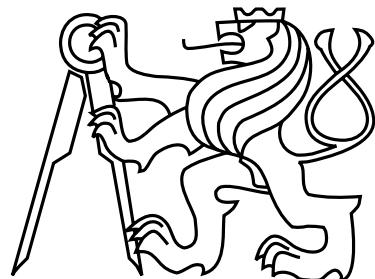


České vysoké učení technické v Praze  
Fakulta elektrotechnická  
Katedra počítačů



Bakalářská práce

**Kalibrační a ovládací software sítě částicových pixelových  
detektorů umístěných uvnitř experimentu ATLAS  
na LHC v CERN**

*Jakub Begera*

Vedoucí práce: Ing. Štěpán Polanský

Studijní program: Otevřená informatika, Bakalářský

Obor: Softwarové systémy

19. května 2016



České vysoké učení technické v Praze  
Fakulta elektrotechnická

katedra počítačů

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Jakub Begera**

Studijní program: Otevřená informatika  
Obor: Softwarové systémy

Název tématu: **Kalibrační a ovládací software sítě částicových pixelových detektorů umístěných uvnitř experimentu ATLAS na LHC v CERN**

### Pokyny pro vypracování:

Cílem této bakalářské práce je návrh a implementace software pro energetickou kalibraci částicových pixelových detektorů pracující Time-Over-Treshold (TOT) režimu a pro řízení sítě těchto detektorů (Atlas TPX), instalované uvnitř experimentu ATLAS na LHC v CERN.

#### Kalibrační část:

Vstupními daty jsou matice hodnot získané měřením dvou až deseti energií rentgenového záření. Pro každý pixel je třeba vytvořit pomocí analytických a statistických metod uvedených v [2] parametry popisující kalibrační funkci, udávající vztah mezi TOT a energií.

#### Řídící část:

Software bude umožňovat ovládat síť hybridních částicových pixelových detektorů a bude umožňovat následující:  
Nastavovat parametry akvizice snímků (akviziční čas, počet snímků, mód)

Nastavovat HW parametry detektoru (bias, TPX clock, DACs)

Výčítání a ukládání dat

### Seznam odborné literatury:

[1] X. Llopart, R. Ballabriga, M. Campbell, L. Tlustos, W. Wong, Erratum to "Timepix, a 65 k programmable pixel readout chip for arrival time, energy and/or photon counting measurements"; [Nucl. Instr. and Meth. A. 581 (2007) 485–494], Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, Volume 585, Issues 1–2, 21 January 2008, Pages 106–108, ISSN 0168-9002, <http://dx.doi.org/10.1016/j.nima.2007.11.003>.

[2] Jan Jakubek, Precise energy calibration of pixel detector working in time-over-threshold mode, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, Volume 633, Supplement 1, May 2011, Pages S262-S266, ISSN 0168-9002, <http://dx.doi.org/10.1016/j.nima.2010.06.183>.

Vedoucí: Ing. Štěpán Polanský

Platnost zadání: do konce letního semestru 2016/2017

prof. Ing. Filip Železný, Ph.D.  
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.  
děkan

V Praze dne 11. 1. 2016



## Poděkování

Zde můžete napsat své poděkování, pokud chcete a máte komu děkovat.



## Prohlášení

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

V Praze dne 20. 5. 2016

.....



# **Abstract**

Translation of Czech abstract into English.

# **Abstrakt**

Abstrakt práce by měl velmi stručně vystihovat její obsah. Tedy čím se práce zabývá a co je jejím výsledkem/přínosem.  
Očekávají se cca 1 – 2 odstavce, maximálně půl stránky.



# Obsah

<b>1</b>	<b>Úvod</b>	<b>1</b>
1.1	Motivace . . . . .	1
<b>2</b>	<b>Detektory ionizujícího záření</b>	<b>3</b>
2.1	Klasifikace detektorů a jejich parametry . . . . .	3
2.2	Geiger-Müllerův počítac . . . . .	3
2.3	Sintilační detektory . . . . .	3
2.4	Polovodičové detektory . . . . .	3
2.4.1	Princip . . . . .	3
2.4.2	Módy . . . . .	3
2.4.3	Medipix . . . . .	3
2.4.4	Timepix . . . . .	3
2.4.5	Cluster analýza . . . . .	3
2.4.6	FitPix . . . . .	3
2.4.7	Pixelman . . . . .	3
2.4.8	Aplikace . . . . .	3
<b>3</b>	<b>Energetická kalibrace</b>	<b>5</b>
<b>4</b>	<b>ATLAS TPX</b>	<b>7</b>
4.1	ATLAS MPX . . . . .	8
4.1.1	Hardwarová a softwarová architektura sítě ATLAS MPX . . . . .	9
4.2	Hardwarová architektura sítě ATLAS TPX . . . . .	10
4.3	Softwarová architektura sítě ATLAS TPX . . . . .	12
4.4	Řídící software a jeho implementace . . . . .	14
4.4.1	Řízení detektorů . . . . .	14
4.4.1.1	Komunikační protokol . . . . .	14
4.4.1.2	Popis příkazů komunikačního protokolu . . . . .	15
4.4.1.3	Synchronní a asynchronní příkazy komunikačního protokolu .	19
4.4.1.4	Implementace modulu pro řízení detektorů na straně serveru	21
4.4.1.5	Emulátor detektoru . . . . .	23
4.4.2	REST API server . . . . .	25
4.4.2.1	Konfigurace a spuštění serveru . . . . .	25
4.4.2.2	Metody poskytované serverem . . . . .	26
4.4.2.3	Implementace serveru . . . . .	27

4.4.3	Zpracování a ukládání dat . . . . .	28
4.4.3.1	Datový server . . . . .	30
<b>5</b>	<b>Závěr</b>	<b>31</b>
<b>A</b>	<b>Konfigurační soubor ATLAS TPX serveru</b>	<b>35</b>
<b>B</b>	<b>Obsah přiloženého CD</b>	<b>39</b>

# Seznam obrázků

4.1	ATLAS TPX detektor - vrstvy a rozmístění konvertorů . . . . .	7
4.2	ATLAS TPX - přehledem rozmístění detektorů . . . . .	8
4.3	Snímek z ATLAS MPX detektoru s výřezem zachycených částic (převzato z [1])	9
4.4	Fotografie znázorňující Medipix2 detektor s neutronovými konvertory (převzato z [8]) . . . . .	9
4.5	ATLAS MPX - řídící aplikace (převzato z [6]) . . . . .	10
4.6	ATLAS TPX - diagram hw komponent . . . . .	11
4.7	ATLAS TPX - fotografie hw komponent . . . . .	11
4.8	ATLAS TPX - diagram softwarových komponent . . . . .	12
4.9	Cluster analýza - 6 základních typů clusterů (převzato z [6]) . . . . .	14
4.10	Příklad použití komunikačního protokolu . . . . .	20
4.11	Diagram tříd modulu pro ovládání detektorů . . . . .	22
4.12	Emulátor detektora - sekvenční diagram připojení klienta a pořízení snímku .	24
4.13	BPMN inicializace REST API serveru . . . . .	27
4.14	Diagram tříd komponenty pro ukládání dat ATLAS TPX serveru . . . . .	29
B.1	Seznam přiloženého CD — příklad . . . . .	39



# Seznam tabulek

4.1 Komunikační protokol - struktura paketů z pohledu serveru . . . . .	15
4.2 Komunikační protokol - přehled příkazů . . . . .	15



# Seznam zdrojových kódů

A.1 Konfigurační soubor ATLAS TPX serveru . . . . .	37
---	----



# Kapitola 1

## Úvod

Tato bakalářská práce se zabývá návrhem a implementací software pro ovládání a kalibraci sítě hybridních částicových pixelových detektorů umístěných uvnitř experimentu ATLAS na LHC v CERN - projekt AtlasTPX. Jelikož proces kalibrace je zcela nezávislý na následném řízení činnosti těchto detektorů, je tento software členěn na dvě nezávislé části, a to na energetickou kalibraci částicových pixelových detektorů (viz kapitola 3) a řízení sítě těchto detektorů - ATLAS TPX (viz kapitola 4).

**TODO**

### 1.1 Motivace

Ionizující záření je spjato s naším světem už od počátku jeho existence. Jeho studium začalo koncem 19. století a pomáhá nám pochopit podstatu hmoty, její interakce s prostředím a další vlastnosti. Tyto poznatky našli své uplatnění v mnoha oborech, jako například v defektoskopii, zdravotnictví, energetice a v mnoha dalších. Spolu s rostoucími poznatkami o ionizujících záření a s technických pokrokem se rozvíjela i detekční technika, která za poslední století prodělala veliký posun. Od prvních bublinových komor, až po nejmodernější polovodičové pixelové detektory, kterými se tato práce zabývá.



## Kapitola 2

# Detektory ionizujícího záření

2.1 Klasifikace detektorů a jejich parametry

2.2 Geiger-Müllerův počítac

2.3 Sintilační detektory

2.4 Polovodičové detektory

2.4.1 Princip

2.4.2 Módy

2.4.3 Medipix

2.4.4 Timepix

2.4.5 Cluster analýza

2.4.6 FitPix

2.4.7 Pixelman

2.4.8 Aplikace



## Kapitola 3

# Energetická kalibrace

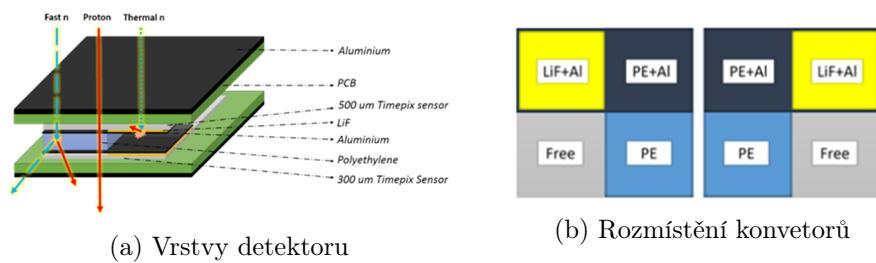


## Kapitola 4

# ATLAS TPX

ATLAS TPX, síť 16<sup>1</sup> hybridních částicových pixelových detektorů typu Timepix [2.4.4](#), instalovaných na různé pozice experimentu ATLAS na LHC<sup>2</sup> v CERN během LS2<sup>3</sup> (leden 2013 až březen 2015) je následníkem svého předchůdce - síť ATLAS MPX (viz [4.1](#)). Hlavní motivací výměny této sítě bylo využití nových technologií, především pak nového detekčního čipu Timepix. Ten na rozdíl od svého předchůdce Medipix2 [2.4.3](#) umožňuje rozšíření naměřené informace i o časovou oblast (viz [2.4.4](#)). To nově umožňuje provozovat detektory v módech TOA<sup>4</sup> a TOT<sup>5</sup>.

Další změnou nové detektorové sítě ATLAS TPX oproti svému předchůdci je, že každý detektor obsahuje dva detekční čipy s tloušťkami  $300\ \mu m$  a  $500\ \mu m$ , umístěné předními stranami k sobě - viz [4.1a](#). To přináší možnost měřit koincidence - když částice projde oběma vrstvami detektoru a zároveň v každé nechá jisté měřitelné množství své energie, je detekována oběma vrstvami a je možné zpětně zrekonstruovat její trajektorii. Tyto koincidence se nejsnáze detekují, pokud oba Timepix čipy pracují v módu TOA - jelikož rychlosť častic se blíží rychlosti světla, je vysoce pravděpodobné, že zasažené pixely budou mít stejnou hodnotu.



Obrázek 4.1: ATLAS TPX detektor - vrstvy a rozmístění konvertorů

<sup>1</sup>V průběhu LS3<sup>3</sup> (plánováno 2017 - 2018) je plánováno rozšíření této sítě o nové detektory

<sup>2</sup>z anglicky Large Hadron Collider

<sup>3</sup>z anglicky long shutdown - dlouhodobá technologická přestávka LHC

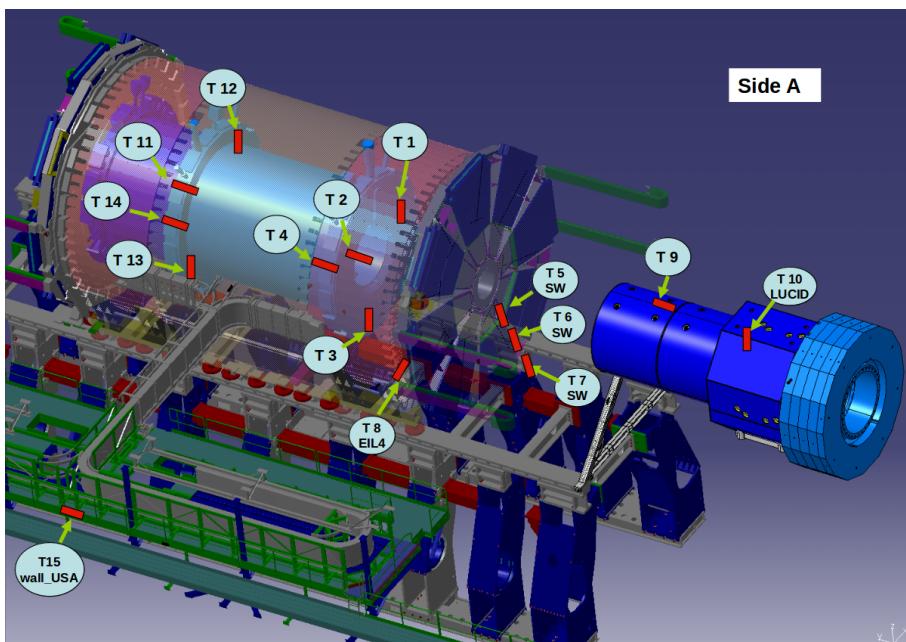
<sup>4</sup>z anglicky Time of Arrival - čas příletu částice v hodinových cyklech detektoru od začátku akvizice

<sup>5</sup>z anglicky Time Over Threshold - počet hodinových cyklů, kdy komparační napětí je větší, než referenční (ekvivalent energie deponované částice, viz kapitola [3](#))

Mezi vrstvami detektoru je umístěn konvertující materiál pro detekci termálních a rychlých neutronů. Rozmístění těchto konvertorů je na obrázku 4.1b.

Hlavním úkolem ATLAS TPX experimentu je online monitorování spektrálních charakteristik velice různorodého radiačního prostředí ATLAS experimentu, založené na prostorovém uspořádání sítě a (vzhledem k aktuálním módům detektoru) i na informaci o deponované energii interagujících částic, či na času jejich interakce.

Detektory, instalované blízko interakčnímu bodu, jsou rovněž použity jako monitory integrované luminozity, což je veličina, která udává počet realizovaných srážek, resp. s intenzitou svazku urychlovače. Podle [7] je to veličina, která v případě srážení dvou proti sobě letících svazků ukazuje, jaký je součin počtu částic v jednotlivých svazcích prolétajících jednotkovou plochou v srážkové oblasti, vynásobený počtem obletů svazků za jednotku času (nejčastěji se vyjadřuje v jednotkách na centimetr čtvereční a sekundu).



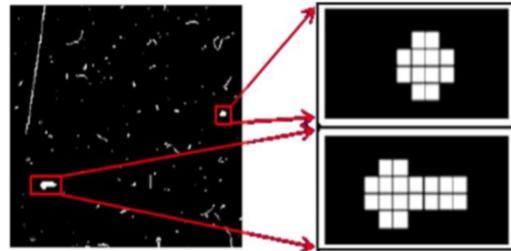
Obrázek 4.2: ATLAS TPX - přehledem rozmístění detektorů

## 4.1 ATLAS MPX

ATLAS MPX[8][1] je předchůdcem detektorové sítě ATLAS TPX, který je v současné době plně nahrazen. Detektorová síť ATLAS MPX se skládala z 16 Medipix2 detektorů, které byly instalovány na různé pozice ATLAS detektoru. Hlavním cílem této sítě bylo měření vlastností radiačního pole uvnitř experimentu Atlas, jeho složení, spektroskopických charakteristik a částečně také přispěla k měření neutronů.

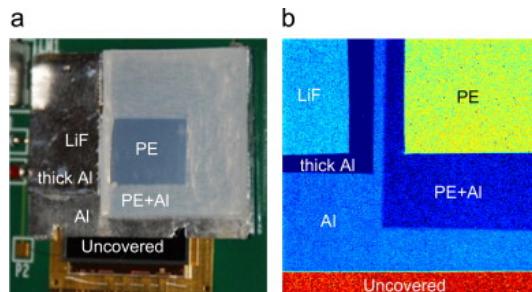
Všechny detektory operovaly v tzv. Medipix módu, který se vyznačuje tím, že v rámci jedné akvizice počítá počet částic, které interagovaly pixelovou maticí detektoru a jejichž deponovaná energie byla vyšší, než prahová. Na obrázku 4.3 je znázorněn snímek z jednoho

detektoru s detailem zachycených částic. Na obrázku vpravo nahoře je částice typu **heavy blob** (těžce nabité částice, jejíž trajektorie byla kolmá s povrchem detektoru), vpravo dole je pak zachycena částice typu **heavy track** (také těžce nabité částice, která ale přiletěla pod větším úhlem a proto zanechala větší stopu) - více klasifikaci částic se dočtete v podkapitole 4.3.



Obrázek 4.3: Snímek z ATLAS MPX detektoru s výřezem zachycených částic (převzato z [1])

Každý z těchto detektorů byl osazen  $300 \mu\text{m}$  tlustým křemíkovým senzorem, který byl pokryt konvertory pro lepší detekční účinnost neutronů (obr. 4.4).



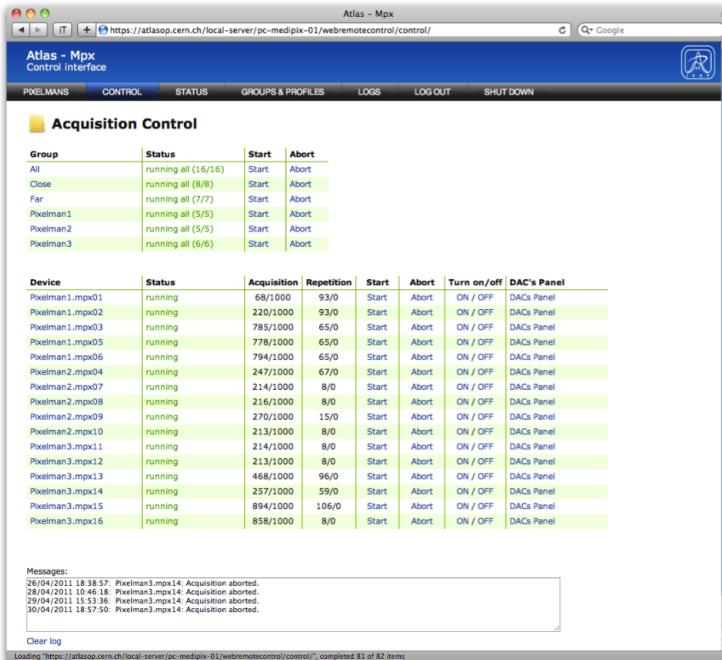
Obrázek 4.4: Fotografie znázorňující Medipix2 detektor s neutronovými konvertory (převzato z [8])

#### 4.1.1 Hardwarová a softwarová architektura sítě ATLAS MPX

Tato síť se skládala z 16 Medipix2 2.4.3 detektorů, které byly pomocí USB vyčítacího rozhraní FITPiX 2.4.6 připojeny ke třem počítačům (z důvodu distribuce toku dat a výkonu). Na každém počítači se o komunikaci s detektory staral software Pixelman 2.4.7, který řídil akvizici dat, nastavování parametrů detektorek apod.

Pro vzdálené obládání byl vyvinut plugin pro Pixelman, který umožňoval jeho rozšíření o TCP/IP ovládací vrstvu. Pomocí jednoduchého textového protokolu bylo tedy možné řídit každý ze třech uzlů. Pro tyto účely byla vyvinuta centrální řídící aplikace [5], pomocí které bylo možné řídit řídit akvizici všech detektorek a nastavovat jejich parametry. Tato aplikace poskytovala webové rozhraní (obr. 4.5), které díky tomu dobu méně striktním nárokům ze

strany CERNu na síťovou bezpečnost bylo možné tento experiment ovládat odkudkoliv z internetu.



Obrázek 4.5: ATLAS MPX - řídící aplikace (převzato z [6])

## 4.2 Hardwarová architektura sítě ATLAS TPX

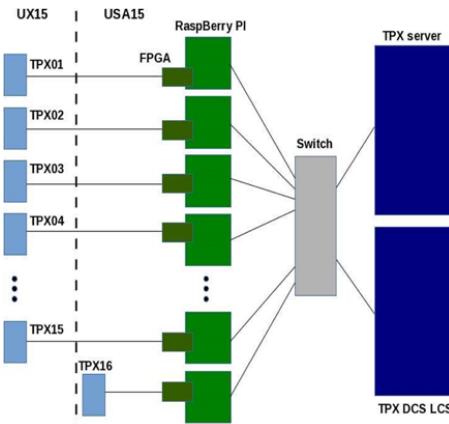
Při návrhu hardwarové architekty sítě ATLAS TPX musela být zohledněna zvýšená intenzita radiačního a elektromagnetického pole v prostorách ATLAS detektoru. Snahou proto bylo, co nejvíce hardwarových komponent umístit z dosahu tohoto pole. Z pohledu hardwarové instalace této detektorové sítě se prostory ATLAS experimentu dělí na dvě části - UX15 a USA15 (viz obr. 4.6). V UX15 se nachází vlastní experiment. V tomto prostoru byly umístěny jen detektory (na obr. 4.6 TPX01 až TPX15) a zbytek sítě byl instalován v USA15, kterou od zbytku experimentu dělí cca 60 m tlustá železobetonová stěna. Tady se nachází vyčítací elektronika a další nezbytný hardware.

Na obrázku 4.7 je fotografie těchto komponent. Jak již bylo zmíněno výše, detektor (z obr. 4.7, na obr. 4.6 jako TPX01 až TPX15) se skládá z dvojice detekčních čipů Timepix, které jsou pomocí LVDS zesilovačů a cca 100 m dlouhých ethernetových kabelů propojeny se zařízením AtlasPix (obr. 4.7 dole), které vzniklo modifikací vyčítacího rozhraní FITPix 2.4.6. Toto zařízení obsahuje FPGA<sup>6</sup>, minipočítač Raspberry Pi a další podpůrnou elektroniku.

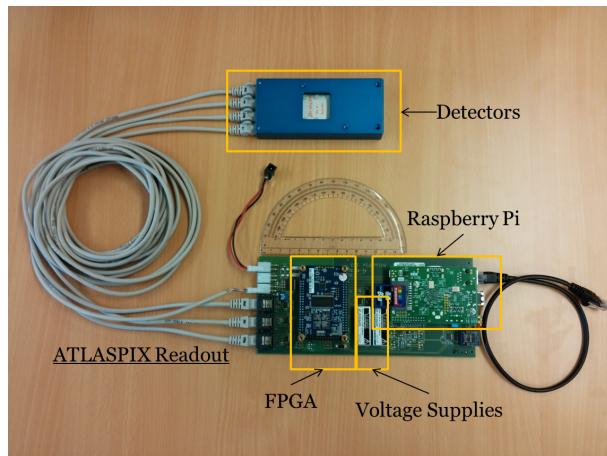
FPGA se stará o komunikaci s Timepix detektory, v rámci které dochází k nastavování řídících registrů Timepix čipů, ovládání akvizice, vyčítání dat, řízení triggeru<sup>7</sup> apod.

<sup>6</sup>z angl. Field Programmable Gate Array (programovatelné hradlové pole)

<sup>7</sup>řídící signál, který spouští resp. zastavuje (dle konfigurace) akvizici detektoru



Obrázek 4.6: ATLAS TPX - diagram hw komponent



Obrázek 4.7: ATLAS TPX - fotografie hw komponent

Dalším článkem tohoto řetězce je minipočítač **Raspberry Pi**, který plní dvě úlohy. Tou první je komunikace s **FPGA** pomocí SPI<sup>8</sup> rozhraní a deserializace (získání dat ze struktury komunikačního protokolu) a derandomizace (není zaručena časová posloupnost) surových dat z **FPGA**. Druhou úlohou tohoto zařízení je poskytování API<sup>9</sup> vyšším řídícím vrstvám této sítě pomocí specifikovaného komunikačního protokolu a klasického ethernetového rozhraní.

Všechny tyto zařízení jsou pomocí ethernetového switche propojeny s **TPX serverem**, centrálním bodem této sítě, který jí pomocí řídícího softwaru [4.4](#) a komunikačního protokolu [4.4.1](#) ovládá. Zároveň je k síti připojen i **TPX DCS**<sup>10</sup> server, pomocí kterého jsou různé stavové informace ATLAS TPX sítě předávány CERNu, resp. ATLAS experimentu. Tyto stavové informace jsou převážně hardwarového charakteru (na př. napětí, časování apod.), ale také jsou předávána data o počtu pořízených snímků, jejich okupanci apod.

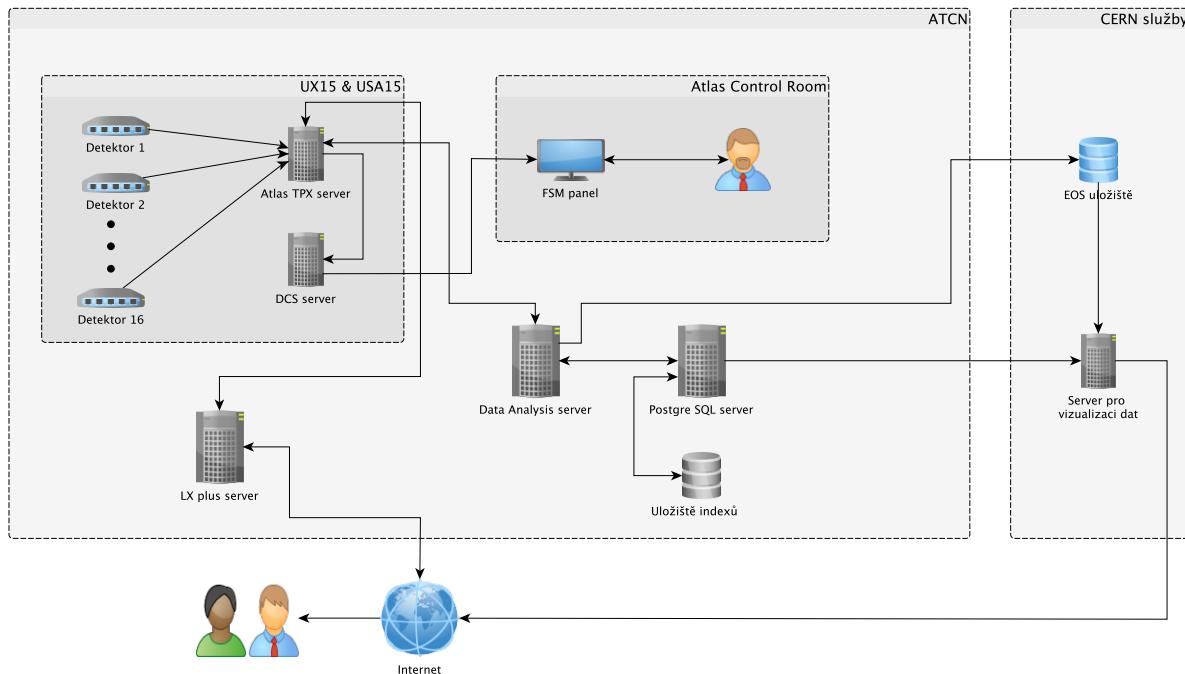
<sup>8</sup>z anglicky Serial Peripheral Interface (sériové periferní rozhraní)

<sup>9</sup>z anglicky Application Programming Interface (aplikaci programovací rozhraní)

<sup>10</sup>z anglicky Data Control System

### 4.3 Softwarová architektura sítě ATLAS TPX

Na obrázku 4.8 je znázorněn diagram návrhu softwarové architektury sítě ATLAS TPX z pohledu jejího řízení, vizualizace dat a předávání stavových informací CERNu. Diagram je členěn do dvou základních částí - ATCN<sup>11</sup> (technická síť ATLAS experimentu, která je oddělena od zbytku ATLAS sítě a obsahuje systémy pro vyčítání dat a pro řízení, včetně TDAQ<sup>12</sup> a DSC [4]) a CERN služby, které poskytují perzistentní úložiště dat a web server pro jejich vizualizaci.



Obrázek 4.8: ATLAS TPX - diagram softwarových komponent

**Popis architektury z pohledu řízení:** Na obrázku 4.8 se nachází ATLAS TPX server, který je umístěn v serverové místnosti (USA15) ATLAS experimentu, umístěně cca 100 m pod zemským povrchem. Tento server pomocí komunikačního protokolu (specifikovaném v 4.4.1) řídí činnost detektorů (nastavování parametrů, ovládání akvizice apod.). Zároveň pomocí JSON REST API poskytuje rozhraní pro své řízení a předávání stavových informací (více v 4.4.2). Díky tomuto rozhraní je možné činnost serveru řídit z ATCN sítě. Pro potřeby vzdáleného ovládání mimo síť ATCN slouží LX plus server, který zajistí spojení vytvořením SSH tunelu.

Předávání stavových informací zajišťuje DCS server, který je od ATLAS TPX serveru získává pomocí jeho API. Hlavním úkolem DCS je zajištění získávání stavových informací ze všech experimentů a detektorů homogenním způsobem a interakce s LHC

<sup>11</sup>z anglicky ATLAS Technical Control Network

<sup>12</sup>z anglicky Trigger and Data Aquisition (trigger a akvizice dat)

(předávání dat luminozitě, stavu svazku urychlovače, radiační pozadí apod.). Tato data jsou dále předávána do ATLAS Control Room, která se nachází na povrchu. Tam jsou tato data operátorů prezentována pomocí FSM panelu, což je aplikace, která vizualizuje stromovou strukturu všech systému a detektorů ATLAS experimentu. Každý list této stromové struktury (detektor, senzor atd.) má několik proměnných, z nichž každá má předem definované intervaly s příslušnými stavami (OK, WARNING, ERROR, FATAL atd.). Výhodou této struktury je, že pokud kterýkoliv list změní svůj stav, tak se tato informace propaguje přes všechny nadřazené uzly, tudíž odhalení případné chyby je pro operátory mnohem snazší.

**Popis architektury z pohledu analýzy a vizualizace dat:** Když kterýkoliv detektor dokončí akvizici snímku, tak vygeneruje a pošle asynchronní událost ATLAS TPX serveru s informací, že data jsou připravena k vyčtení. Následně server vyčte snímek z detektoru (i s jeho metadaty), zpracuje a připojí k němu informace o nastavení detektoru. Poté je třeba data přenést do Data analysis serveru, což v principu je možné dvěma<sup>13</sup> způsoby:

1. ATLAS TPX server uloží získaná data v textové podobě do lokálního (či síťového) datového úložiště, odkud jsou přenesena do Data analysis serveru pomocí automatického kopírovacího skriptu.
2. Druhou možností je přenesení dat pomocí JSON REST API protokolu, který je Data analysis serverem implementován. Tento druhý způsob je výhodnější, neboť minimalizuje prodlevu mezi dobou pořízení snímku a následném zpracováním Data analysis serverem a dostupnosti jeho vizualizace pomocí web serveru a zároveň přináší úsporu objemu přenesených dat.

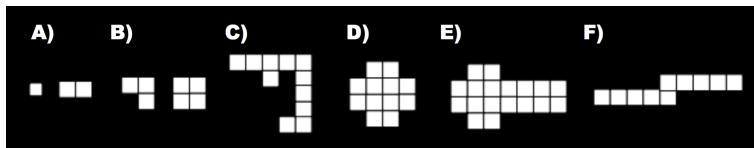
Hlavní úlohou Data analysis serveru je provedení tzv. Cluter analýzy. Jde o proces, při kterém jsou z každého snímku získány shluky sousedních pixelů (tzv. clusterů), které mají nenulovou hodnotu. Z těchto clusterů, resp. z jejich tvaru a celkové deponované energie částice (pokud zasažení pixely operovaly v TOT módu) je možné zjistit typ částice, která danou událost způsobila. Na obrázku 4.9 můžete vidět 6 základních dělení clusterů, kde

- (a) je tzv. DOT, způsobený fotony, či elektrony o energii do 10 keV
- (b) je tzv. SMALL BLOB, způsobený fotony, či elektrony s energií většinou nad 10 keV
- (c) je tzv. CURLY TRACK, způsobený elektrony do 10 MeV
- (d) je tzv. HEAVY BLOBS, způsobený těžce nabitémi částicemi (na př. alfa)
- (e) je tzv. HEAVY TRACK, způsobený těžce nabitémi částicemi (na př. protony)
- (f) je tzv. STRAIGHT TRACK, způsobený těžce nabitémi částicemi (muony apod.)

Po dokončení analýzy dat, jsou data uložena do ROOT<sup>14</sup> souborů (do souborové struktury po detektorech a hodinách). ROOT je framework, který je vyvíjen v CERN a je určený pro

<sup>13</sup>V současné době je možný pouze první způsob, protože díky politice CERNské síťové bezpečnosti a dlouhými schvalovacími termíny je Data analysis server a všechny s ním související systémy (web server, databáze s indexem a vlastní úložiště dat) prozatím umístěn v ÚTEF ČVUT v Praze.

<sup>14</sup><<https://root.cern.ch/>>



Obrázek 4.9: Cluster analýza - 6 základních typů clusterů (převzato z [6])

ukládání dat a jejich analýzu. Vygenerované soubory jsou ukládány do EOS úložiště, což je služba pro perzistentní ukládání ROOT souborů provozovaná CERN.

Jelikož každý ROOT soubor má velikost řádově v jednotkách *GB*, jakékoli operace nad nimi (na příklad vyhledávání) jsou velice časově náročné. Z tohoto důvodu vznikla PostgreSQL databáze s indexem na jednotlivé clustery, obsažené v ROOT souborech. Pro vizualizaci dat slouží web server, který pomocí databáze s indexem a ROOT souborů poskytuje online výsledky cluster analýzy a další informace.

## 4.4 Řídící software a jeho implementace

Tato kapitola je věnována návrhu a implementaci řídícího software sítě ATLAS TPX, který je nasazen na ATLAS TPX serveru (viz obr. 4.8). Úkolem tohoto software se zajištění komunikace s detektory, zejména pak řízení akvizice, nastavování parametrů a vyčítání dat - tato komunikace bude popsána v kapitole 4.4.1. Další úlohou tohoto software je poskytování rozhraní pro řízení své činnosti a pro předávání stavových informací o ATLAS TPX síti. Toto rozhraní je implementováno pomocí JSON REST API serveru a detailně bude popsáno v kapitole 4.4.2.

### 4.4.1 Řízení detektorů

Jak již bylo zmíněno výše, detektor se skládá ze dvou **Timelpix** detekčních čipů, **FPGA** a minipočítače **Raspberry Pi**, který implementuje komunikační protokol, díky kterému umožňuje řídícímu software ovládání činnosti detektoru. V podkapitole 4.4.1.1 naleznete popis tohoto komunikačního protokolu.

#### 4.4.1.1 Komunikační protokol

Tabulka 4.1 znázorňuje strukturu komunikačního rámce (tzv. paketu) tohoto protokolu pomocí posloupnosti bytů z pohledu ATLAS TPX serveru, resp. z pohledu řídícího software. V horní části se nachází struktura odchozího paketu, kde:

**0x55**<sup>15</sup> je tzv. START BYTE, který značí začátek paketu,

**CMD** je typ příkazu (tzv. COMMAND TYPE) - viz tabulka přehledu příkazů 4.2,

<sup>15</sup>značení v hexadecimální soustavě

**SIZE 1..4** je pole vždy o velikosti čtyřech bytů značících velikost (resp. počet bytů) položky **DATA**, zakódovaných v **BIG ENDIAN**,

**DATA 1 .. DATA n** je pole vlastních přenesených dat o velikosti  $n$ ,

**0xAA STOP BYTE**, který značí konec paketu.

Struktura odchozího paketu je velice podobná, až na byte **ERR**, který oproti příchozího paketu obsahuje. Tento byte může nabývat hodnoty 0x00 (když zpracování požadavku serveru detektorem proběhlo bez chyby), nebo 0x01 (jinak).

0x55	CMD	SIZE 1	SIZE 2	SIZE 3	SIZE 4	DATA 1 .. DATA $n$	0xAA	} odchozí } paket
0x55	CMD	ERR	SIZE 1	SIZE 2	SIZE 3	SIZE 4	DATA 1 .. DATA $n$	0xAA } příchozí } paket

Tabulka 4.1: Komunikační protokol - struktura paketů z pohledu serveru

Hodnota příkazu	Název příkazu
0x01	Ping
0x02	Get status of the detector
0x03	Reset of the device
0x04	Set Bias and Timepix Clock
0x05	Get Bias and Timepix Clock
0x06	Set Pixel Configuration
0x07	Get Pixel Configuration
0x08	Set DAC
0x09	Get DAC
0x0A	Perform Digital Test
0x0B	Perform Acquisition
0x0C	Readout Measured Data
0x0D	Direct FITPix Command
0x0E	Stop acquisition
0xFD	Asynchronous Event from device
0xFE	Reboot device
0xFF	Shut down

Tabulka 4.2: Komunikační protokol - přehled příkazů

#### 4.4.1.2 Popis příkazů komunikačního protokolu

Následuje stručný popis příkazů komunikačního protokolu z pohledu obsahu vlastních přenesených dat odchozího a příchozího paketu (viz tabulka 4.1)

**0x01 - Ping** : Příkaz pro ověření spojení s detekorem. Na základě rozdílu času odeslání a přijetí paketu je vypočtena prodleva spojení v *ns* (tzv. ping)

**Odchozí data:** nic

**Příchozí data:** nic

**0x02 - Get status of the detector** : Příkaz pro zjištění stavu detektoru.

**Odchozí data:** nic

**Příchozí data (2 B):** GST MST

- GST - General Status (obecný status)
  - 0x00 - Ok
  - 0x01 - Chyba detekčního čipu
  - 0x02 - Obecná chyba
- MST - Measurement Status (měřící status)
  - 0x00 - Nečinný
  - 0x01 - Probíhá akvizice
  - 0x02 - Čekání na trigger
  - 0x03 - Data připravena k vyčtení
  - 0x04 - Chyba akvizice

**0x03 - Reset of the device** : Tento příkaz slouží pro vyresetování FPGA a dalších řídících struktur detektoru.

**Odchozí data:** nic

**Příchozí data:** nic

**0x04 - Set Bias and Timepix clock** : Příkaz nastavující napětí (Bias) na obou Timepix čipech a jejich měřící frekvenci (Timepix clock)

**Odchozí data (24 B):** BIAS1(8 B) BIAS2(8 B) CLK(8 B)

- BIAS1, BIAS2 - hodnota napětí pro Timepix čipy (zakódovaná jako 64-bitové double-precision číslo dle standartu IEEE 754)
- CLK - Měřící frekvence obou Timepix čipů (zakódovaná jako 64-bitové double-precision číslo dle standartu IEEE 754)

**Příchozí data:** nic

**0x05 - Get Bias and Timepix clock** : Příkaz pro vyčtení úrovně napětí z obou Timepix čipů a jejich měřící frekvencí

**Odchozí data:** nic

**Příchozí data (24 B):** BIAS1(8 B) BIAS2(8 B) CLK(8 B)

- BIAS1, BIAS2 - hodnota napětí pro Timepix čipy (zakódovaná jako 64-bitové double-precision číslo dle standartu IEEE 754)
- CLK - Měřící frekvence obou Timepix čipů (zakódovaná jako 64-bitové double-precision číslo dle standartu IEEE 754)

**0x06 - Set Pixel Configuration** : Příkaz pro nastavení konfigurace pro každý pixel detektoru

**Odchozí data ((1 + (65536 nebo 2 \* 65536)) B):** TYPE(1 B) PIXCFG((65536 nebo 2 \* 65536) B)

- TYPE - Výběr čipu, pro který je konfigurace určena
  - 0x00 - Konfigurace určena oboum čipům (délka PIXCFG je 2 \* 65536 B)
  - 0x01 - Konfigurace určena prvnímu čipu (délka PIXCFG je 65536 B)
  - 0x02 - Konfigurace určena druhému čipu (délka PIXCFG je 65536 B)
- PIXCFG - Pole konfiguračních bytů (každý byte je pro jeden pixel) pro jednotlivé pixely. Délka tohoto pole je závislá na parametru TYPE. Struktura bytu pro konfiguraci pixelu je následovná:  
MASK\_BITE(1 b) TEST\_BITE(1 b) THL(4 b) MODE(2 b)
  - MASK\_BITE - Pixel je aktivní při hodnotě 1, zamaskovaný při 0.
  - TEXT\_BITE - Slouží pro zapnutí testovacího módu pixelu, pomocí externího generátoru pulzů.
  - THL - Tyto čtyři byty udávají číslo od 0 do 15, které je použito pro posun globální hodnoty THL
  - MODE - Mód pixelu (0 - Medipix, 1 - Time-Over-Treshold, 2 One-Hit, 3 - Time-of-Arrival)

**Příchozí data:** nic

**0x07 - Get Pixel Configuration** : Příkaz pro vyčtení konfigurace pixelů detektoru.

**Odchozí data (1 B):** TYPE - Výběr čipu, pro který je konfigurace určena (viz předchozí příkaz)

**Příchozí data ((65536 nebo 2 \* 65536) B):** PIXCFG (viz předchozí příkaz)

**0x08 - Set DAC** : Příkaz pro nastavení hodnot DAC převodníků detektoru.

**Odchozí data (3 B):** DAC\_IDX DAC\_VAL DAC\_VAL

- DAC\_IDX - DAC index
- DAC\_VAL - DAC hodnota

**Příchozí data:** nic

**0x09 - Get DAC** : Příkaz pro vyčtení hodnoty DAC převodníků detektoru.

**Odchozí data (1 B):** DAC\_IDX

- DAC\_IDX - DAC index

**Příchozí data (2 B):** DAC\_VAL DAC\_VAL

- DAC\_VAL - DAC hodnota

**0x0A - Perform Digital Test** : Příkaz pro vykonání testu digitálních částí detektoru.

**Odchozí data:** nic

**Příchozí data (3 B):** BPC BPC BPC

- BPC - Počet chybných pixelů detektoru

**0x0B - Perform Acquisition :** Příkaz pro zahájení akvizice snímku/snímků.

**Odchozí data (13 B):** ACQTM(8 B) ACQCNT(4 B) TGRMOD

- ACQTM - Akviziční čas v sekundách (zakódovaný jako 64-bitové double-precision číslo dle standartu IEEE 754).
- ACQCNT - Počet snímku (pokud je 0, detektor bude opakovat akvizici s těmito parametry až do její zastavení)
- TRIGMOD - Mód triggeru
  - 0x00 - bez triggeru
  - 0x01 - akvizice zahájena na signál triggeru
  - 0x02 - akvizice ukončena na signál triggeru

**Příchozí data:** nic

**0x0C - Read Measured Data :** Tento příkaz slouží pro vyčtení naměřených dat z detektoru.

**Odchozí data (6 B):** DET TYPE FRAME\_ID FRAME\_ID FRAME\_ID FRAME\_ID

- DET - selektor Timepix čipu
  - 0x00 - oba čipy
  - 0x01 - jen první čip
  - 0x02 - jen druhý čip
- TYPE - typ vyčítaných dat
  - 0x00 - deserializovaný a derandomizovaný snímek
  - 0x01 - surová data z FPGA
  - 0x02\_ID - metadata k danému snímku (akviziční čas, napětí apod.)
- FRAME\_ID - ID naměřeného snímku

**Příchozí data:** Velikost a typ příchozích dat se liší dle použitých parametrů DET a TYPE

- Pro TYPE 0x00 přijde pole bytů hodnot čítačů jednotlivých pixelů (hodnota každého pixelu je reprezentována pomocí dvou bytů)
- Pro TYPE 0x01 přijdou blíže nespecifikovaná surová data z FPGA
- Pro TYPE 0x02 přijdou metadata, vázající se k danému snímku s následující strukturou:
  - UNIX časové razítka začátku akvizice [ms] (5 B)
  - Bias1 - měřící napětí prvního čipu [V] (8 B, double-precision dle IEEE 757)
  - Bias2 - měřící napětí druhého čipu [V] (8 B, double-precision dle IEEE 757)
  - Měřící frekvence [Hz] (8 B, double-precision dle IEEE 757)
  - Doba akvizice [s] (8 B, double-precision dle IEEE 757)

**0x0D - Direct FPGA Command** : Příkaz pro přímé poslání dat do FPGA a získání odpovědi.

**Odchozí data:** Dle použitého příkazu

**Příchozí data:** Dle použitého příkazu

**0x0E - Stop acquisition** : Příkaz pro zastavení aktuálně probíhající akvizice.

**Odchozí data (1 B):** TYPE

- TYPE: Typ zastavení
  - 0x00 - Zastavení akvizice po dokončení aktuálně pořizovaného snímku
  - 0x01 - Bezprostřední zastavení akvizice

**Příchozí data:** nic

**0xFD - Asynchronous Event From Device** : Tento typ příkazu je asynchronní a detektor ho posílá samovolně dle nastalé události - na příklad dokončení akvizice.

**Příchozí data (5 B):** EVID VAL VAL VAL VAL

- EVID - Typ nastalé události (na př. 0x00 pro událost dokončení akvizice)
- VAL - Doplňující data (na př. ID snímku pro událost dokončení akvizice)

**0xFE - Reboot of the Device** : Příkaz pro restartování detektoru.

**Odchozí data:** nic

**Příchozí data:** nic

**0xFF - Shutdown of the Device** : Příkaz pro vypnutí detektoru.

**Odchozí data:** nic

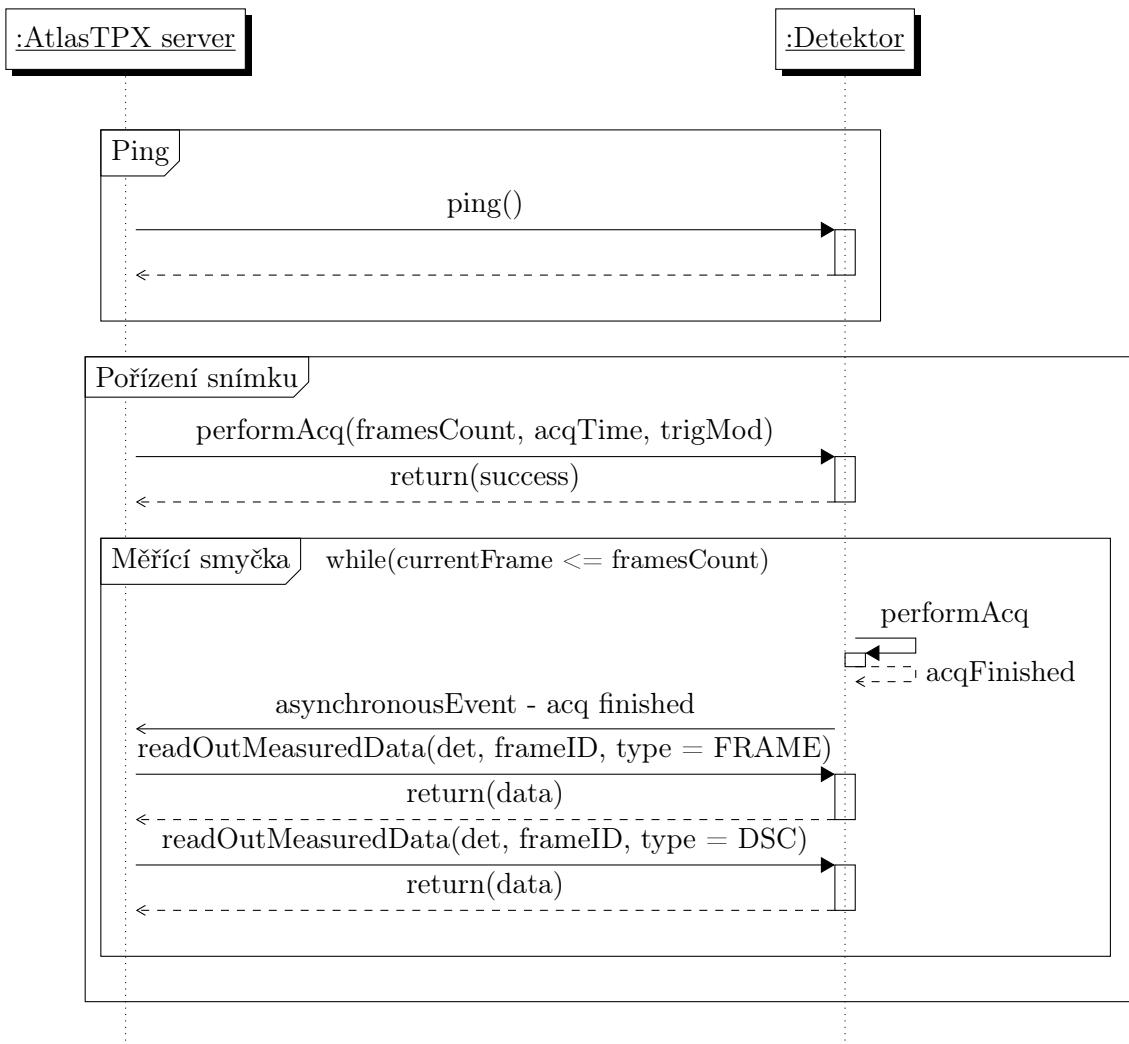
**Příchozí data:** nic

#### 4.4.1.3 Synchronní a asynchronní příkazy komunikačního protokolu

Téměř všechny příkazy komunikačního protokolu jsou synchronní, tzn. server vyšle odchozí paket (s typem příkazu a jeho daty), který detektor zpracuje a bezprostředně vygeneruje a pošle odchozí paket (s příslušnými daty, typem příkazu a informací, zda-li při jeho zpracování došlo k chybě). Příkladem této synchronní komunikace je příkaz ping na obrázku 4.10 nahoře.

Vedle synchronních příkazů existuje i jeden asynchronní - 0xFD - Asynchronous Event From Device. V současné verzi komunikačního protokolu je tento příkaz využit jen pro oznámení serveru, že detektor dokončil akvizici a má data připravena k vyčtení.

Na obrázku 4.10 je znázorněn příklad kombinace synchronní a asynchronní komunikace pro pořízení snímku. Nejprve server vyšle synchronní příkaz s žádostí o provedení akvizice (s parametry: doba akvizice, počet snímků a mód triggeru) na který hned dostane odpověď. Následuje vyčítací smyčka - detektor udělá akvizici snímku (pokud již všechny neudělal) a



Obrázek 4.10: Příklad použití komunikačního protokolu

vyšle serveru asynchronní příkaz s kódem právě dokončené akvizice a s ID<sup>16</sup> snímku. Tuto asynchronní zprávu zachytí a provede vyčítací sekvenci (pomocí příkazu **0x0C - Read Measured Data**), jejíž kroky se mohou lišit dle konfigurace. Ve výchozím nastavení tato sekvence vypadá následovně:

1. Vyčtení vlastního snímku (hodnot jednotlivých pixelů).
2. Vyčtení metadat (tzv. DSC). Tato data obsahují dodatečné informace ke snímku, jako na příklad přesnou dobu akvizice, čas začátku akvizice a měřící napětí a frekvenci Timepix čipů.

<sup>16</sup>unikátní číslo snímku od spuštění detektoru

#### 4.4.1.4 Implementace modulu pro řízení detektorů na straně serveru

V této podkapitole bude popsána implementace řízení detektorů v řídícím softwaru sítě ATLAS TPX. Celý software byl implementován v jazyce JAVA za pomocí build nástroje Maven<sup>17</sup> a knihoven Dropwizard<sup>18</sup>, Retrofit<sup>19</sup> a RxJava<sup>20</sup>.

Jak již bylo zmíněno výše, detektory jsou připojeny k ATLAS TPX serveru přes ethernetové rozhraní a pomocí TPC/IP protokolu je vlastní komunikace realizována pomocí výše popsaného komunikačního protokolu. Z pohledu navazování spojení byla použita architektura klient-server tak, že detektor plní roli serveru a ATLAS TPX server zase klienta. Tato architektura bylo použita s ohledem na robustnost a stabilitu této sítě a aby úloha navazování spojení zůstala v kompetenci ATLAS TPX serveru. Když by na příklad došlo k přerušení napájení nebo jiné chybě jednoho z detektorů, ATLAS TPX server by nebyl schopen se pokusit o znovu navázání spojení.

Na obrázku 4.11 můžete vidět diagram tříd modulu pro práci s detektory řídícího softwaru. Tento diagram není úplný a obsahuje jen několik nejdůležitějších tříd, zbytek je k nalezení na přiloženém CD.

Při návrhu tohoto modulu bylo vycházeno z návrhového vzoru Model-View-Controller [2], resp. z jeho modifikace Model-Controller, protože tento modul žádnou prezentační vrstvu nemá.

Začněme popisem balíčku `model`. Jeho nejdůležitější třídou je třída `DetectorUnit`, která uchovává všechny informace o jednom detektoru, jako na příklad název, tpxID, unitID, ip adresu, port, `SettingsStorage` (úložiště nastavení detektoru, jako na příklad bias, clock, parametry akvizice, DAC hodnoty atd.) a `HwStatus` (tentot objekt nese informace o posledních naměřených údajích získaných z detektoru, jako třeba aktuální hodnoty bias, ping, obecný a měřící status apod.).

Pro uchovávání všech instancí třídy `DetectorUnit` slouží singleton [2] třída `DetectorUnitStorage`, která tyto instance uchovává v kolekci `HashMap`, jejímž klíčem je `TpxID` (`Integer`). Instanci tohoto singletonu je možné získat pomocí třídy `DetectorUnitStorageFactory`, resp. pomocí její statické metody `getStorage()`. Před prvním použití je třeba nejprve toto úložiště inicializovat pomocí statické metody `DetectorUnitStorageFactory.initStorage(String initFilePath)`, které se coby parametr předá cesta v souborovém systému k `".csv"`souboru s tabulkou s detektory. Každý řádek tohoto souboru reprezentuje jeden detektor a je v následujícím formátu:

```
název_detektoru;ip_adresa;unit_id;tpx_id;port
```

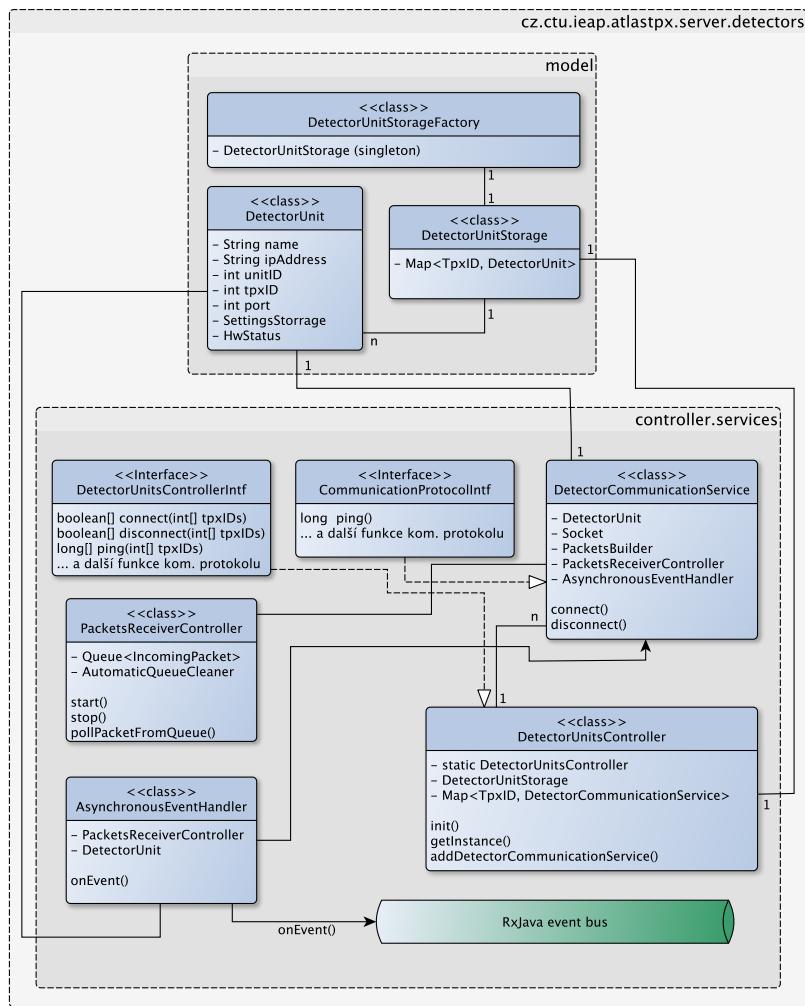
V balíčku `controller.services` se nachází vlastní logika tohoto modulu. Třída `DetectorCommunicationService` se stará o vlastní komunikaci s detektorem (jedna instance = jeden detektor). Tato třída obsahuje instanci třídy `DetectorUnit` (předané v konstruktoru), která mimo jiné obsahuje IP adresu a port příslušného detektoru. Tyto parametry se používají v metodě `connect()`, ve které se vytvoří spojení s detektorem (socket, `BufferedOutputStream`, `BufferedInputStream` apod.). Pro odpojení detektora zase slouží metoda `disconnect()`.

<sup>17</sup><<https://maven.apache.org/>>

<sup>18</sup><<http://www.dropwizard.io>>

<sup>19</sup><<http://square.github.io/retrofit/>>

<sup>20</sup><<https://github.com/ReactiveX/RxJava>>



Obrázek 4.11: Diagram tříd modulu pro ovládání detektorů

Krom metod pro síťovou komunikaci obsahuje i všechny metody implementované z interface `CommunicationProtocolIntf` - všechny synchronní metody komunikačního protokolu.

Pro příjem příchozích paketů z detektoru vznikl `PacketsReceiverController`. Ten ve vlastním vlákně vyčítá proud bytů, přicházejících z detektoru a následně provádí jejich parsování, jehož výsledkem je instance objektu `IncomingPacket`, které obsahuje typ příkazu, příznak chyby a vlastní data. Takto vzniklý paket je zařazen do fronty příchozích paketů detektoru (implementované jako `ConcurrentLinkedQueue<IncomingPacket>`). Nad touto frontou pracují jednak všechny metody se synchronními příkazy komunikačního protokolu (které sledují, zda-li se v ní v daném časovém intervalu objeví odpověď), ale také instance objektu `AsynchronousEventHandlerController`. Ten ve vlastním vlákně tuto frontu sleduje a objeví-li se paket s typem příkazu `0xFD` (`Asynchronous Event From Device`), tak ho z fronty odebere, jeho vlastní data rozparsuje a dále zpracuje. Pokud se na příklad jedná o událost dokončené akvizice, tak `ReadOutService` snímek z detektoru vyčte a pomocí RxJava event

`bus` vyšle asynchronní zprávu napříč celou aplikací s vyčteným snímkem. Zde byl použit návrhový vzor Producer - Consumer, kde `AsynchronousEventHandlerController` představuje Producer a na kterémkoliv jiném místě aplikace se Consumer (možno i více Consumerů) může zaregistrovat ke sledování událostí v event bus. Příkladem takového Comsumera může být `FrameSaverController`, který se postará o uložení získaného snímku (viz kapitola 4.4.3).

Aby bylo možné pohodlně ovládat více detektorů současně, vznikl `DetectorUnitsController`. Ten implementuje interface `DetectorUnitsControllerIntf`, jež metody pokrývají veškerou funkcionalitu detektoru, jako třeba metody k navázání a ukončení spojení, ale také všechny metody komunikačního protokolu. Všechny tyto metody mají jeden společný parametr - `int[] tpxIDs` (pole TpxID detektorů). Tento parametr určuje, nad kterými detektory se má daná metoda hromadně vykonat. Jejich výstupem je zase pole, jehož typ se liší dle příkazu (na příklad pro metodu connect je to pole boolean proměnných - značící úspěch připojení, pro ping zase double čísel, udávajících odezvu spojení v ns). Tento objekt je zase typu singleton a jeho instanci lze získat pomocí metody `getInstance()`. Po spuštění aplikace je však třeba tento singleton nainicializovat pomocí metody `init()`, která pomocí `DetectorUnitStorageFactory` vytvoří datovou strukturu s `DetectorCommunicationService` jednotlivých detektorů.

#### 4.4.1.5 Emulátor detektoru

Pro účely vývoje a testování řídícího software pro ATLAS TPX server byl vyvinut emulátor detektoru, který plně emuluje jeho činnost. Emulátor byl rovněž napsán v jazyce JAVA a jeho zdrojové kódy a spustitelný `JAR` soubor naleznete na přiloženém CD.

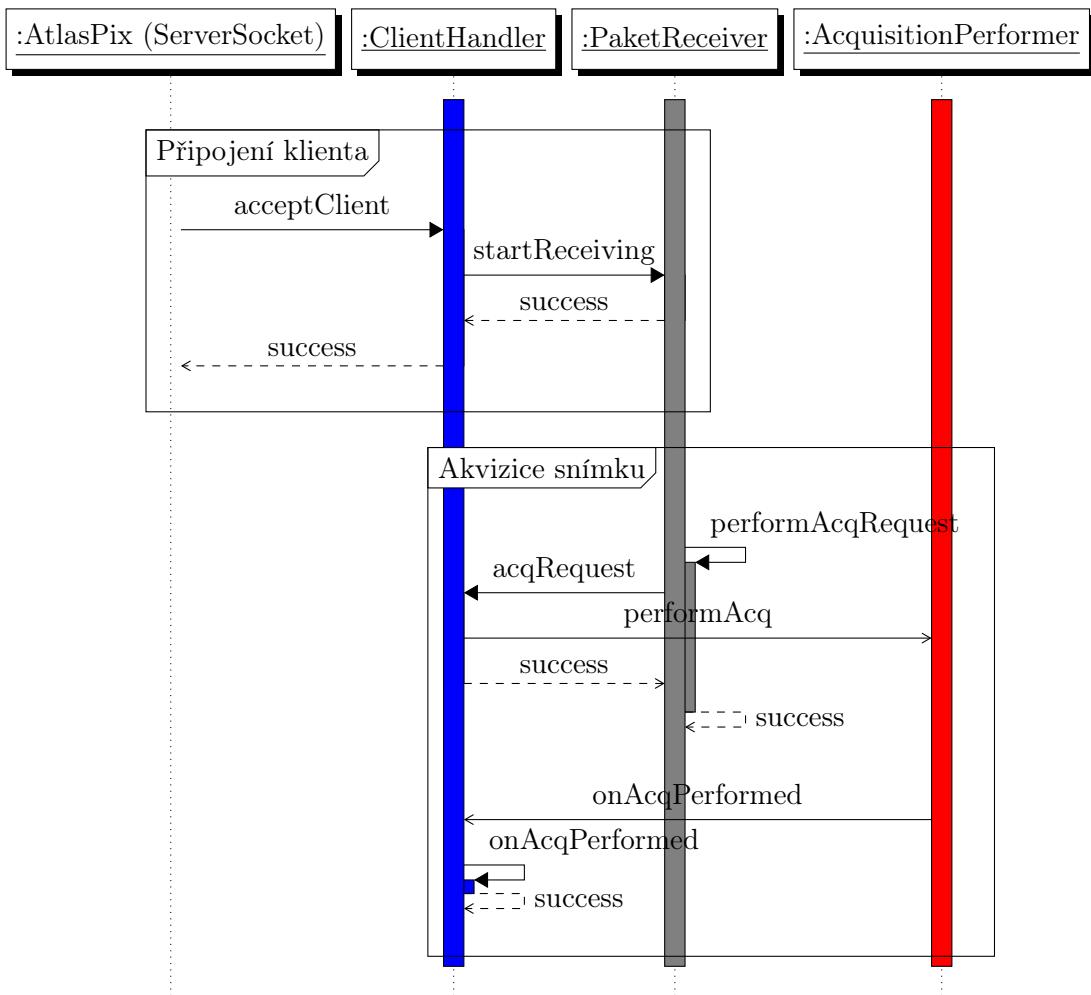
Emulátor se spouští se dvěma povinnými parametry, kde

1. je port, na kterém server emulátoru naslouchá (celé číslo),
2. je pravděpodobnost, s jakou při zpracovávání požadavku bude simulována chyba - 0 znamená bez chyb, 1 samé chyby (desetinné číslo).

Na obrázku 4.12 je zobrazen sekvenční diagram dvou případů užití tohoto emulátoru. První případ užití znázorňuje připojení klienta (na př. ATLAS TPX serveru) a druhý pak proces zpracování žádosti o snímek a jeho vygenerování.

Pojďme si nyní popsat první část - připojení klienta. Ve třídě `AtlasPix`, resp. v její metodě `startServer()` se nachází nekonečná smyčka, ve které je pomocí instance třídy `ServerSocket` navázáno spojení s novými klienty. Když spojení s klientem je navázáno, je vytvořen klientský socket, pomocí kterého dojde v vytvoření instance třídy `ClientHandler`. Ta se v samostatném novém vláknu stará o obsluhu nově připojeného klienta. Zároveň `ClientHandler` vytvoří instanci třídy `PaketReceiver`, který čte proud bytů přijatých od klienta a parsuje jej do objektů typu `IncomingPacket`.

Tím se dostáváme k druhému příkladu případu užití - akvizici snímku. Poté, co `PaketReceiver` přijme nový paket, tak ho vloží do fronty paketů. Tuto frontu z druhé strany čte a zpracovává `ClientHandler`. Všechny příchozí pakety od klienta mohou obsahovat jen synchronní příkazy komunikačního protokolu, takže může být okamžitě nasimulován příslušný stav emulátoru a vygenerována odpověď klientovi.



Obrázek 4.12: Emulátor detektoru - sekvenční diagram připojení klienta a pořízení snímku

V našem případě akvizice snímku je situace trochu složitější. **PaketReceiver** odchytí nový paket, obsahující žádost o provedení akvizice, který předá do fronty příchozích paketů. Když se **ClientHandler** dostane k jeho zpracování, tak vytvoří objekt typu **AcquisitionRequest** a předá ho instanci třídy **AcquisitionPerformer**. Ten žádost o akvizici zařadí do příslušné fronty, kde čeká, až samostatné akviziční vlákno se dostane k jejímu zpracování. Při zpracovávání žádosti o akvizici je vygenerováno náhodná matice hodnot pixelů (o velikosti  $256 \times 512$ ) a také příslušná metadata, jako na příklad akviziční čas, bias obou čipů (s přičtenou náhodnou veličinou, simulující fluktuaci napětí na křemíkovém povrchu detekčních čipů) apod. Po vygenerování těchto hodnot se akviziční vlákno uspí na dobu akvizice, aby se chování emulátoru co nejvíce přiblížilo skutečnému detektoru.

Následně je připojenému klientovi poslána asynchronní zpráva o dokončené akvizici. Tato zpráva obsahuje příznak, že naměřená data jsou připravena k vyčtení a také ID vygenerovaného snímku.

Poté klient pošle příkaz pro vyčtení naměřených dat (resp. dva příkazy - jeden pro vlastní

snímek a druhý pro metadata), což bylo popsáno v kapitole 4.4.1.3 - viz obr. 4.10.

#### 4.4.2 REST API server

Řídící software se svému ovládání poskytuje rozhraní přes JSON REST<sup>21</sup> API. Pro toto rozhraní byla použita knihovna Dropwizard<sup>[3]</sup>, která vznikla složením několika dalších knihoven, zejména pak:

**Jetty** je open-source software, v současné době vyvíjen Eclipse Foundation. Tato knihovna obsahuje Java HTTP webový server, který na rozdíl od běžných webových serverů (které většinou přes HTTP protokol poskytují soubory koncovým uživatelům) byl navržen pro strojově orientovanou komunikaci.

**Jersey** je open-source knihovna (která dle [3] vychází z JAX-RS<sup>22</sup>) a byla použita pro zajištění REST API. Tato knihovna umožňuje elegantně pomocí anotací mapovat HTTP dotazy na jednoduché Java objekty.

**Jackson** je výkonný nástroj pro práci s JSON<sup>23</sup> objekty v jazyce Java. Tato knihovna umožňuje ergonomicky mapovat data v JSON do Java objektů pomocí anotací.

V této podkapitole bude popsána implementace této knihovny do ATLAS TPX serveru a její provázanost na modul pro řízení detektorů (4.4.1.4).

##### 4.4.2.1 Konfigurace a spuštění serveru

Server je možné spustit z příkazové řádky pomocí příkazu "`java -jar atlasTpXServer.jar server config.yml`", kde `atlasTpXServer.jar` je spustitelný binární soubor serveru, `server` je povinný parametr (znamenající spuštění programu v módu server) a `config.yml`, což je také povinný parametr, udávající cestu v souborovém systému ke konfiguračnímu souboru serveru.

V příloze A naleznete strukturu tohoto konfiguračního souboru, ve kterém je možné nastavit následující:

###### Umístění tabulky s detektory (detectorsConfigPath - viz A řádek 8)

Tento parametr udává cestu v souborovém systému ke konfiguračnímu `*.csv`souboru, obsahující tabulkou všech detektoru a jejich parametrů, popsanou v 4.4.1.4.

###### Nastavení webového serveru (server - viz A řádek 11)

V rámci tohoto objektu je možné nastavit použitý typ protokolu spojení (HTTP/HTTPS), port, počty vláken a další.

---

<sup>21</sup>z angl. Representational State Transfer

<sup>22</sup><<http://jcp.org/en/jsr/detail?id=311>>

<sup>23</sup>JSON je v dnešní době standardem pro zápis a výměnu strukturovaných, člověkem i strojem čitelných dat.

#### Logování (logging - viz [A](#) řádek 30)

Pomocí tohoto parametru je možné nastavit úroveň a cíl vytvářených logů. V přiloženém konfiguračním souboru bylo použito dvojího logování a to na standardní výstup a do souboru (oba úrovně **INFO**). Při logování do souboru je možné navíc nastavit automatickou archivaci.

#### Automatické vyčítání snímků (readOutDataAutomatically - viz [A](#) řádek 54)

Tento parametr typu boolean udává, když přijde asynchronní událost z detektoru o datech připravených k vyčtení, zda-li budou vyčtena automaticky (při hodnotě parametru **true**), nebo manuálně (při hodnotě **false**).

#### Výstupní adresář (outputDir - viz [A](#) řádek 57)

Parametr udávající cestu v souborovém systému pro ukládání dat z detektorů.

#### Formát výstupních dat (outputFramesType - viz [A](#) řádek 68)

Tento parametr obsahuje pole výstupních formátu dat, ve kterých získaná data z detektorů budou ukládána. V této verzi programu je podporovaný jediný formát - **MULTIFRAME**.

#### Konfigurace data serveru (v [A](#) od řádku 77)

Na tomto místě je možné nastavit parametry (url a port) serveru, sloužícího pro příjem a zpracování naměřených dat a zda-li má být použit. Více o tomto serveru bude zmíněno v [4.4.3.1](#).

### 4.4.2.2 Metody poskytované serverem

Tento JSON REST API server poskytuje metody pro získání stavových ATALS TPX serveru, ale také pro jeho řízení, zejména pak pro ovládání detektorů. Seznam všech těchto metod s jejich popisem je k nalezení v dokumentaci na přiloženém CD.

URL schéma jednotlivých metod je následující - protokol://host:port/vx/skupina/metoda, kde

**protokol** je použitý protokol (HTTP/HTTPS),

**host** je doména, či IP adresa serveru,

**port** - port serveru,

**vx** je verze API, na které komunikace bude probíhat (současná verze je "v1"),

**skupina a metoda** udávají cestu a název metody (u některých metod stačí zadat pouze skupinu).

Server byl implementován za použití následujících standardů:

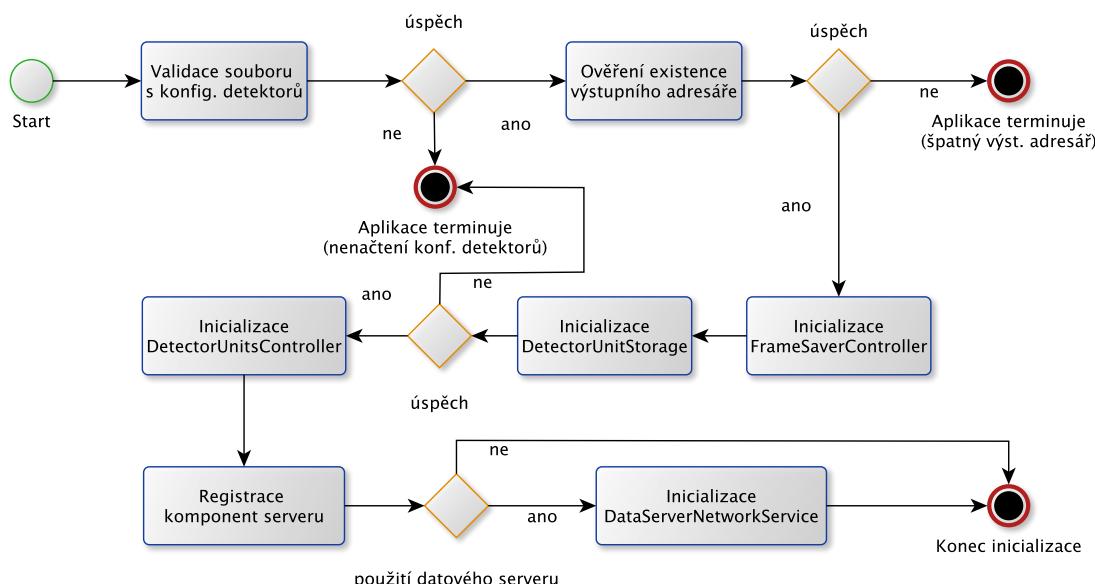
- Všechny zde popsané metody jsou založeny na HTTP metodě POST.
- Všechna předávaná data jsou ve formátu JSON.

- Všechny časy a data jsou reprezentovány v milisekundách od 1.1.1970 (tzv. UNIX Time) a časové intervaly v sekundách, není-li specifikováno jinak.
- Server může vracet zprávy s následujícími HTTP kódy: 200 (OK), 400 (Bad Request), 404 (Not Found) a 500 (Internal Server Error).

#### 4.4.2.3 Implementace serveru

Jak již bylo zmíněno výše, server byl implementován za použití knihovny Dropwizard a veškeré s ním související třídy jsou v balíčku `rest_server`.

V tomto balíčku se nachází třída `RestServerApplication`, která obsahuje statickou metodu `main()`, která slouží ke spuštění celé aplikace. Zároveň tato třída dědí od `io.dropwizard.Application` a přepisuje její metodu `run()`, ve které dochází ke konfiguraci serveru, spuštění jeho podpůrných služeb a k registraci komponent REST API serveru (v URL výše uvedených jako skupina).



Obrázek 4.13: BPMN inicializace REST API serveru

Na obrázku 4.13 je znázorněn BPMN<sup>24</sup> diagram procesu inicializace REST API serveru (tělo metody `run()` zmíněné výše). V rámci tohoto procesu nejprve dojde k validaci existence konfiguračního `"*.csv"`souboru, obsahujícím tabulkou s detektory a jeho parametry (viz 4.4.2.1), pokud tato validace selže, program vypíše chybovou hlášku a ukončí se. Následuje ověření platnosti výstupního adresáře (pokud selže, program terminuje). Poté dojde k inicializaci `FrameSaverController`(viz kapitola o zpracování a ukládání dat 4.4.3) a `DetectorUnitStorage` (úložiště všech `DetectorUnit`, již zmíněné v 4.4.1.4). Pokud se toto

<sup>24</sup>z anglicky Business Process Model and Notation

úložiště nepodaří vytvořit (z důvodu chyby parsování tabulky detektorů) aplikace opět termínuje. Dalším krokem je inicializace nástroje pro řízení detektorů (`DetectorUnitsController`) a registrace jednotlivých komponent REST API serveru. Na závěr procesu inicializace dojde k vytvoření spojení s datovým serverem (viz [4.4.3.1](#)) pro odesílání naměřených dat, je-li v konfiguračním souboru serveru povolen.

#### 4.4.3 Zpracování a ukládání dat

Zpracovávání a ukládání dat, resp. získaných snímků z detektoru, bylo navrženo s ohledem modularitu a možnosti rozšíření o podporu nových výstupních formátů. V současné verzi je podporovaný jen jeden formát, a to tzv. **Multiframe**. Tento formát spočívá v ukládání snímků do tří souboru a to:

**Frame File** V tomto souboru je ukládán vlastní snímek - hodnoty jednotlivých pixelů.

Struktura tohoto souboru je taková, že při ukládání snímku jsou po řádcích zapsány indexy a hodnoty jednotlivých pixelů (oddělených znakem tabulátoru a mezerou), které mají nenulovou hodnotu (z důvodu úspory místa). Jednotlivé snímky jsou oddeleny novým řádkem se znakem '#'. Tento soubor má příponu `".txt"`.

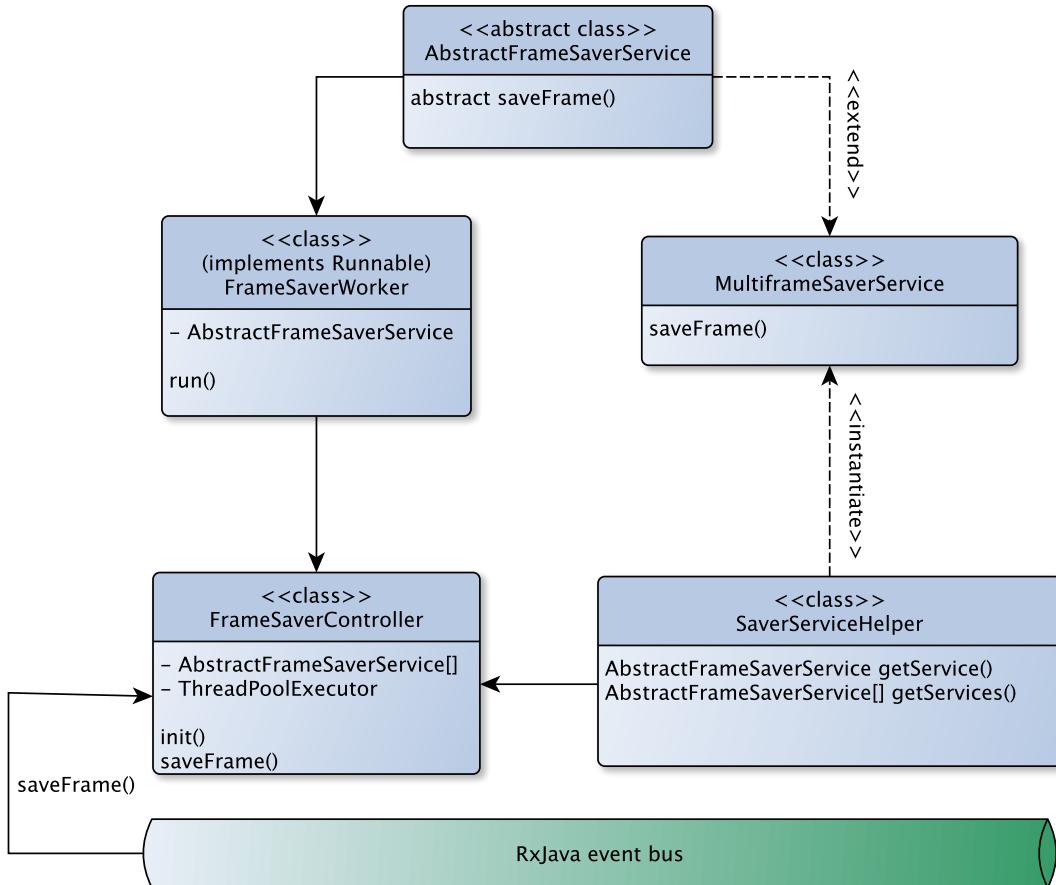
**Description File** Tento soubor nese metadata o snímku, jako na příklad akviziční čas, časové razítka začátku akvizice, ale také hardwarové parametry detektoru v době akvizice (napětí detekčního čipu, měřící frekvenci, hodnoty DAC apod.). Každá snímek v tomto souboru začíná svým ID (začínajícím od nuly, unikátní v rámci souboru), následovaným výčtem parametru, oddelenými prázdnými řádky. Každý parametr je uložen na tři řádky, kde

1. řádek obsahuje název parametru, jeho jednotku, po př. další informace.
2. řádek udává datový typ a velikost vlastní hodnoty parametru - třetího řádku.
3. řádek obsahuje hodnotu parametru.

Přípona tohoto souboru je `".txt.dsc"`.

**Index File** Třetím souborem je tzv. **Index File**. Tento soubor nese binární informace, které spojují **Data File** a **Description File**. Pro každý snímek je to tohoto souboru zapsána bytová adresa začátku snímku v obou výše zmíněných souborech (pro každý 8 B). Z důvodu možného rozšíření do budoucna a zachování zpětné kompatibility, těchto 16 B je následováno dalšími 8 B (s nulovou hodnotou). Přípona tohoto souboru je `".txt.idx"`.

Když ATLAS TPX server odchytí asynchronní událost od nějakého z detektorů o tom, že daný detektor dokončil akvizici a že má data připravena k vyčtení, postará se o jejich vyčtení a vygeneruje objekt typu **Frame**. Tento objekt obsahuje všechna data, vázající se k dané akvizici, jako na příklad hodnoty jednotlivých pixelů, ale i různá metadata (akoviziční čas, bias apod.). Po vygenerování objektu **Frame** je napříč celou aplikací vyslána asynchronní událost o této skutečnosti, obsahující právě vygenerovaný objekt typu **Frame**, pomocí RxJava event bus.



Obrázek 4.14: Diagram tříd komponenty pro ukládání dat ATLAS TPX serveru

Nyní je třeba tuto asynchronní událost odchytit a zpracovat, k tomu slouží singleton třída `FrameSaverController` - viz obr. 4.14. Ta je po spuštění aplikace inicializována (viz 4.4.2.3). Tomuto procesu jsou předány všechny výstupní typy dat, uvedených v konfiguračním souboru 4.4.2.1. Pomocí třídy `SaverServiceHelper` dojde k vytvoření instancí tříd všech příslušných služeb pro ukládání dat (dědících od třídy `AbstractFrameSaverService`).

Jelikož každý z 16 detektorů může generovat několik snímků za sekundu, bylo třeba vyvinout robustní řešení, které se bude přizpůsobovat dané zátěži. Za této účely byl implementován `ThreadPoolExecutor`, který sleduje a vykonává frontu úkolů (instancí třídy `FrameSaverWorker`, implementujících interface `Runnable`). Dle velikosti této fronty, resp. podle počtu snímků čekajících na uložení, je počet exekucích vláken dynamicky měněn (ve stávající konfiguraci od 3 o 50).

Nyní se vratme ke příchodu asynchronní události s vyčteným snímkem. Když je tato událost zachycena, dojde vytvoření instance třídy `FrameSaverWorker` pro každou ukládací službu (dědící od `AbstractFrameSaverService`, na př. `MultiframeSaverService`) a následném zařazení této instance do fronty, kterou obsluhuje již výše zmíněný `ThreadPoolExecutor`.

#### 4.4.3.1 Datový server

Současně s možností ukládání dat ATLAS TPX serverem na lokální, či síťové úložiště (jak bylo popsáno výše) existuje i druhý způsob nakládání s pořízenými daty - jejich přímé odesílání datovému serveru (viz 4.3), pomocí jeho API, což bude předmětem této podkapitoly.

Přehled všech metod s jejich technickými specifikacemi je k nalezení v dokumentaci na přiloženém CD, zde bude uveden jen jejich stručný popis.

Kvůli optimalizaci přenášenému obejmu dat byl navržen úsporný komunikační protokol, který odstraňuje redundanci přenášených dat (oproti Multiframe formátu, popsaném výše). Toho je dosaženo především tím, že s každým snímkem jsou přenášena jen data, přímo související a unikátní k dané akvizici. To jsou zejména vlastní hodnoty jednotlivých pixelů, časové razítka začátku akvizice, napětí v době akvizice na polovodičovém povrchu detekčních čipů apod. Mezi údaje, které se v průběhu akvizice nemění (a tudíž je zbytečné je s každým snímkem přenášet) patří konfigurace pixelů, měřící frekvence, hodnoty DAC apod.

**Metoda Status** Metoda status slouží pro získání základních informací datového serveru, jako třeba volné místo v jeho úložišti, ale také informaci o přenesených snímcích (celkový počet, zpracováno, atd.).

**Metoda pro odeslání snímků** V rámci této metody je přenesen vlastní snímek a příslušná metadata s ním související, uvedená výše. Zde je dobré zmínit, že oproti formátu Multiframe jsou vyloučeny pixely ne s nulovou, nýbrž s nejčetnější hodnotou. Tím je docíleno minimalizace počtu přenesených pixelů. Tento mechanismus je účinný především na přeexponované snímky. Jako jeden z parametrů se rovněž odesílá i ID konfigurace, získané z datového serveru - viz další bod.

**Metoda pro odeslání konfigurace** Touto metodou je přenášena statická konfigurace jednotlivých detektorů. Tuto metodu je třeba provést nad všemi detektory před zahájením odesílání snímků a potom až při změně některého z parametrů konfigurace. Výstupem této metody je totiž seznam ID, které byly jednotlivým konfiguracím přiděleny datovým serverem, které je třeba datovému serveru poslat společně se snímkami.

Spojení s tímto serverem zajišťuje modul `data_sender`, resp. jeho třída `DataSenderController`. Jedná se opět o singleton, který při své inicializaci vytvoří dvě fronty - frontu snímků a frontu konfigurací (obě implementované, jako `ConcurrentLinkedQueue`) a také vytvoří mapu (mapující ID interních konfigurací detektorů na ID, získané z datového serveru). Zároveň dojde ke spuštění samostatného vlákna, které tyto fronty sleduje a jejich položky odebírá a posílá datovému serveru.

Odesílání dat datovému serveru je realizováno pomocí knihovny `Retrofit`<sup>25</sup>, který poskytuje nástroje pro implementaci jednoduchého HTTP klienta.

---

<sup>25</sup><<http://square.github.io/retrofit/>>

## Kapitola 5

### Závěr



# Literatura

- [1] BOUCHAMI, J. et al. Estimate of the neutron fields in ATLAS based on ATLAS-MPX detectors data. *Journal of Instrumentation*. 2011, 6, 01, s. C01042. Dostupné z: <<http://stacks.iop.org/1748-0221/6/i=01/a=C01042>>.
- [2] GAMMA, E. et al. *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1995. ISBN 0-201-63361-2.
- [3] Kolektiv autorů Dropwizard. *Dropwizard online dokumentace - Getting Started* [online]. 2016. [cit. 16. 5. 2016]. Dostupné z: <<http://www.dropwizard.io/0.9.2/docs/getting-started.html>>.
- [4] S. Ballestrero, A. Bogdanchikov , F. Brasolin, A. C. Contescu, S. Dubrov, M. Hafeez, A. Korol , C. J.Lee, D. A. Scannicchio, M. Twomey, A. Voronkov, A. Zaytsev. *ATLAS TDAQ application gateway upgrade during LS1* [online]. 2014. [cit. 7. 5. 2016]. Dostupné z: <<https://cds.cern.ch/record/1664275/files/ATL-DAQ-SLIDE-2014-054.pdf>>.
- [5] TURECEK, D. et al. Remote control of ATLAS-MPX Network and Data Visualization. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*. 2011, 633, Supplement 1, s. S45 – S47. ISSN 0168-9002. doi: <http://dx.doi.org/10.1016/j.nima.2010.06.117>. Dostupné z: <<http://www.sciencedirect.com/science/article/pii/S0168900210013070>>. 11th International Workshop on Radiation Imaging Detectors (IWORID).
- [6] TURECEK, D. Software for Radiation Detectors Medipix. Master's thesis, Czech Technical University in Prague, Czech Republic, 2011.
- [7] Vladimír Wagner. *Článek: Jak se daří urychlovači LHC* [online]. 2009. [cit. 4. 5. 2016]. Dostupné z: <[http://hp.ujf.cas.cz/~wagner/popclan/lhc/lhc\\_rok2010.htm](http://hp.ujf.cas.cz/~wagner/popclan/lhc/lhc_rok2010.htm)>.
- [8] VYKYDAL, Z. et al. The Medipix2-based network for measurement of spectral characteristics and composition of radiation in {ATLAS} detector. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*. 2009, 607, 1, s. 35 – 37. ISSN 0168-9002. doi: <http://dx.doi.org/10.1016/j.nima.2009.03.104>. Dostupné z: <<http://www.sciencedirect.com/science/article/pii/S0168900209005956>>. Radiation Imaging Detectors 2008Proceedings of the 10th International Workshop on Radiation Imaging Detectors.

*LITERATURA*

---

## Příloha A

# Konfigurační soubor ATLAS TPX serveru

```
1 #####  
2 # Basic configuration  
3 #####  
4 # Path to config csv file with detectors config  
5 # File structure: 1 detector struct per line  
6 # - detector_name[String];detector_ip[String];unit_id[int];tpx_id[int]  
7 # ;port[int]  
8 detectorsConfigPath: data/detectors2.csv  
9  
10 # Control REST API server configuration  
11 server:  
12     applicationConnectors:  
13         - type: http  
14             port: 9179  
15             outputBufferSize: 32KiB  
16             idleTimeout: 30 seconds  
17             minBufferPoolSize: 64 bytes  
18             bufferPoolIncrement: 1KiB  
19             maxBufferPoolSize: 64KiB  
20             acceptorThreads: 1  
21             selectorThreads: 2  
22             acceptQueueSize: 1024  
23             reuseAddress: true  
24             soLingerTime: 600s  
25     adminConnectors:  
26         - type: http  
27             port: 9180  
28  
29 # Logging settings  
30 logging:
```

```

31   level: INFO
32   appenders:
33     - type: file
34       currentLogFilename: log/atlastpx_server_app.log
35       threshold: ALL
36       archive: true
37       archivedLogfilenamePattern: log/atlastpx_server_app-%d.log.gz
38       archivedFileCount: 5
39       timeZone: UTC
40
41 logging:
42   level: INFO
43   appenders:
44     - type: console
45       threshold: ALL
46       timeZone: UTC
47       target: stdout
48
49 #####
50 # Data readout & saving configuration
51 #####
52 # If true asynchronous event handler will takes care about automatical data
53 # (frame + DSC) readout from the detector.
54 readOutDataAutomatically: true
55
56 # Directory for saving output data
57 outputDir: ../frames
58
59 # Local data saving configuration
60 # Allowed types:
61 # - MULTIFRAME: Pixelman's multiframe format (3 files per hour).
62 #               Output files:
63 #               1) Frames: Line [X, C], where X is pix. index and
64 #                  C pix value. Frames are separated by '#'
65 #               2) Description file ( DSC)
66 #               3) Index file (indexing file 1 and 2)
67 # Example - outputFramesType: [MULTIFRAME]
68 outputFramesType: [MULTIFRAME]
69
70 #####
71 # Data server configuration
72 #####
73 # If true all data (configs on change, performed frames) will be automatically
74 # send to the data server
75 # and also data server status will be available by info endpoint.
76

```

---

```
77 useDataServer: true
78
79 # URL of dataServer
80 dataServerBaseUrl: atlastpx.utef.cvut.cz
81
82 # port of dataServer
83 dataServerPort: 8042
```

Zdrojový kód A.1: Konfigurační soubor ATLAS TPX serveru

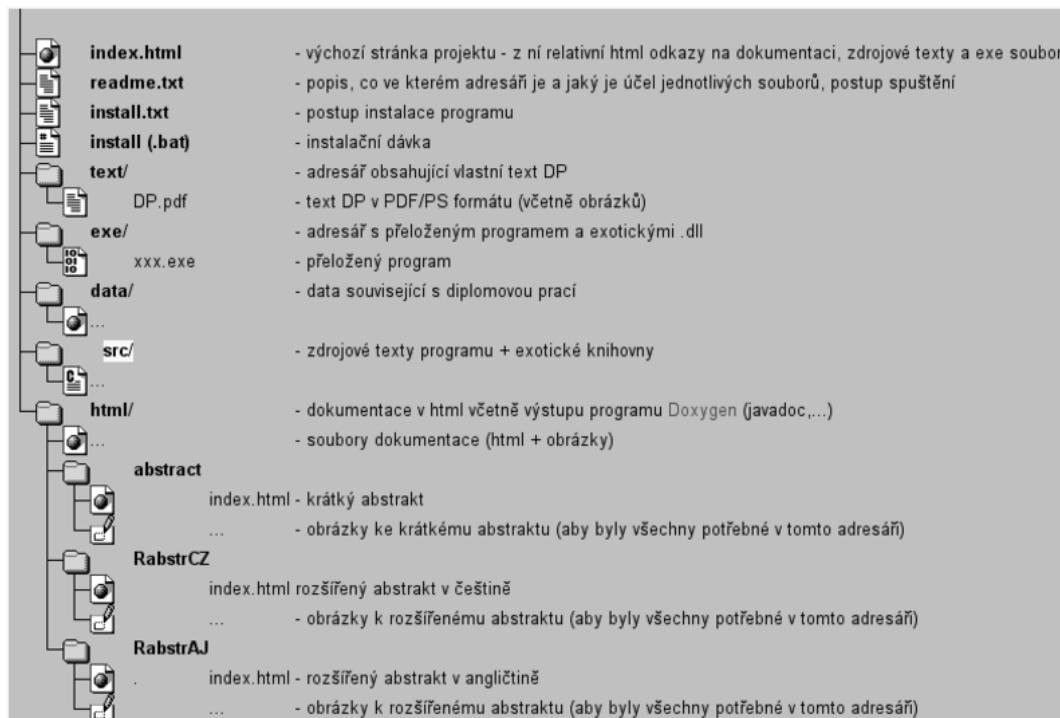


## Příloha B

# Obsah přiloženého CD

Tato příloha je povinná pro každou práci. Každá práce musí totiž obsahovat přiložené CD. Viz dále.

Může vypadat například takto. Váš seznam samozřejmě bude odpovídat typu vaší práce...  
[6] fasfassaafsaf



Obrázek B.1: Seznam přiloženého CD — příklad

Na GNU/Linuxu si strukturu přiloženého CD můžete snadno vyrobit příkazem:  
\$ `tree . >tree.txt`  
Ve vzniklém souboru pak stačí pouze doplnit komentáře.

## *PŘÍLOHA B. OBSAH PŘILOŽENÉHO CD*

---

Z **README.TXT** (případne index.html apod.) musí být rovněž zřejmé, jak programy instalovat, spouštět a jaké požadavky mají tyto programy na hardware.

Adresář **text** musí obsahovat soubor s vlastním textem práce v PDF nebo PS formátu, který bude později použit pro prezentaci diplomové práce na WWW.