatá přímo z kamery

a jinak se data zkomprimují pomocí JPEG encoderu o vybrané kvalitě a odešlou se.

Zpracování černobílého bufferu probíhá neustále od spuštění snímání kamery a provádí se pomocí makra RGB2GRAY, které barvy převádí dle následujícího vzorce:

```
PIXEL = 0.3*R + 0.59*G + 0.11*B
```

Jelikož se při převádění do černobílé škály nastavují hodnoty všech třech barevných kanálů na stejnou hodnotu, může se odesílat, kvůli snížení přenášeného datového objemu, pouze třetina těchto dat. Pokud má tedy klient nastavený černobílý videostream, bude přijímat stream o třetinovém datovém toku oproti barevnému streamu v plné kvalitě bez komprese.

3.6.5 Dělení snímkovací frekvence

Při testování programu byl zjištěn problém se zpožďováním snímků. Přenos videa probíhá v reálném čase pouze za předpokladu, že jsou klient a server propojení dostatečně rychlou nevytíženou sítí, jinak se snímky začnou zpožďovat a nejsou na klientské straně vykreslovány plynule. Z toho důvodu byla do klienta naprogramována možnost dělit snímkovací frekvenci za účelem snížení objemu datového toku při zachování rozlišení. Klient si tak nastavením dělení snímkovací frekvence na hodnotu 3 může snížit frekvenci přijímaného videa z 30 fps na 10 fps, nastavením hodnoty na 5 dosáhne snížení z 30 na 6 fps apod.

Dělení je realizováno počítadlem snímků na serveru a doplněním hodnoty dělení do třídy ClientInfo. Při odesílání snímku se kontroluje, jestli je hodnota počítadla beze zbytku dělitelná hodnotou, kterou má klient nastavenou jako dělení. Pokud není, snímek se klientovi vůbec neodešle. Takto je zaručeno, že klient dostává vždy pouze každý n-tý snímek, dle nastavení dělení na hodnotu n.

3.7. Klient – algoritmy a struktury

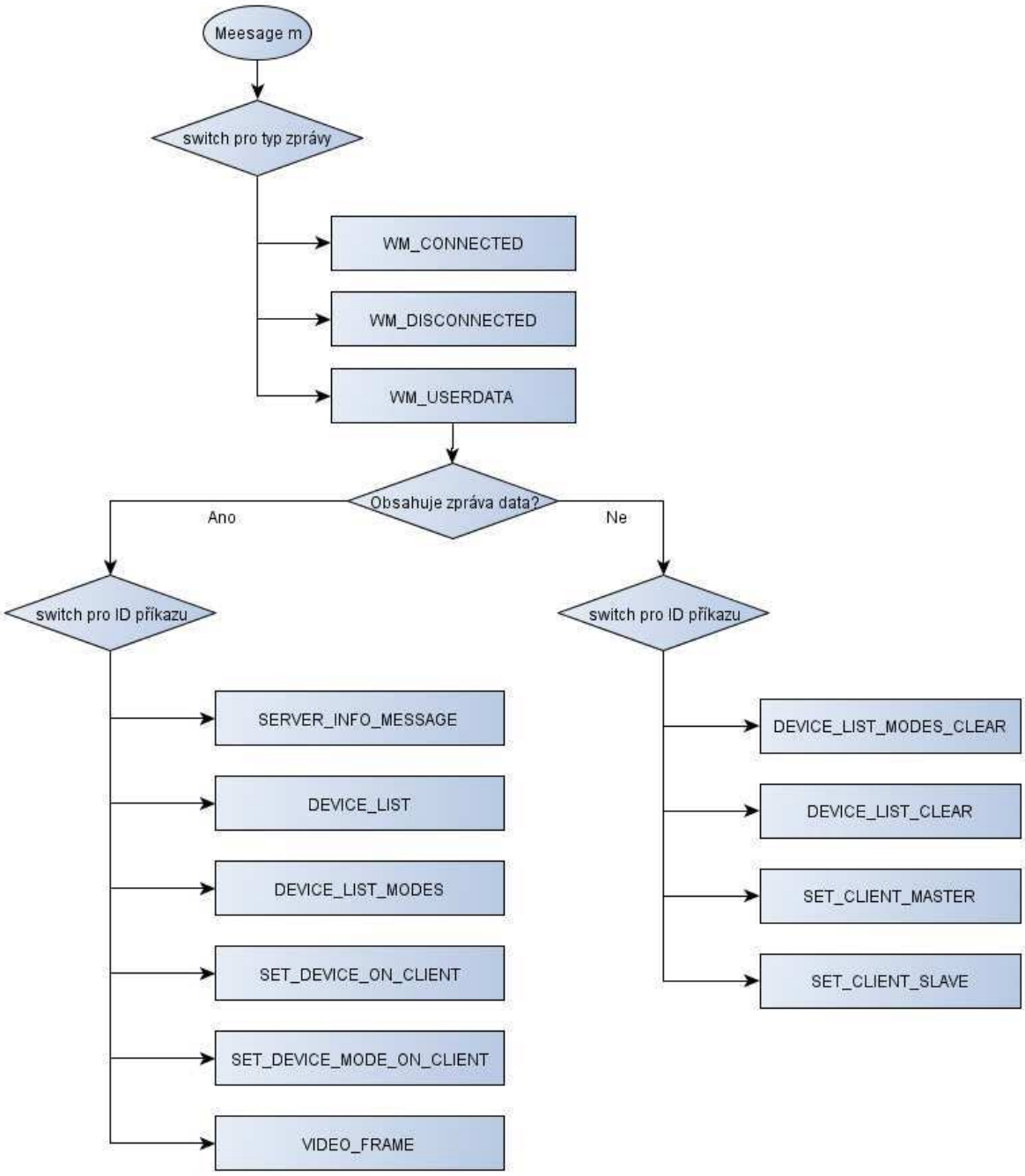
Klientská strana je ohledně datových struktur a algoritmů o mnoho jednodušší než strana serveru. Mezi nejdůležitější objekty patří především objekt `CMiniJpegDecoder`, který slouží pro dekompresi jednotlivých JPEG snímků.

3.7.1 Zpracování přijatých zpráv

Zprávy se u klienta zpracovávají obdobným způsobem jako na serveru. Nejprve se zpracují v nižší vrstvě pro TCP komunikaci a uživatelské zprávy se předávají formuláři, kde se zpracovávají ve virtuálně přepsané metodě `WndProc`.

Podle ID uložené v proměnné `msg` se nejprve rozliší typ formulářové zprávy. V případě, že dorazí zpráva typu `WM_USERDATA`, jedná se o uživatelskou zprávu odeslanou serverem. V parametru `LParam` se nacházejí přijatá data a v parametru `WParam` je uložena naše vlastní hlavička zprávy (Více v kapitole 3.4. Protokol pro posílání zpráv). Podle ID zprávy v naší hlavičce a přítomnosti nebo absenci dat se zpracování dále větví. V případě, že nejsou data přítomna, vyhodnotí se zpráva pouze podle příkazu. Například příkaz SET_CLIENT_MASTER znamená, že klient obdržel práva k nastavení kamery a že se mají na jeho formuláři povolit příslušné ovládací prvky. Naopak u zpráv obsahujících data se datová část podle příkazu zpracuje. Například zpráva s příkazem SERVER_INFO_MESSAGE obsahuje v datové části textový řetězec obsahující informace o serveru. Text se zpracuje a zobrazí na formuláři klienta.

Zpracování všech přijatých zpráv formuláře probíhá dle diagramu uvedeného na následující stránce.



Obrázek 12 - schéma zpracování zpráv na formuláři klienta

3.7.2 Vykreslování přijatých snímků

Pro vykreslování snímků je použita vlastní odvozená formulářová komponenta PaintBox. Komponentě stačí předat data a informace o snímku uložené ve struktuře `BITMAPINFOHEADER`. Vykreslování se provádí ve virtuálně přepsané metodě `OnPaint()` pomocí metody `StretchDIBits` mezi jejíž nejdůležitější parametry patří:

- Proměnná typu HDC (Handle to device context) – proměnná, pomocí které lze přistupovat k jednotlivým prvkům a překreslovat je
- Souřadnice a rozměry obdélníku ke zkopírování ze zdrojových dat
- Souřadnice a rozměry obdélníku kam se mají data zkopírovat
- Pointer na data (char *)
- Pointer na strukturu BITMAPINFOHEADER, ve které jsou uloženy důležité informace o obrázku, který chceme vykreslit
- Systémovou konstantou definovanou barevnost – v našem případě RGB obrázek – konstanta DIB_RGB_COLORS

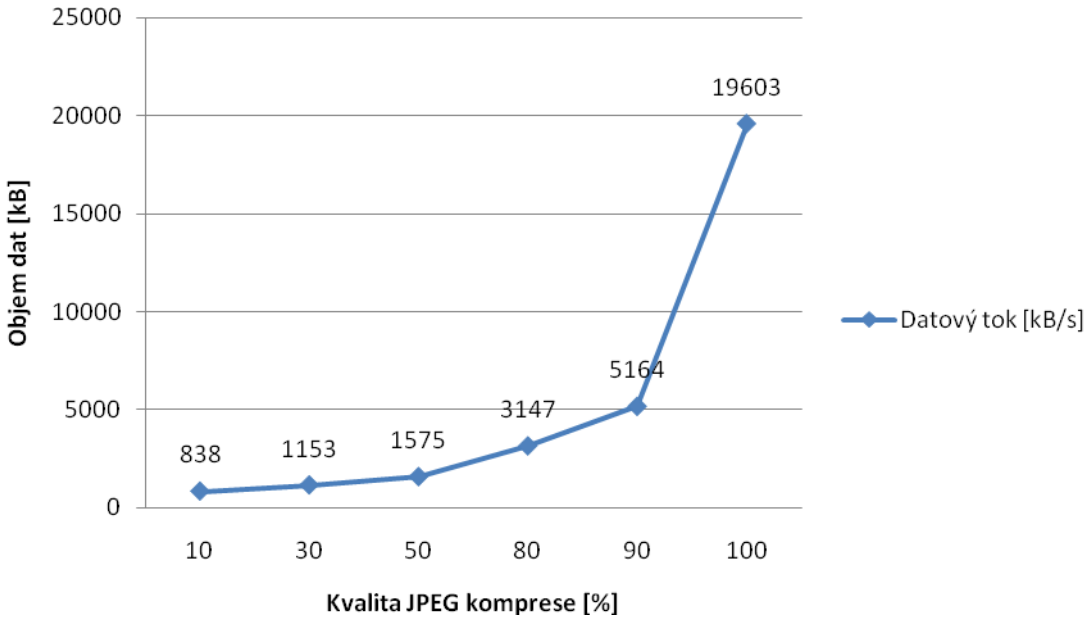
Tato metoda je pro aplikaci výhodná především v tom, že dokáže přijatý obrázek roztahovat nebo smršťovat podle velikosti komponenty na formuláři. Okno lze tedy podle potřeby zmenšit nebo zvětšit a přijímané video se bude přizpůsobovat velikosti okna. Před vykreslením se navíc vypočte poměr stran snímku. Obraz se tedy roztahuje nebo zmenšuje, ale zároveň se zachovávají proporce snímku.

Klient přijaté snímky identifikuje pomocí příkazu `VIDEO_FRAME` uloženého v hlavičce přijaté zprávy. Při zpracování z hlavičky rovněž přečte informace o velikosti snímku, kompresi a barevnosti. Podle toho se vykreslování dále větví. Pokud jsou přijatá data barevná a nekomprimovaná, vykreslí se přímo na PaintBox. Je-li v datech uložený komprimovaný barevný snímek, je potřeba snímek pomocí dekodéru dekomprimovat do pomocného bufferu, který se vykreslí. Pokud jde o data černobílá, musí se dekomprimovat překopírováním do nového bufferu, kde se každá hodnota původního snímku zkopíruje třikrát (do každého barevného kanálu) do dekomprimovaného bufferu.

4 Testování aplikace

4.1. Datová náročnost

Testování datové náročnosti bylo provedeno s využitím webkamery iSlim značky Genius, která poskytuje snímky o nejvyšším rozlišení 1280×1024 pixelů a snímkovací 30 fps. Měření bylo provedeno v pěti různých rozlišeních při různé kvalitě komprese. Pro všechna rozlišení a kvalitu byla použita stejná testovací scéna. Výsledky měření jsou shrnuty v následujících grafech.



Graf 1 – závislost objemu datového toku na kvalitě komprese při rozlišení 1280×1024 pixelů a snímkovací frekvenci 30fps

V plně kvalitě bez komprese, kde každý pixel obrázku zabírá 3 byty (1 na každý barevný kanál) by se datový tok vypočetl podle vzorce:

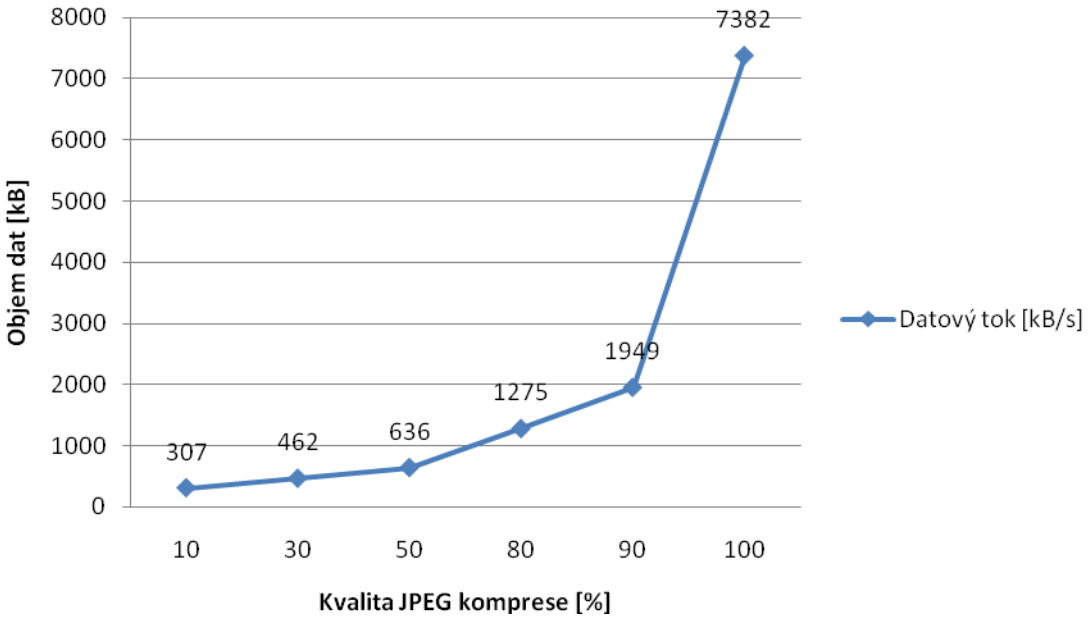
```
šířka × výška × počet barevných kanálů × fps
```

Pro dané rozlišení tedy:

```
1280 × 1024 × 3 × 30 = 117964800 bytů = 117964,8 kB
```

Bez komprese by byl datový tok šestkrát vyšší než videostream komprimovaný se stoprocentní kvalitou. Při použití nejnižší kvality lze

dosáhnout až stonásobné úspory, kvalita obrazu je už ale velmi degradována. Vhodným kompromisem mezi velikostí datového toku a kvalitou obrazu je kvalita 50-80 % kdy je úspora oproti nekomprimovanému toku přibližně padesátinásobná.



Graf 2 - závislost objemu datového toku na kvalitě komprese při rozlišení 800×600 pixelů a snímkovací frekvenci 30fps

Pro rozlišení 800×600 pixelů by byl nekomprimovaný datový tok 43200 kB. Použitím komprese lze dosáhnout obdobné úspory jako při vyšším rozlišení. Grafy pro ostatní rozlišení měly podobný průběh. Všechny pohromadě jsou uvedeny v Příloze A.

4.2. Zpoždění videa

Odezva mezi serverem a připojenými klienty je při vhodném nastavení videa z hlediska datového toku a průchodnosti sítě vzhledem k tomuto toku téměř nepostřehnutelná. Základní požadavek na software – odezva v reálném čase je tedy splněna. Záleží ale na rychlosti a vytížení sítě. Pokud se snažíme odeslat více dat, než je síť schopna posílat, začnou se data v síti hromadit a video se začne výrazně zpožďovat. Pro rychlost odezvy je tedy vždy nutné nastavit ve vhodném poměru rozlišení, kvalitu komprese a dělení snímkovací frekvence tak, aby je síť stihla odesílat.

5 Závěr

Software byl úspěšně realizován v prostředí Visual Studio 2008 v programovacím jazyce C++/CLI. Do programu byly implementovány nové datové struktury a algoritmy, díky kterým se může k jednomu serveru připojovat více klientů. Klienti si navíc mohou sami nezávisle na ostatních nastavovat kvalitu komprese, barevnost a vlastnosti video-streamu. Dále byl vypracován jednoduchý model přidělování práv k nastavení kamery, díky kterým má po připojení více klientů práva k nastavení kamery jen první z nich. Práva k nastavení kamery je navíc možné jednotlivým klientům dle libosti přidělovat z formuláře serveru. Také je lze všem přidělit nebo všem odebrat.

V průběhu testování náročnosti bylo zjištěno, že v dostatečně rychlé síti probíhá přenos opravdu v reálném čase. Zpoždění videa u klienta je oproti videu na serveru neznatelné. Viditelné zpoždění se začne projevovat až vlivem nízké průchodnosti sítě, ve které se snímky začnou opožďovat a docházejí poté s určitou prodlevou i po ukončení přenosu. Za tímto účelem bylo doprogramováno jednoduché dělení snímkovací frekvence, díky kterému si může klient jednoduše nastavit kombinaci rozlišení, kvality komprese a snímkovací frekvence tak, aby byl datový tok dostatečně nízký pro posílání videa v konkrétní síti.

Na závěr byla otestována datová náročnost videostreamu při různých rozlišeních a kvalitách komprese. Výsledky měření jsou zobrazeny pomocí přehledných grafů.

Seznam použité literatury

[1] *DirectShow System Overview* [online], 5.3. 2010 [cit 2010-04-19]. Dostupné na WWW: http://msdn.microsoft.com/en-us/library/dd375470%28v=VS.85%29.aspx.

[2] *Introduction to DirectShow Application Programming* [online], 5.3. 2011 [cit 2011-04-19]. Dostupné na WWW: http://msdn.microsoft.com/en-us/library/dd375470%28v=VS.85%29.aspx.

[3] TIŠNOVSKÝ,P. *Ztrátová komprese obrazových dat pomocí JPEG* [online], 14. 12. 2006 [cit 2011-04-25]. Dostupné na WWW: http://www.root.cz/clanky/ztratova-komprese-obrazovych-dat-pomoci-jpeg/.

[4] NEFF,O. *Co je to EXIF?* [online], [cit 2011-04-25]. Dostupné na WWW: http://www.digineff.cz/cojeto/ruzne/exif.html.

[5] *Image coding fundamentals* [online], 8. 5. 2007 [cit 2011-04-25]. Dostupné na WWW: http://videocodecs.blogspot.com/2007/05/image-coding-fundamentals_08.html.

[6] *TCP* [online], [cit 2011-04-25]. Dostupné na WWW: http://cs.wikipedia.org/wiki/TCP.

[7] *Referenční model ISO/OSI* [online], [cit 2011-04-25]. Dostupné na WWW: http://cs.wikipedia.org/wiki/Referen%C4%8Dn%C3%AD_model_ISO/OSI.

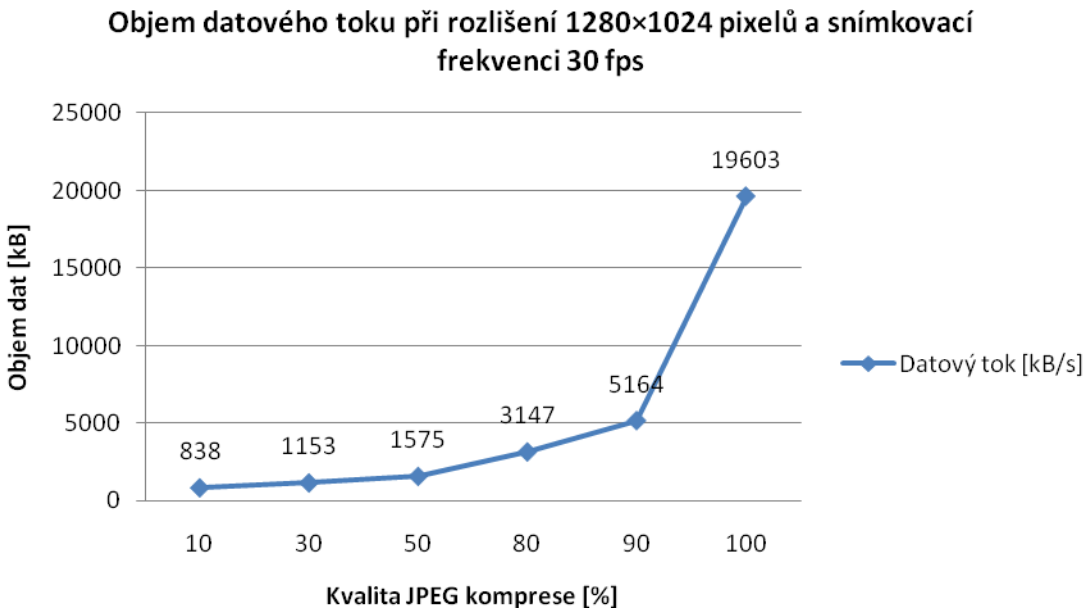
[8] LIN,T. *Classes to read and write BMP, JPEG and JPEG 2000* [online], 1. 2.2003 [cit 2011-05-25]. Dostupné na WWW: http://www.codeproject.com/KB/graphics/tonyjpeglib.aspx

[9] ŽABČÍK,T. *Softwarový interface a ovládání robotů katana,* Liberec, 2009. 32 s.

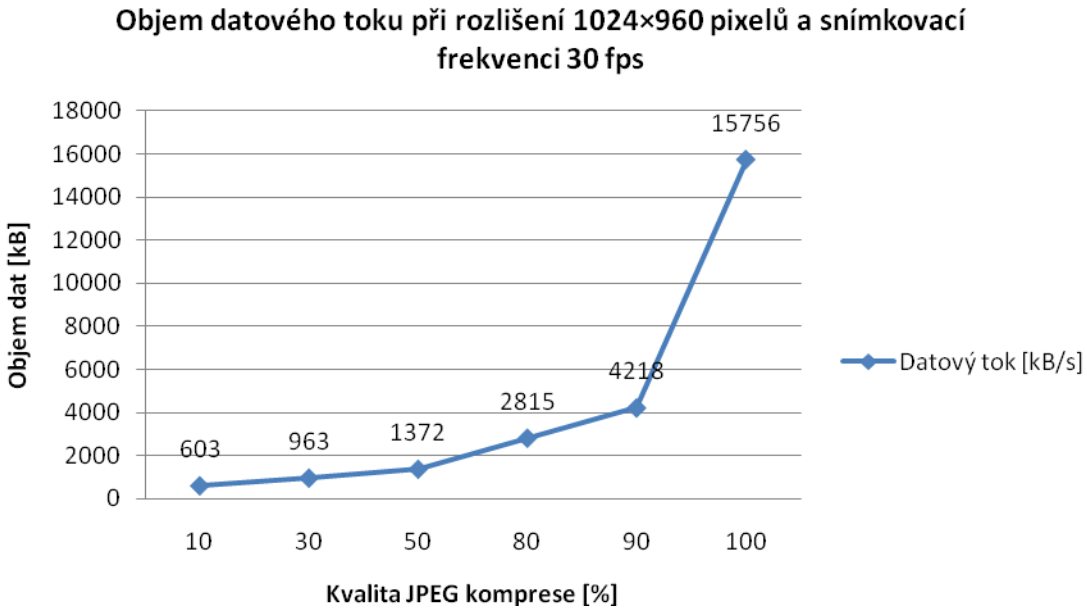
[10] ČERMÁK,J. *C++/CLI a interoperabilita managed a unmanaged kódu: Úvod do jazyka a základni konstrukce* [online], 3. 9. 2009 [cit 2011-04-29]. Dostupné na WWW: http://www.vbnet.cz/clanek--135-c_cli_a_interoperabilita_managed_a_unmanaged_kodu_dil_1_uvod_do_jazyka_a_zakladni_konstrukce.aspx

[11] KÁLDY,R. *Component Object Model: Úvod na začátek* [online], 22. 9. 2003 [cit 2011-04-29]. Dostupné na WWW: http://antonio.cz/com/1.html

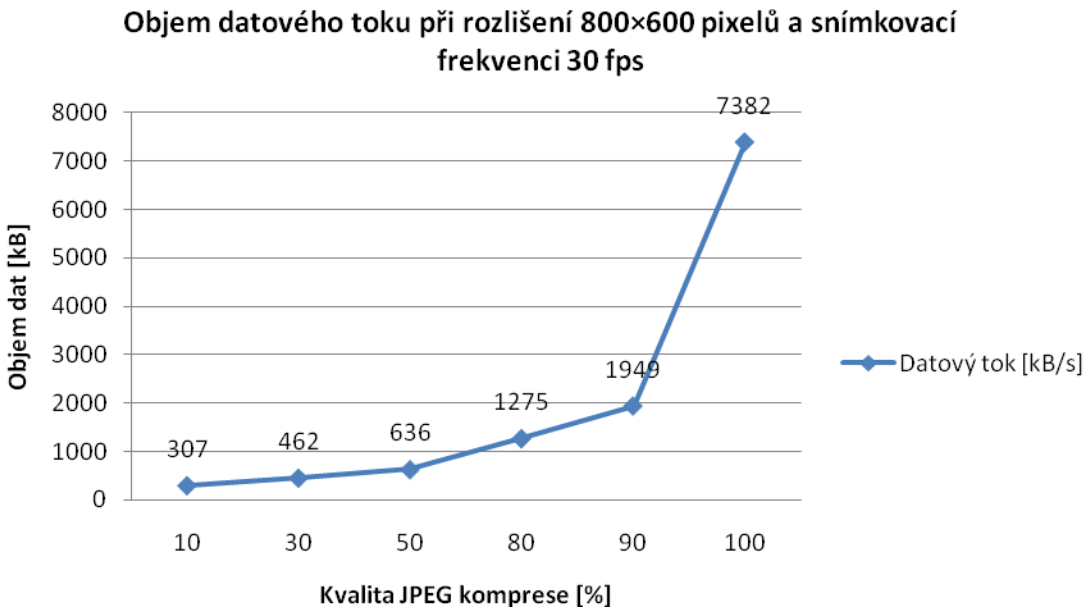
Příloha A – kompletní grafy objemu datového toku



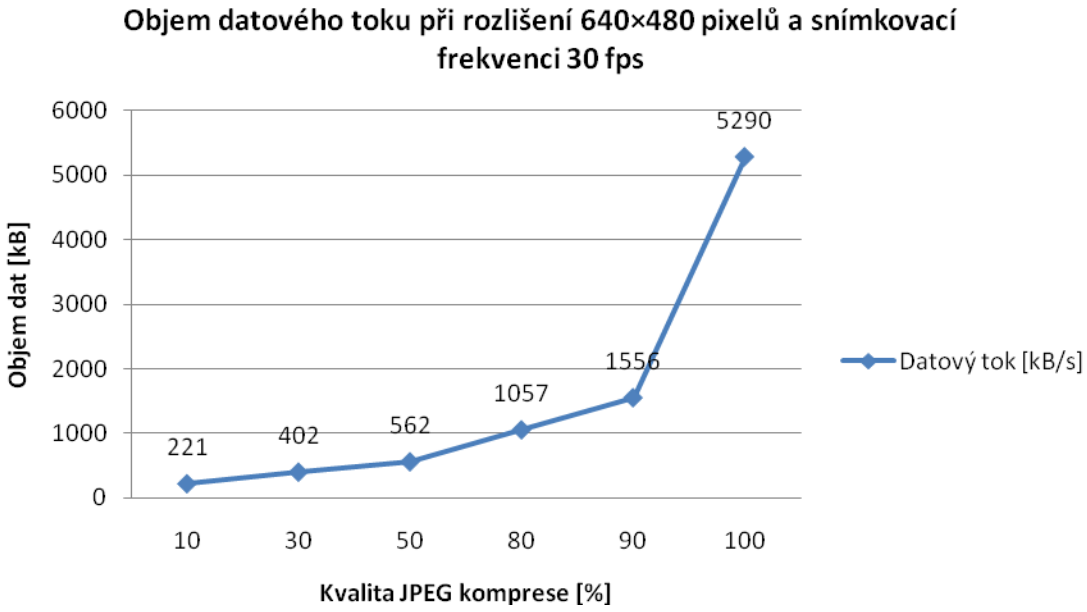
V plné kvalitě bez komprese by byl datový tok 117964 kB/s.



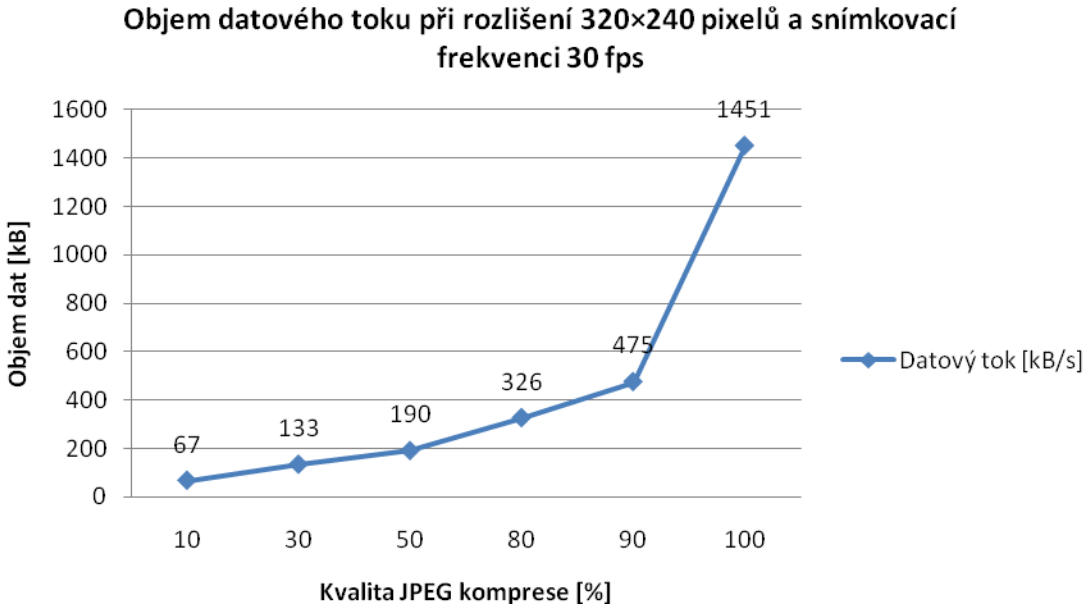
V plné kvalitě bez komprese by byl datový tok 88473 kB/s.



V plné kvalitě bez komprese by byl datový tok 43200 kB/s.



V plné kvalitě bez komprese by byl datový tok 27648 kB/s



V plné kvalitě bez komprese by byl datový tok 6912 kB/s

Příloha B – seznam konstant pro identifikaci příkazů

```
//######## - klientské zprávy bez dat - ########
#define SEND_PIC                    2000
//žádost klienta o jeden snímek
#define SEND_STREAM                 2010
//žádost klienta o posílání celého videa
#define GREYSCALE                   2015
//žádost o posílání černobílého streamu
#define COLOR                       2016
//žádost o posílání barevného streamu
#define JPEG_COMPRESSION            3005
//zapnutí JPEG komprese
#define NO_COMPRESSION              3006
//vypnutí komprese
#define STOP_STREAM                 2020
//žádost o zastavení videa
#define GET_DEVICE_LIST             2030
//žádost o poslání seznamu kamer
#define GET_DEVICE_MODES            2040
//žádost o poslání módů kamery
#define SHUTTING_DOWN               2050
//odeslání zprávy o vypínání klienta

//######## - klientské zprávy obsahující data - ########
#define SET_QUALITY                 3000
//nastavení požadované kvality komprese
#define SET_DEVICE_ON_SERVER        3010
//nastavení kamery na serveru
#define SET_DEVICE_MODE_ON_SERVER   3020
//nastavení módu zařízení
#define CLIENT_INFO                 3030
//informace o klientovi
#define FPS_DIVIDING                3040
//nastavení dělení snímkovací frekvence

//######## - zprávy serveru bez dat - ########
#define DEVICE_LIST_CLEAR           6000
//vymazání obsahu nabídky kamer
#define DEVICE_LIST_MODES_CLEAR     6050
//vymazání obsahu nabídky módů kamery
#define SET_CLIENT_MASTER           7000
//nastavení klienta na MASTER
#define SET_CLIENT_SLAVE            7010
//nastavení klienta na SLAVE

//######## - zprávy serveru obsahující data - ########
#define SERVER_INFO_MESSAGE         3080
//zpráva s informacemi o serveru
```

```
#define VIDEO_FRAME                5000
//jeden snímek videa
#define DEVICE_LIST                6000
//položka nabídky kamer
#define DEVICE_LIST_MODES          6050
//položka nabídky módů kamery
#define SET_DEVICE_ON_CLIENT       3060
//nastavení zařízení u klienta
#define SET_DEVICE_MODE_ON_CLIENT  3070
//nastavení módu kamery u klienta
```