Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Simulační model železničního provozu a jeho využití pomocí API

Roman Štěpánek

Bakalářská práce

2019

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 10. 5. 2019

Roman Štěpánek

PODĚKOVÁNÍ

Rád bych touto cestou poděkoval vedoucímu své bakalářské práce doc. Ing. Michaelu Bažantovi, Ph.D. za podnětné připomínky, cenné rady a odborné vedení.

ANOTACE

Bakalářská práce je zaměřena na vytvoření základního modelu železničního provozu bez složitějších mechanik, a především pak na realizaci API s použitím uživatelsky definované nadstavby.

V teoretické části práce jsou popsány základní principy modelování a simulací, je zde současně charakterizován simulační nástroj OpenTrack, podpořený ukázkami jeho uplatnění. Další část práce je věnována seznámení s aplikační nadstavbou a popsání komunikace a celkové funkčnosti se simulačním nástrojem. Součástí teoretické části práce jsou také ukázky nastiňující možné uplatnění API v oboru simulování železniční dopravy.

Praktická část práce popisuje použití aplikační nadstavby na konkrétní model železničního úseku. Zahrnuje rovněž rozbor aktuálního vytížení železniční stanice a základní statistické vyhodnocení zaměřené na konkrétní řešenou problematiku.

KLÍČOVÁ SLOVA

mikrosimulace, simulační studie, OpenTrack, vlaková doprava, dopravní inženýrství, železniční síť, dopravna

TITLE

Simulation model of railway traffic and its use by API

ANNOTATION

The bachelor thesis is focused on creating a basic model of railway traffic. It is especially focused on implementation of API using user-defined superstructure without adding any complicated mechanics to it.

In the theoretical part of the thesis there are described the basic principles of modeling and simulation, there is also described the simulation tool OpenTrack, supplemented by its utilization. The next part of the thesis is devoted to the introduction with application superstructure and to the description of communication and overall functionality of simulation tool. The theoretical thesis includes examples outlining the possible application of API in the field of simulation of the railway transport.

The practical part describes the utilization of the application superstructure to certain model of the railway section. It also includes an analysis of the current workload of the railway station and a basic statistical evaluation focused on the specific problematic.

KEYWORDS

microsimulation, simulation studies, OpenTrack, railway transport, traffic engineering, rail network, train set

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM UKÁZEK KÓDU

SEZNAM GRAFŮ

SEZNAM ZKRATEK

AN	aplikační nadstavba
API	Application Programming Interface
EoA	End of Authority
ETH	Eidgenössische Technische Hochschule
JVM	Java Virtual Machine
MA	Movement Authority
OT	OpenTrack
PDF	Portable Document Format

ÚVOD

V současné době simulační nástroje v železniční dopravě téměř nepodporují aplikační nadstavbu s možností zasahovat do simulace při jejím vykonávání. Mnohdy tak simulační nástroj neumožňuje implementaci složitějších mechanik, které by přiblížily simulační model reálnému systému.

Z tohoto důvodu se autor práce rozhodl prozkoumat využití simulačního programu OpenTrack. Tento nástroj má širokou škálu uplatnění pro libovolnou kolejovou dopravu. Lze v něm efektivně simulovat železniční dopravu, metro či tramvajovou dopravu. Schopnost implementovat vlastní API, která následně dokáže simulaci dynamicky řídit, je však skutečným potenciálem, který si bere autor za cíl.

Jakákoli nadstavba usnadňuje proces simulování a jejím cílem je zaměřit se na konkrétní části simulačního modelu, nebo například realizovat krizové řízení systému, na které simulátor nevlastní nástroje. Nejprve bude provedena analýza možného využití aplikační nadstavby a následně bude realizována nad konkrétním modelem se zaměřením na zvolenou problematiku. V závěru práce bude uskutečněno statistické vyhodnocení dokládající využitelnost aplikační nadstavby programu OpenTrack.

1 MODELOVÁNÍ A SIMULACE

Simulace je výzkumná technika či metoda, jejímž smyslem je nahradit zkoumaný dynamický systém jeho simulátorem. To je prováděno s cílem získat, za pomoci experimentů nad simulátorem, informace o originálním dynamickém systému [3].

1.1 Modelování a simulace v železniční dopravě

Železniční dopravní síť je dynamický systém, který nelze separovat. Neumožňuje oddělit své dílčí prvky, které by pak bylo možné zvlášť analyzovat. Vždy je zapotřebí započítat vnější vlivy systému, ovšem nelze realizovat simulaci na celém kontinentu, a právě z tohoto důvodu je zapotřebí provést abstrakci okolí a stanovit hranice, které jsou dělícím prvkem. Je ale zapotřebí zachování vnějších vlivů [6, 11, 12].

Simulace jako taková se stává jedinou možností pro ověření změn před samotným aplikováním na systému, a proto je výzkumná technika simulace v železniční dopravě široce využívanou. Jakoukoli plánovanou změnu je třeba nejprve verifikovat za pomoci simulace, a tak ověřit její funkčnost v rámci systému [6, 11].

1.2 Důvody pro tvorbu API nadstavby

Simulačních nástrojů je celá řada. Některé jsou zaměřené obecnějším směrem (Arena Simulation Software), jiné zase na konkrétní problematiku (ICAP/4, LTspice IV, Micro-Cap, OpenTrack - Simulation of Railway Systems). Simulační programy s širší škálou uplatnění jsou jen stěží použitelné na specifika vznikající v reálném světě, a tak po nich nelze vyžadovat realizaci náročnějších mechanik. Oproti tomu simulační nástroje s daným zaměřením postrádají široké uplatnění, které v závěru stejně ani není zapotřebí. Ceněným přínosem je v tomto případě především pokrytí konkrétní problematiky ve zvolené oblasti [7, 12, 15].

I přes to, že jsou využívány simulační programy zaměřené na konkrétní problematiku, mohou vznikat požadavky na software, které simulační nástroj není schopen poskytnout. Zejména pokud je vyžadováno zasahovat do simulace při jejím běhu. Proto některé simulační nástroje umožňují realizovat vlastní API nadstavbu. Cílem této práce je právě realizace této nadstavby nad simulačním nástrojem OpenTrack - Simulation of Railway Systems [12, 15].

2 SIMULAČNÍ NÁSTROJ OPEN TRACK

V této kapitole je představen simulační nástroj OpenTrack. Především je zde věnován prostor možnostem jeho uplatnění. Příklady jsou podpořeny konkrétními pracemi, které samy o sobě dokazují širokou škálu uplatnění tohoto simulačního nástroje.

Dále jsou v této části popsány základní nástroje tohoto programu, které budou využívány v praktické části práce. Tato bakalářská práce se nezaměřuje na samotnou realizaci simulace a z toho důvodu budou některé části tvorby simulačního modelu popsány pouze rámcově. Současně jsou zde uvedeny nástroje programu OpenTrack pro komunikaci s API nadstavbou.

2.1 Základní informace

Simulační nástroj OpenTrack původně vyvinut společností ETH Zurich. Jeho účelem je simulace reálného provozu kolejové dopravy. Dnes je software zastřešen pod společností OpenTrack Railway Technology GmbH, která je oddělena od původního vývojáře ETH Zurich. Společnost OpenTrack vznikla v roce 2006 [12].

Pod kolejovou dopravou se skrývá opravdu velké množství různých druhů železničních systémů a typů samotných vlaků:

- vysokorychlostní vlaky,
- meziměstská doprava,
- nákladní vlaky,
- důlní dráhy,
- metro,
- lehké metro,
- tramvaje,
- osobní dopravník (nekonvenční systém osobní dopravy bez řídícího personálu),
- ozubnicové dráhy / horské dráhy,
- maglev (vlak využívající magnetickou levitaci).

Simulační nástroj OT poskytuje nepřeberné množství informací:

- výpočet doby cesty,
- stabilita a studie proveditelnosti jízdních řádů,
- důkaz potřeb infrastruktury,
- výpočet minimálního času vlaku,
- plánování fází přeměny na pomalou jízdu,

- analýza chování stávajících nebo budoucích trakčních vozidel,
- vyšetřování týkající se používání bezpečnostních systémů,
- vyšetřování chování sítě v případě poruch,
- výkonnost a výpočet energie jízdy vlaku,
- simulace trakčních proudových systémů (pomocí OpenPowerNet) [12].

2.2 Vstupní moduly

OT spravuje vstupní data ve třech základních modulech: kolejová vozidla, infrastruktura a jízdní řád. Zadání těchto vstupních modulů je klíčové pro běh simulace. V rámci prvního modulu je zapotřebí definovat všechna kolejová vozidla. Simulační nástroj vlastní databázi kolejových vozidel s cílem usnadnit definování vlastních lokomotiv. Vymezit nové kolejové vozidlo je nepochybně také umožněno. Databáze popisuje všechny možné typy lokomotiv z hlediska technických specifikací, jako jsou: tahová síla, maximální rychlost, adhezní hmotnost, celková délka a jiné. Další modulem je infrastruktura. Při definování dráhy je zapotřebí specifikovat: délku úseku, gradient, maximální rychlost atd. Posledním modulem jsou údaje o jízdním řádu, které se skládají z informací o pohybu vlaků. Tyto informace zahrnují časy příjezdů a odjezdů, informace o připojení či zastavení a minimální čas zastavení [4, 12].

2.3 Výstupní moduly

Simulační nástroj OT poskytuje možnost komunikace s API, tuto výměnu informací ovšem nelze považovat za tradiční výstupní modul. Nehledě na fakt, že tato možnost není součástí základní verze, a navíc se nejedná o výstup po ukončení běhu simulace. Mimo tuto komunikaci OT zpřístupňuje standardní výstupní data v mnoha dalších formách. Tato data lze specifikovat na základě vazby s vlakem, stanicí či tratí. V rámci informací o vlaku, OT poskytuje: grafy rychlosti, grafy závislosti zrychlení na dráze, návěstidla nedovolující jízdu, zpožděné přípoje či jiná narušení jízdy vlaku všeho druhu.

Další součástí je trať, o které OT nabízí plán nákresných jízdních řádů, diagramů obsazení jednotlivých kolejí a grafického přehledu traťových poměrů. Posledním cílem výstupních dat jsou informace o stanicích, kde lze získat: přehledy vlaků, včetně údajů o časech příjezdů, odjezdů a době pobytu ve stanici. Veškerá tato data lze vyobrazit v libovolném grafu, který je součástí programu OT. Tento simulační nástroj rovněž umožňuje ukládat výstupní data pro tabulkový editor či exportovat do formátu ACSII pro další libovolné zpracování [4, 12].

3 SEZNÁMENÍ S OPENTRACK API

Následující kapitola zevrubně popisuje AN. Dále si bere za cíl odhalit prostor pro realizaci AN na dříve realizovaných simulacích v simulačním nástroji OT, a na základě získaných poznatků poté provést její efektivní aplikování.

3.1 Základní informace

Simulační nástroj OT umožnuje komunikaci s aplikacemi třetích stran. Nejedná se pouze o získávání dat ze strany simulačního nástroje, ale jedná se i o zasílání zpráv a instrukcí, které zasahují do samotného běhu simulace. Lze tak například simulovat dispečink společně s jeho logickými postupy, a tak se přiblížit realitě [4].

Zprávy, které nabízí program OT pro aplikační nadstavbu, jsou odvíjeny od reálného provozu. Simulační nástroj nemůže nabídnout informace, které není možné v reálném světě získat, docházelo by k nekonzistenci simulace a reality [12, 13].

Přesto OT poskytuje celou řadu zpráv o:

- aktuálním stavu vlaku,
- odjezdech a příjezdech na stanici,
- trase a jejím obsazení či uvolnění,
- stavu a běhu simulace.

Nicméně aby se nejednalo pouze o alternativní zobrazení nebo o pouhé vykreslování grafů, které má již simulační nástroj OT v dostatečné míře realizován, umožňuje OT zasílání zpráv, a tím dovoluje zasahovat do běhu simulace. Za pomocí takovýchto signálů se zpřístupní možnost řídit samotnou simulaci a provádět složitější logické operace. Požadavky, které lze zasílat simulačnímu nástroji OT jsou:

- nastavit čas odjezdu či příjezdu vlaku,
- vytvořit nebo odstranit vlak,
- rezervovat trasu pro vlak či jeho rezervaci zrušit,
- spustit, zastavit nebo ukončit simulaci.

Jedná se pouze o výčet základních vlastností. V realitě umožňuje OT mnohonásobně větší škálu vstupních instrukcí a výstupních zpráv [13].

3.2 Příklady použití Open Track s využitím API

Pro kvalitní realizaci AN je zapotřebí analyzovat projekty ze simulačního nástroje OT s cílem získat povědomost o schopnostech a uplatněních toho softwaru. Nelze považovat za samozřejmost, že realizace AN bude vždy vhodnou volbou. OT je kvalitní simulační nástroj a jeho schopnost komunikovat s aplikací třetích stran je stále jenom doplněk jinak skvěle fungujícího celku. Přesto může být často velkým přínosem [12].

3.2.1 Simulace metra

Otázka, která může vyvstat, je, zda nástroj OpenTrack dokáže skutečně efektivně realizovat simulaci metra. Například rovnoběžná (paralelní) rychlost jízdy vlaků je specifikem, které by mohlo realizaci simulace činit potíže.

V bakalářské práci [1] s názvem *Využití softwaru OpenTrack pro simulaci provozu metra* od Tomáše Adamce realizované na Univerzitě v Pardubicích na Dopravní fakultě Jana Pernera se nachází realizace linky C pražského metra. Autor práce zde přistupuje k simulačnímu nástroji OT jako k neprobádanému a bere si za cíl pokusit se vytvořit jeho hodnotný simulační model. Popisuje zde náročnost realizace simulačního modelu pro již existující linku metra oproti plánu nové linky, která umožňuje aplikovat poznatky nabyté zjištěnými výsledky simulace. Především pak pro výstavbu plánované linky D pražského metra. Eventualitou by bylo změnit nevyhovující parametry infrastruktury v čase před začátkem výstavby nebo provést návrh systému dnes tolik zmiňované automatizace jízdy vlaku, která je určitě pohledem do budoucna.

Jedním z nedostatků nástroje OT, které autor v práci zmiňuje, je schopnost optimalizovat jízdu vlaku. Přesněji snížit spotřebu trakční energie při zachování včasného dojezdu do stanice dle jízdního řádu. To vše provádět za běhu samotné simulace. Nástroj OT umožňuje zjistit optimum mezi minimální jízdní dobou a minimální spotřebou, ovšem dovede tak pouze z celkového vyhodnocení simulace, nelze provádět operace dynamicky při samotném běhu simulace. Toto je adekvátní pozice pro realizaci nadstavby, která by uskutečňovala potřebné výpočty a pracovala se simulačním nástrojem při jejím běhu. Dynamicky by prováděla výpočty, které by vedly k výraznější nižší spotřebě trakční energie.

V závěru práce jsou porovnávány výsledky výpočtu simulace a skutečného průběhu jízdy vozidla. Autor vytvořil plnohodnotný simulační model odpovídající reálnému systému. Tím adekvátně dokázal schopnost simulačního nástroje OT simulovat metro.

3.2.2 Simulace automatického provozu

Automatizace dopravy je v dnešní době značně skloňované téma. Většinou se uplatňuje na metra a městské či příměstské dráhy. Automatizace s sebou přináší řadu výhod jako například: úspora provozu personálu a zvýšení spolehlivosti systému. Tímto tématem se zabývá diplomová práce toho času bakaláře Erika Tischera [16] s názvem *Simulace automatického provozu na trase metra B*, realizovaná na Dopravní fakultě Jana Pernera na Univerzitě v Pardubicích. Již víme, že OT dokáže simulovat metro, ovšem realizace automatizovaného provozu je v mnoha ohledech rozdílná. Autor výše zmíněné práce si bere za cíl provést simulaci automatického provozu pražského metra na lince B za pomoci simulačního nástroje OT.

Automatizaci dělíme celkem na několik kategorií a hovoříme o nich jako o stupních automatizace. V rámci městských drah hovoříme pak o pěti stupních automatizace (*Grade of Automation* (GoA)). Číslo u zkratky GoA udává její stupeň. Od 0 do 2 hovoříme o konvenčním způsobu provozu a od 3 do 4 o automatickém systému provozu.

Jednotlivými kategoriemi jsou:

- provoz vlaku podle rozhledu – GoA0,
- neautomatizovaný provoz vlaku – GoA1,
- poloautomatizovaný provoz vlaku – GoA2,
- provoz vlaku bez strojvedoucího – GoA3,
- bezobslužný provoz vlaku – GoA4.

Systém jako celek se nemusí v rámci vybavení shodovat na jednom stupni automatizace. Nicméně měly by být vždy vzájemně kompatibilní. Lze provádět postupný přestup k automatizovanému systému. Dříve projektované tratě ani nemohly být navrženy na dnešní požadavky. Automatizaci však lze aplikovat i v takovýchto případech. Nemusíme vždy dosáhnout zcela automatizovaného sytému, bezobslužného provozu vlaku, ale i částečnou inovací můžeme získat řadu výhod.

Cílem autora bylo aplikovat na lince pražského metra B nejvyšší stupeň automatizace GoA4. Aktuálně je linka na stupni GoA1. Autor realizuje i návrh tzv. smíšeného provozu, který by byl aplikován při přestupu z konvenčního na automatický provoz. Cíle byly dosaženy a autorovi se podařilo pomocí nástroje OT vytvořit simulaci automatického i smíšeného provozu na lince B pražského metra. V práci je zmiňováno hned několik požadavků na zkvalitnění zabezpečovacího zařízení a zlepšení vybavení stanic, které by vedly ke zlepšení provozu.

V součinnosti s těmito kroky by jistě stála za zvážení obnova aktuálního vozového parku s vyšší přepravní kapacitou.

V průběhu práce se autor potýkal s několika problémy v rámci aplikování moderních požadavků na již existující systém. Vytvoření automatizovaného provozu na již fungujícím systému s sebou přináší mnohá úskalí a je zřejmé, že bez vynaložení prostředků na zabezpečovací zařízení či na nové vozy je nikdy nepřekonáme, a nedocílíme tak plně automatizovaný bezpečný systém doprav.

3.2.3 Simulace propustnosti výkonnosti tratě

Železniční doprava je jedním z páteřních prvků dopravního systému, ale rovněž i tranzitní dopravy. Historie železniční stanice v České republice zasahuje do minulého i předminulého století. Dá se očekávat, že dnešní požadavky na přepravu se oproti původním záměrům změnily. Největším dopadem v rámci vytíženosti tratě bylo otevření železničního trhu pro osobní dopravu. Tímto tématem a konkrétně propustností na jednom z nejvytíženějších úseků v ČR mezi stanicemi Přelouč a Pardubice hl. n. se zabývá diplomová práce od Pavla Kracíka. Konkrétně se jedná o práci [8] s názvem *Návrh zvýšení propustné výkonnosti tratě 501 v úseku Přelouč – Pardubice hl. n.*, která byla napsána v rámci studia na Univerzitě v Pardubicích na Dopravní fakultě Jana Pernera.

Autor se zaměřuje nejprve na aktuální stav a provádí rovněž analýzu jízdního řádu. Provádí výpočty propustnosti a zjišťuje jeho hranice. Byla zjištěna mnohá úskalí aktuálního stavu, a to především v možnosti předjíždění vlaků. Zčtyřkolejnění úseku mezi ŽST Přelouč a zastávkou Valy u Přelouče uvádí autor jako řešení aktuálního stavu a rovněž provádí analýzu a výpočet propustnosti po aplikování změn na trati.

V práci je využit simulační nástroj OT pro:

- simulace železničního provozu,
- analytický výpočet jízdních dob,
- výpočet provozních intervalů,
- funkčnost jednotlivých návrhů.

Všechny tyto části jsou plně v kompetenci nástroje OT, proto zde nevzniká téměř žádná možnost realizace jakékoli nadstavby. Simulační nástroj pokrývá plně požadavky autora práce. V závěru práce jsou uvedeny dosažené cíle, a především schopnost nástroje OT otestovat

funkčnost navržených změn. Tím autor dokazuje schopnost nástroje OT simulovat trať a zjistit její propustnost.

3.2.4 Volba vhodné posunovací lokomotivy

Při volbě vhodného hnacího vozidla nebo pouze při jeho modernizaci se mnohdy zvolí hnací vozidlo s podobnými či se zcela identickými parametry. Pokud se jedná o hnací vozidlo pro konkrétní vlečku, pak tím spíše můžeme naše nároky upřesnit dle konkrétních požadavků. Nedochází tak k předimenzování. Původní volba posunovací lokomotivy nemusela být zcela vhodná. Na tuto problematiku je zaměřena diplomová práce [3] nesoucí název *Modelování provozních parametrů posunovacích lokomotiv* napsaná v rámci studia na Dopravní fakultě Jana Pernera na Univerzitě v Pardubicích. Autorem je toho času bakalář Ondřej Fojtů. Autor zmíněné práce zdůrazňuje mnohdy nevhodnou volbu nových hnacích vozidel v porovnání s původními.

Setkáváme se zde se specifickým požadavkem na simulační nástroj OT. Nejedná se o komplexní simulaci s požadavky na propustnost či zefektivnění provozu, ale jde o zjištění specifických nároků podle parametrů vlečky a také podle potřeb vyplývajících z tratě.

Autor se zabývá řadou kritérií, které je zapotřebí sledovat. Požadavky na posunovací lokomotivu jsou specifické. Jedná se především o působení adheze. Ovšem to není jediným kritériem, které autor uvádí. Dalšími jsou maximální výkon lokomotivy nebo množství zátěže. Velká většina těchto kritérií směřuje k provozovateli, který by měl tyto vstupní hodnoty adekvátně vyhodnotit.

Práce samotná je důkazem, že za pomoci simulačního nástroje OT s podporou Microsoft Excel lze zvolit vhodné hnací vozidlo a následně provést jeho otestování v provozu v rámci simulace. Autor v práci rovněž provádí modelování a následně simulování reálného běžného provozu. Shodné výsledky běžného provozu a simulace samy o sobě dokazují korektní aplikování programu OT. Proto lze výsledky ze zmíněného simulačního nástroje považovat za reálné. Samozřejmě vždy je zapotřebí brát na zřetel odchylku.

3.3 OpenPowerNet API nadstavba na OpenTrack

Jedná se o aplikaci třetích stran, která je uzpůsobená pro komunikaci se simulačním nástrojem OT. OpenPowerNet je program pro simulaci výkonu trakčního napájení a simulaci výkonu vlaků. Program byl vyvinut společností *Institut für Bahntechnik GmbH, BO Dresden*. Jejím cílem je upravit jízdu vlaku tak, aby byly náklady sníženy na minimum při zachování dojezdové doby. Tato AN se zaměřuje především na spotřebu energie a provoz vlaků s co nejnižší možnou spotřebou. Hlavními funkcemi OpenPowerNet jsou:

- analýza zatížení pro 1AC, 2AC a DC sítě,
- výpočet tažné síly pro OpenTrack,
- výpočet brzdné síly a obnovené energie pro síťovou simulaci a analýzu,
- pokročilý model vlaku s omezením tahu a brzdného proudu, účiníku proti napětí, obnovení brzdění, brzdění na principu vířivých proudů,
- automatický výpočet magnetické vazby mezi vodiči,
- modelování napájecích zdrojů jako je usměrňovač / invertor, transformátor, konvertor (SFC),
- modelování skladování energie stanice pro stabilizaci síťového napětí a úsporu energie,
- modelování skladování energie motoru poskytnuté z OCS a umožnění jízdy na volných trasách OCS,
- modelování pomocného transformátoru,
- modelování zařízení pro omezení napětí,
- výpočet zkratových proudů podél linie,
- výpočet napětí vedení pro motory s konstantním proudem pro analýzu elektrické sítě,
- energetická bilance,
- zátěžová analýza zařízení,
- dosažení výpočtu napětí,
- výpočet a vizualizace elektromagnetického pole,
- vizualizace výsledků pomocí automatické analýzy analytického nástroje,

- ❖ generování tabulek a diagramů automatizovanou analýzou v libovolném jazyce podle konfigurace uživatele pomocí znaků v UTF-8.

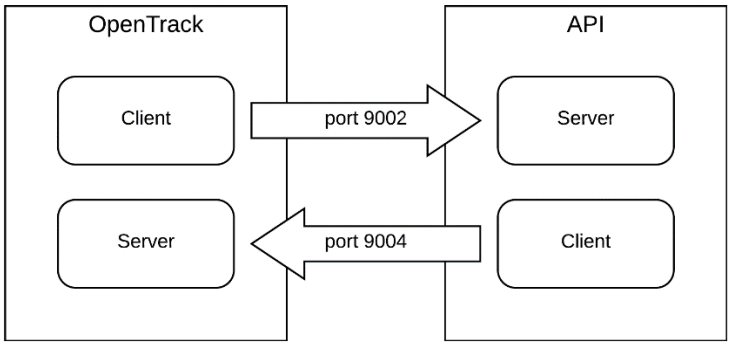
Schopnost tohoto systému je získávání informací od simulačního nástroje OT a následné aplikování poznatků vede k úspoře energie, a tedy i nákladů na celkový provoz. Aplikace jako taková je důkazem využitelnosti AN, která nemusí být chápána pouze jako menší doplněk k původnímu systému, nýbrž jako software, který svou velikostí a složitostí překoná svého předchůdce. Stále je ovšem nutné mít na paměti, že OpenPowerNet je především nadstavbou, kterou nelze využívat bez programu OT [12, 15].

4 OPENTRACK API

Ve čtvrté kapitole je popsána především AN. Je zde kladen důraz na kapacity simulačního nástroje OT pro realizaci komunikace s aplikací třetích stran. Současně je v této kapitole konkrétně popsána realizace serveru a klienta v jazyce Java. V závěru jsou uvedeny a popsány konkrétní příklady aplikování OT API.

4.1 Komunikace

Komunikace je výchozím prvkem pro realizaci AN. Jedná se o obousměrné spojení, kdy aplikace musí umět reagovat na informace poskytnuté od simulačního nástroje a obráceně. Jak již bylo zmíněno, principem komunikace je klient, zasílající zprávy na server. Zpáteční zprávy však nejsou odesílány jako odpověď serveru na stejném portu, nicméně jsou odesílány jako jednosměrná zpráva ze strany klienta na jiném portu. Níže umístěná vizualizace zobrazuje formu komunikace [13].



Obrázek 1: Vizualizace komunikace. Zdroj: [13]

4.1.1 Mód OTD

Jedná se o speciální režim programu OT sloužící pro komunikaci a realizaci samotné nadstavby. Tento mód je běžnému uživateli nedostupný a pro jeho zpřístupnění je zapotřebí rozšíření licence. Zpřístupňuje správu nastavení komunikace pro použití rozhraní API. OT se rozšíří pouze o dvě volby v hlavním panelu menu, viz Obrázek 2: Nastavení OTD. Zdroj: [13] První přidanou položkou je OTD Settings sloužící pro nastavení komunikace. Již bylo zmíněno, že OT dovoluje obousměrnou komunikaci, pro jejíž realizaci využívá dvojici portů. Ze strany simulačního nástroje se jedná o realizaci komunikace klient-server, kdy na OT na prvním portu

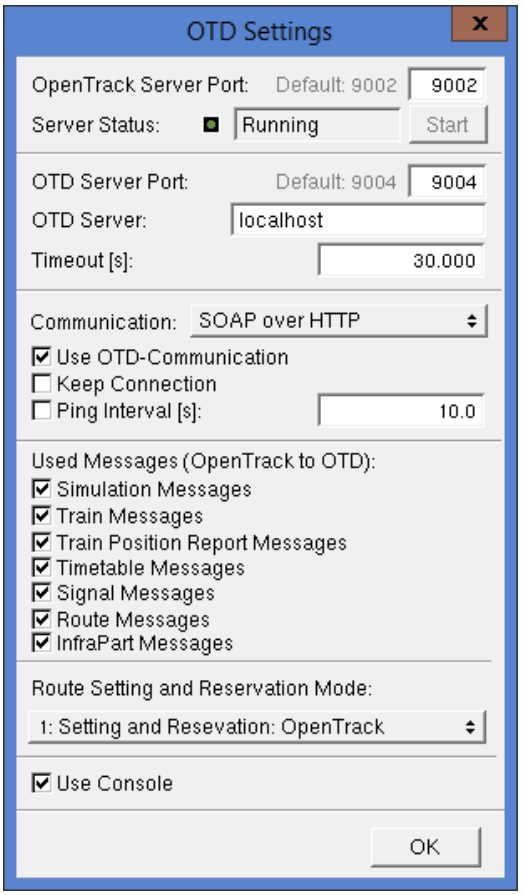
ve výchozím stavu 9002 realizuje Windows server sloužící pro naslouchání, a tedy přijímání zpráv. Druhým portem je v počátečním nastavení 9004 sloužící pro odesílání zpráv API. Nastavení rovněž umožňuje odesílat zprávy na základě libovolné IP adresy. Nedochází tak k omezení na localhost [9, 12].

Dalším prvkem nastavení je volba formy jednotlivých zpráv, která rozhoduje o tom, zda budou zprávy zasílány ve formě SOAP, tedy Simple Object Access Protocol, či jeho modifikací SOAP/DIME (Direct Internet Message Encapsulation) od společnosti Microsoft. Podrobný popis zpráv SOAP je umístěn níže. V nastavení je dále možnost zapnout odesílání těchto zpráv aplikací třetích stran. Nyní je vhodné zmínit, že bez realizace serveru v rámci API, který by naslouchal na zmíněném portu, bude již při spuštění program upozorňovat na absenci spojení, a znemožňovat tak další práci v simulačním nástroji. Volitelná je i možnost udržovat spojení a nepřerušovat jej po každé zprávě. Toto nastavení se doporučuje uplatňovat v případě vyššího množství transakcí [13].

Součástí OTD Settings je rovněž možnost volby jednotlivých zpráv. Lze tak zcela zrušit zasílání zpráv určitého druhu, či povolit zprávy podle konkrétní volby. Zprávy jsou rozděleny do několika kategorií, a to na:

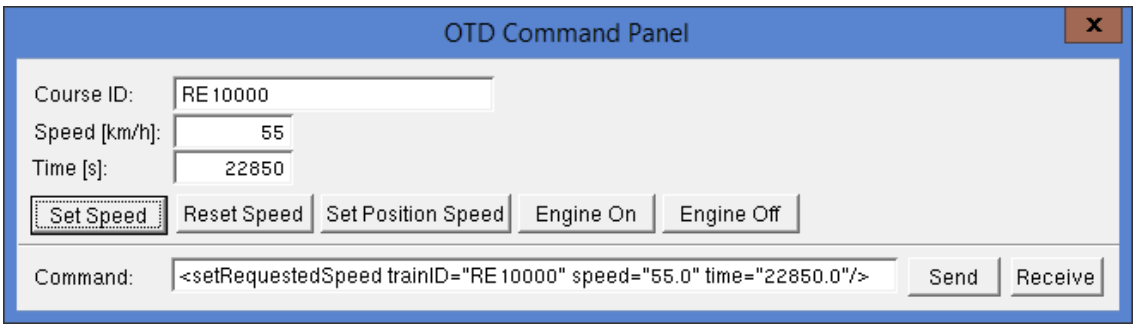
- zprávy o aktuálním stavu simulace,
- zprávy o vlacích,
- zprávy o pozici vlaků,
- zprávy o trasách vlaků,
- zprávy o stavu zabezpečovacího zařízení,
- zprávy o obsazenosti trati.

Předposlední položkou nastavení je volba rezervace trati, která vymezuje, zda bude rezervaci trati pro vlak provádět OT, či zda bude API zasílat zprávy a rezervovat jednotlivé úseky. Poslední položkou je možnost zobrazit okno s podrobným výpisem odeslaných, popřípadě přijatých zpráv. Toto okno lze uplatnit především při ladění aplikace. Celé nastavení OTD je zobrazeno níže na obrázku [13, 18].



Obrázek 2: Nastavení OTD. Zdroj: [13]

Druhým rozšiřujícím prvkem programu OT je OTD Command Panel sloužící pro zasílání zpráv. Pomocí tohoto panelu lze simulovat zasílání zprávy simulačnímu nástroji OT, ale rovněž pomocí něj simulovat odeslanou zprávu aplikační nadstavbě. Příkazový panel dokáže realizovat SOAP zprávu, kdy v dolní části panelu se zobrazuje samotné tělo zprávy. Mimo nastavování rychlosti a ovládání energie lze editací těla zprávy zaslat jakoukoli zprávu simulačního nástroje OT [13, 18].



Obrázek 3: Panel pro zasílání zpráv. Zdroj: [13]

4.1.2 Zprávy

Jak již bylo zmíněno, zprávy jsou realizovány ve formátu protokolu SOAP. Tento protokol je privilegovaně určen pro komunikaci po síti, tedy především za pomoci HTTP. SOAP je založený na značkovacím jazyku XML. Zprávy jsou navrženy tak, aby byly nezávislé na operačním systému či programovacím jazyku, ve kterém budou vytvořeny, popřípadě přijaty.

Zprávy SOAP jsou uzavřeny v obálce, kterou tvoří párový tag, první a poslední ve zprávě. Dále se zpráva dělí na dva základní prvky. Prvním je hlavička, kterou ovšem zprávy v programu OT postrádají. Druhým prvkem je tělo zprávy znovu ohraničeno párovým tagem. V rámci zvoleného simulačního nástroje je ve zprávách pouze jediný nepárový tag, což velice usnadňuje čtení zpráv. Takovýto tag pak obsahuje pouze atributy odpovídající druhu zprávy. Počet a druh těchto atributů se může měnit podle typu zprávy, a některé atributy jsou dokonce zcela volitelné. Tím pádem nelze podle druhu zprávy určit pořadí, druh či počet atributů. Jméno tohoto nepárového tagu v těle obálky je hlavním prvkem a samotnou zprávou, kdy atributy jsou většinou doplňující informace. Čas ve zprávách je udáván vždy v sekundách a rychlost v kilometrech za hodinu. Níže je přiložena ukázka zprávy SOAP, která nastavuje rychlost vlaku s označením IC4001 na 80 km/h v čase simulace 32400 sekund [13, 18].

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <setRequestedSpeed trainID="IC4001" speed=”80”
        time="32400" />
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Ukázka kódu 1: Zpráva SOAP

4.1.3 Vlastní server

Obrázek 2 znázorňuje stěžejní prvky komunikace a je z něj patrné nezbytné vytvoření serveru a klienta na straně API. Za výchozí byl zvolen programovací jazyk Java, a právě v něm bude popsána a provedena kompletní implementace [2, 13].

Zásadním balíkem pro realizaci síťových aplikací je *java.net* a jeho stěžejní třídou je *ServerSocket*, která implementuje serverové sokety. Server čeká na požadavky na portu a po příchodu zprávy provádí konkrétní operaci založenou na tomto požadavku. Následně vrací výsledek žadateli. Takovouto odpověď OT nepoužívá. Pro zasílání odpovědi či výsledků libovolných úkonů používá komunikaci na jiném portu. Odpověď je předána pomocí API klientovi, který zašle zprávu na server OT a ten zprávu zpracuje. Spojení tak vždy běží pouze jedním směrem [5, 13].

Konkrétní implementace je velmi prostá. Třídě *ServerSocket* je v konstruktoru nadefinováno číslo portu, na kterém bude naslouchat. Voláním metody *accept* lze odstartovat přijímání zpráv na portu. Snadno lze pak za pomoci instance třídy *Socket*, již právě vrací zmíněná metoda *accept*, získat *InputStream*, který je následně zpracován pomocí *InputStreamReader* a následně i pomocí *BufferedReader*. Tím je zpřístupněn obsah zpráv, který je možné v reálném čase vypisovat na výstup. Zprávy SOAP jsou na první pohled nečitelné, proto je vhodné je dále zpracovat a vypisovat pouze obsah těla zprávy [5, 18].

4.1.4 Vlastní klient

Realizace klienta obnáší především vytvoření SOAP zprávy. K tomu slouží balík *java.xml.soap*, který poskytuje nástroje pro vytváření zpráv. Lze tak snadno zformovat obálku, její prvky či jednotlivé atributy v prvku. Mimo to tato třída poskytuje nástroje i pro realizaci klienta na zadané IP adrese a portu. Současně je možno vytvořené zprávy snadno předat klientovi, jenž následně tyto zprávy odešle na server v našem případě serveru programu OT [5].

4.2 Klíčové příklady uplatnění

4.2.1 Získávání informací o jízdách vlaků

Schopnost získávat informace o poloze vlaků je výchozím prvkem dalších příkladů užití, proto bude v úvodu detailně rozebrána. Jedná se o vytyčení trasy, kdy na zaslaný požadavek ze strany

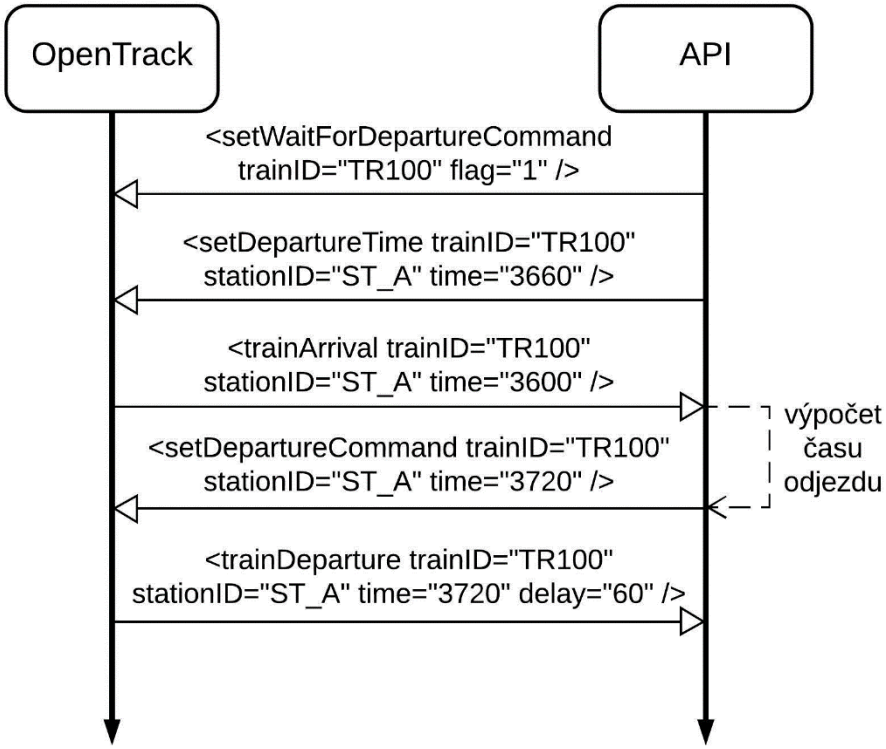
API program OT začne v nastavitelném intervalu zasílat zprávy o stavu a pozici vlaku. Tento způsob lze aplikovat i alternativě. Simulační nástroj nebude informovat periodicky, nýbrž pouze při dosažení klíčového bodu. Klíčovým bodem může být libovolný úsek trati či stanice. Trasovací zprávy jsou odeslané z programu OT aplikační nadstavbě a nesou informace o skutečné poloze vlaku včetně dalších specifikací, jako je skutečná rychlost v km/h, zrychlení v m/s^2 a zpoždění v s. Skutečná poloha čela vlaku je definována aktuálním ID trasy společně se vzdáleností uvedenou v m od počátku trasy. Součástí zprávy je rovněž čas simulace uvedený také v s, který je obsažen téměř v každé zprávě zaslané simulačním nástrojem OT aplikační nadstavbě. Za využití těchto dvou způsobů získávání informací o jízdách vlaků lze pak aplikovat složitější mechanismy uvedené níže [13].

4.2.2 Opuštění stanice

Za pomoci zpráv od aplikace třetích stran lze ovládat proces zastavení vlaku ve stanici. S využitím příkazů *Departure Command,* tedy zpráv pro opuštění stanice, lze skrze API kontrolovat a řídit opouštění stanic. Jinými slovy lze spustit možnost, aby žádný vlak neopouštěl stanici bez přijetí příkazu k jejímu opuštění. Tato možnost byla již popsána v rámci nastavení OTD. S využitím uživatelského nastavení OTD se jedná pouze o globální nastavení pro veškerou simulovanou vlakovou dopravu, a to není ve většině případů vždy žádoucí. Z tohoto důvodu OT umožňuje uplatit tuto možnost pouze na konkrétní vlak. K tomu slouží příkaz *setWaitForDepartureCommand* obsahující parametry: pro identifikaci vlaku a příznak, který určuje, zda tento mód má být spuštěn, či vypnut. Samotný odjezd se ovšem řídí několika dalšími nutnými prvky: rezervace trasy, která musí být uvolněna a rezervována pro zvolený vlak, minimální doba pobytu ve stanici, je-li definována, a čas odjezdu, je-li rovněž definován. Pakliže jsou všechny podmínky splněny, vlak se dává do pohybu. Ve výchozím stavu je tato možnost řízení opuštění stanic pro všechny vlaky vypnutá. Vlakové soupravy se nebudou omezovat v odjezdu ze stanice kromě zmíněných tradičních nutností [4, 13].

Na následujícím obrázku je naznačena komunikace při použití manuálního ovládání opuštění stanice. Prvním příkazem je spuštěn samotný způsob ovládání, který byl již popsán. Následuje zpráva nastavující čas běžného odjezdu vlaku. Vlak by měl tedy opustit stanici A v čase 1:01. Tyto zprávy může OT obdržet již před samotným startem simulace. Po spuštění simulace a po uplynutí nutné doby se simulace dostane do klíčového bodu, kdy dorazí vlak do stanice A. O této skutečnosti informuje OT zprávou obsahující čas příjezdu a identifikační údaje o stanici a vlaku. Jelikož vlak přijel v čase 1:00 a odjezd, jak stanovila API zprávou, je nastaven na 1:01,

měl by za normálního stavu vlak počkat 60 sekund a poté pokračovat v jízdě. Spuštěním módu čekání na povolení odjezdu však vlak stanici bez svolení neopustí. Poté, co API vykalkuluje dle interních náležitostí vhodný čas odjezdu, odešle zprávu s touto informací simulátoru. Povolení od API tedy vlak obdržel na čas 1:02. Při odjezdu vlak odešle informace o stanici, sobě, času odjezdu a době zpoždění, které v uvedeném příkladu aktuálně činí 60 sekund [13].

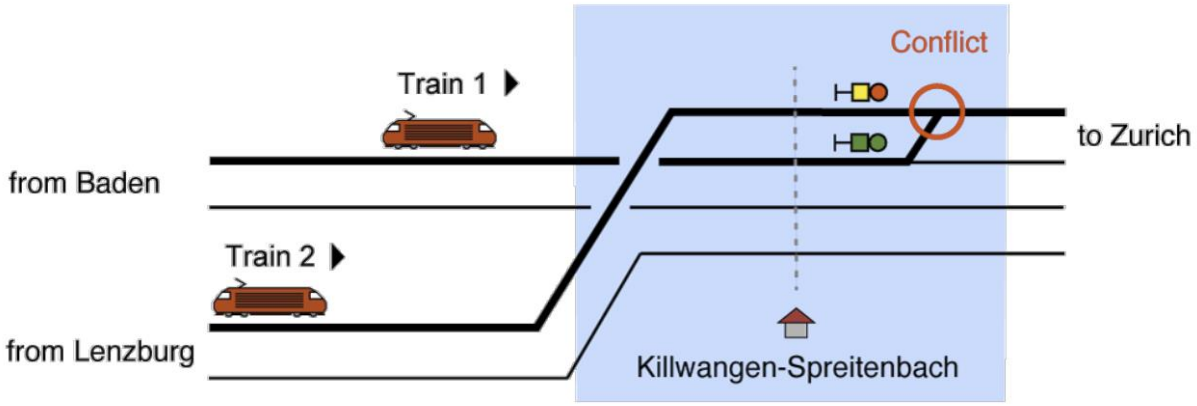


Obrázek 4: Ukázka komunikace na příkladu opuštění stanice

4.2.3 Lokální omezení rychlosti

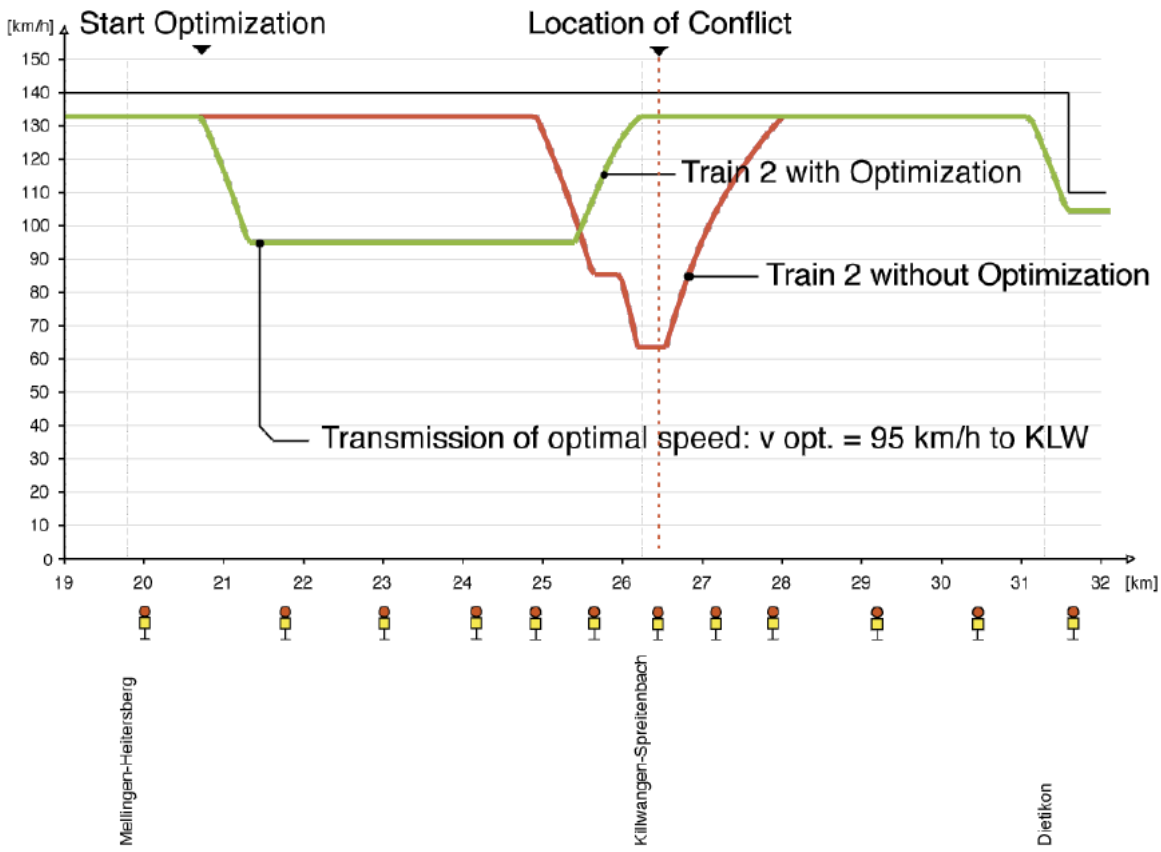
Za pomoci zprávy *setPositionSpeed* lze ve zvoleném úseku linky nastavit omezení rychlosti. Toto nastavení se dá aplikovat na jeden vlak nebo pro všechny vlaky na daném úseku. Omezení je možno uplatnit i v časovém rozmezí, nebo jej nastavit pro celkový časový rámec. Hlavním účelem je výpočet optimální rychlosti pro všechny vlaky. Nedochází tak ke konfliktům, které vedou ke zbytečnému brzdění, či dokonce k samotnému zastavení celé vlakové soupravy. Konflikty rozumíme omezení z vlivu obsazenosti trati a nutné zastavení u návěstidla. Aby vlak nemusel před návěstidlem výrazně brzdit, či zcela zastavit, může již ve větší vzdálenosti snížit svoji rychlost, a tak vlak dojede k signalizaci později, a nebude tedy nucena náhle brzdit nebo

zbytečně čekat. Díky tomuto kontrolovanému zpoždění vlak projíždí plynule kolem návěstidla. Níže je vyobrazený možný konflikt dvou vlaků v křížícím se bodu trati [13].



Obrázek 5: Konflikt na trati. Zdroj: [13]

V následující rychlostním grafu je červenou čarou vyobrazena jízda vlaku číslo 2 bez aplikování omezení rychlosti. Z grafu je patrné, že vlak byl před návěstidlem nucen snížit svou aktuální rychlost až na 65 km/h. Následně po uvolnění trati a uvedení návěstidla do stavu „volno“ vlak znovu zrychloval na svou stálou rychlost 130 km/h. Tímto brzděním a zrychlováním dochází ke zbytečnému plýtvání energie, ať už jde o dieselovou či elektrickou lokomotivu. Zelená linka naznačuje průběh identické situace s aplikováním včasného zpomalení a plynulejšího projetí klíčového bodu. Dřívější zpomalení na 95 km/h vede k pozdějšímu příjezdu na konfliktní bod na trati, a díky tomu se vlak 2 zcela vyhne akutnímu nucenému brzdnému manévru před návěstidlem. Jak je vidět z grafu, vlak následně rovněž nabírá rychlost na 130 km/h, ovšem nedošlo zde ke zrychlení o 65 km/h, ale pouze o 35 km/h. Aplikováním tohoto mechanismu dochází k úspoře energie [4, 13].



Obrázek 6: Optimalizace jízdy vlaku. Zdroj: [13]

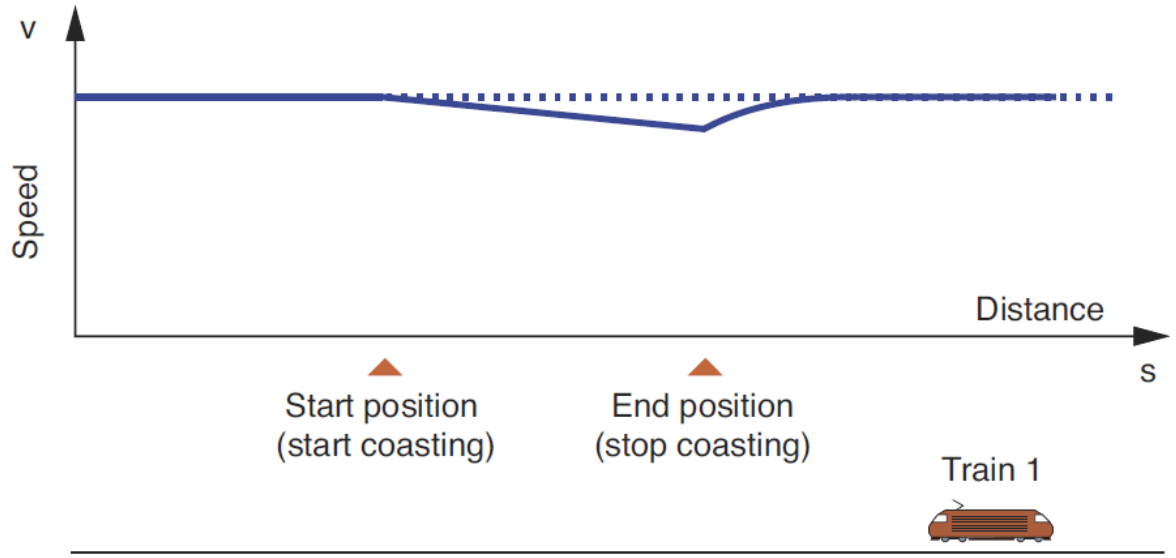
Zpráva, kterou lze regulovat rychlost, nese několik volitelných parametrů. Jsou jimi ID vlaku, čas od kdy do kdy má být omezení uplatňováno a binární hodnota příznaku udávající, zda bude omezení v provozu. Dalšími tentokrát povinnými parametry jsou rychlost, identifikace počátečního a koncového úseku trati s doplňujícím parametrem, který udává délku od počátku úseku. Následuje ukázka konkrétní zprávy zaslané API programu OT nastavující zpomalení, které bylo popsáno na výše uvedeném rychlostním grafu [13].

```
<setPositionSpeed trainID="Train 2" speed="95.0" startRouteID="Topo_LT-
12-R:MELH-LANG" startRouteOffset="1314.0" endRouteID="Topo_LT-58-R:LANG-
KLW" endRouteOffset="110.0" />
```

Ukázka kódu 2: Ukázka příkazu omezení rychlosti na trati

4.2.4 Jízda výběhem

Jedná se o takovou jízdu vlaku, kdy není zapojena jeho hnací síla a rovněž nedochází ani k účelnému brzdění. Vlak se pohybuje na počátku určitou rychlostí, kterou v průběhu setrvační jízdy postupně ztrácí, to má za následek jeho zpomalování. Účelem této varianty je úsporné zpomalení bez využívání brzdného systému, který je tak zbytečně namáhán. Na následujícím grafu je vyobrazeno použití jízdy výběhem a její vliv na rychlost. V bodě konce jízdy výběhem vlak znovu zrychluje na svou běžnou rychlost, která je naznačená tečkovanou čarou [13].



Obrázek 7: Jízda vlaku výběhem. Zdroj: [13]

5 REALIZACE VLASTNÍ APLIKACE S VYUŽITÍM OPEN-TRACK API

Cílem praktické části práce je realizovat AN a využít ji k řešení konkrétní problematiky. Za cíl byla stanovena problematika předjíždění, přesněji její realizace, neboť simulační nástroj OT tento mechanismus postrádá. Díky možnému rozšíření za pomoci AN je vybaven nástroji pro její realizaci. Jedná se o komplexní problém, který pro svou úplnou realizaci přesahuje rámec této práce. Autor si bere za cíl vytvořit obecněji uplatnitelný algoritmus, za jehož pomoci s aplikováním nad konkrétním úsekem trati bude aplikace nad API schopna sama detekovat a následně realizovat manévr předjíždění zcela automaticky.

5.1 Seznámení s problematikou předjíždění v kolejové dopravě

Předjíždění v železniční dopravě je manévr, při němž je první, většinou pomalejší, vlak zastaven na vedlejší koleji v dopravně s kolejovým rozvětvením a po hlavní koleji projíždí rychlejší vlak. Po předjetí pokračuje první zastavený vlak v cestě.

Dopravnou se rozumí úsek železniční tratě sloužící pro určování pořadí vlaků na trati a posunu mezi dopravnami. Rozdělují se podle kolejového rozvětvení, zda jím disponují, či jej postrádají. Za dopravny se považují:

- stanice,
- výhybny,
- hradla,
- hlásky,
- oddílová návěstidla automatického bloku,
- oddílová návěstidla automatického hradla,
- odbočky k tomu určené.

Hradla a především pak hlásky závislé na své obsluze hradlařem a hláskařem se dnes již téměř nevyužívají, z toho důvodu tyto dvě položky nelze brát jako uplatnitelné pro předjíždění vlaků. Dále ani u oddílových návěstidel nepředpokládám možné předjíždění vlaků [10, 17].

5.2 Návrh aplikace

Pro realizaci AN byl zvolen jazyk Java. Především díky své nezávislosti na architektuře počítače s nutností vlastnit pouze interpreta JVM. Grafické rozhraní bude realizováno v JavaFX. Realizace grafického rozhraní není nikterak rozsáhlá a určitě ji lze realizovat s využitím knihovny Swing. Autor se rozhodl pro volbu novější softwarové platformy především kvůli ukončenému dalšímu vývoji knihovny Swing od společnosti Sun Microsystems.

Prvotním požadavkem a cílem práce bylo analyzovat schopnost komunikace simulátoru s API. Z tohoto důvodu byla vyžadována realizace aplikace s grafickým rozhraním, která by sloužila k manuálnímu zadávání a odesílání příkazů simulátoru. Druhou částí je následně automatické uplatňování složitějších mechanismů pro řízení železniční dopravy. V tomto případě již nebude grafické rozhraní tolik rozsáhlé, neboť je prioritou automatický, pro uživatele nezajímavý, proces předjíždění. Cílem bude pouze aplikovaný výsledek na průběhu simulace.

5.2.1 Obecné uplatnění

Aplikace umožňuje zasílat libovolnou zprávu simulátoru OT. Rovněž obsahuje sadu všech příkazů, které simulátor pro svoje řízení zpřístupňuje. Pro větší přehlednost jsou tyto příkazy rozloženy do hierarchické struktury. Součástí grafického návrhu je i panel obsahující popisek k danému příkazu. Tento komentář by měl obsahovat především předpis samotného příkazu, komentář jednotlivých parametrů, popřípadě rozsah u každého parametru. Takovýto rádce se zobrazí vždy po zvolení daného příkazu. Triviální verze příkazu, bez volitelných atributů, bude při zvolení rovněž umístěna do konzole, která bude taktéž součástí aplikace. Z konzole je zpráva následně odeslána simulačnímu nástroji. Součástí budou i ovládací prvky, jakými jsou spuštění, zastavení a ukončení simulace. Nastavitelná bude i možnost ovládání rychlosti simulace z AN. Rozložení grafických komponent bude popsáno v kapitole 5.5 Grafické rozhraní.

5.2.2 Aplikace konkrétní problematiky

Tato část aplikace bude zaměřena na realizaci předjíždění vlaků. Jedná se o velice komplexní problematiku, a proto její širší uplatnění je jenom stěží realizovatelné. Z tohoto důvodu bude tato část řešena na konkrétním úseku tratě v bezprostřední blízkosti železniční stanice Pardubice hlavní nádraží. Požadavky na grafické prvky tedy jsou pouze minimální, po uživateli je požadováno zadat pouze minimum vstupních dat. Nutnými prvky bude volba a realizace

vhodných datových struktur, rovněž uplatnění vláken v rámci aplikace, které si bere autor za cíl realizovat se záměrem lepší odezvy programu. Je nutné, aby API takzvaně „stíhala“ přijímat příchozí zprávy, zpracovávat je, a na základě toho adekvátně jednala. Důležitým faktorem se stává schopnost pracovat pouze se simulačním časem, a to z toho důvodu, aby výsledky simulace byly pro uživatele přínosné.

5.3 Klíčové body API pro realizaci automatického předjíždění

Tato kapitola slouží k lepšímu pochopení programu a poznání způsobu jeho vyhodnocování. Využité technologie budou popsány v následující kapitole, kde bude rozebrána realizace jednotlivých vláken a uplatněných datových struktur. Cílem kapitoly je seznámit s řešením tohoto problému. Celý proces zajišťující předjíždění lze rozdělit na tři části. První z nich je detekce sloužící k přijímání nejenom pozičních zpráv od vlaků. Druhým prvkem je vyhodnocení na základě získaných dat, kdy a se kterými vlaky má být předjíždění uskutečněno. Posledním je provedení samotného předjetí, které se stává klíčovým bodem této práce.

5.3.1 Detekce

Jak již bylo zmíněno, náplní detekce je zpracovávat vstupní data a adekvátně je připravit na vyhodnocení. V této části jsou separovány veškeré zprávy, které jsou pro předjíždění potřebné. Nejedná se pouze o poziční zprávy vlaku, ale současně o zprávy oznamující příjezd vlaku do stanice, odjezd vlaku ze stanice nebo projetí vlaku stanicí a řadu dalších zpráv. Na detekci je kladen důraz vysoké odezvy, neboť může sloužit pro realizaci více mechanismů a všem by měla předkládat adekvátní data s co nejmenším zpožděním. Proto je tato část separována ve vlastním vláknu programu tak, aby nedocházelo k jakémukoli zpoždění. Veškeré zprávy jsou nejprve separovány o nepotřebné údaje. Výsledkem je pouze tělo zprávy, které je dále zpracováváno a přetvořeno do objektové podoby. V této formě je uloženo na zásobník a připraveno na další zpracování. Ne všechny zprávy jsou pro nadstavbovou aplikaci nad API zajímavé, a z tohoto důvodu nedochází ani k analyzování všech příchozích zpráv. Tyto nežádoucí zprávy jsou pouze zjednodušeně zaznamenány do triviální objektové podoby.

Nyní blíže ke konkrétní problematice předjíždění. Mějme tedy zpracované zprávy od simulátoru o pozicích jednotlivých vlaků na trati. Tyto zprávy je zapotřebí ukládat a následně provést vyhodnocení. Zprávy přichází od každého vlaku v sekundovém intervalu a jakákoli zastaralá zpráva, myšleno taková zpráva od vlaku, o které již máme čerstvější informaci

stejného druhu, je nezajímavá, a není nutné ji nadále nést. Z toho plyne, že pro vložení je vyžadováno vyhodnocování vstupních dat, zda již touto zprávu o vlaku nedisponujeme, a v takovém případě novou zprávu nahrazujeme původní. Množství příchozích zpráv zvláště při zrychlení simulačního času muže být enormní, z toho důvodu je zapotřebí minimalizovat asymptotickou složitost algoritmu a provést vhodnou volbu datové struktury. Tato problematika je blíže popsána v kapitole 5.4.2 Datové struktury. Uložením zprávy v objektové podobě je detekce završena.

5.3.2 Vyhodnocení

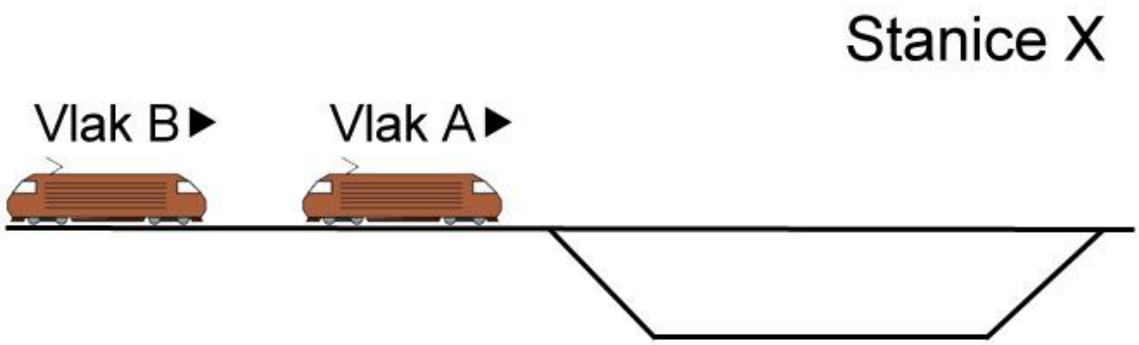
Cílem vyhodnocování je vybrat takovou dvojici vlaků, se kterou má být předjetí provedeno. Zprávy o poloze vlaků jsou v podobě objektů uloženy v datové struktuře hashovací tabulky, a tím je zaručena jedinečnost v rámci jednotlivého vlaku. Aby nebylo prováděno vyhodnocování se všemi vlaky vzájemně, postačuje vyhodnocovat pouze ty, které jedou přímo za sebou. Tím se snižuje počet volání vyhodnocovací funkce, ovšem je nutné mít vlaky seřazené v identickém pořadí, v jakém se pohybují po trati. Na to lze uplatnit libovolný řadící algoritmus, navíc budou zaznamenávány pouze vlaky právě se pohybující po simulovaném úseku tratě, takže jich neočekáváme velké množství. Nad malým množstvím dat se výrazně neprojevuje efektivita důmyslnějších řadících algoritmů, z toho důvodu byl pro realizaci zvolen Insert sort. I přes jeho asymptotickou složitost $O(n^2)$ se může například oproti Bouble sortu při práci se seřazenými daty vyznačovat složitostí blížící se $O(n)$. Dalším požadavkem je udržovat data aktuální. Pokud vlak opustí hranice simulace nebo se objeví nový vlak na simulovaném úseku, je zapotřebí množinu zpráv adekvátně editovat. K tomu účelu je využito detekování zpráv od OT o odjezdech a příjezdech do simulované části tratě [14].

Nad seřazenými daty lze použít vyhodnocovací algoritmus pro realizaci předjetí. Požadavky na něj mohou být jednoduché, ale může se jednat o poměrně složitý soubor podmínek. V základním návrhu je ověření maximální rychlosti obou vlaků ve spojení s přihlédnutím na druh vlaku, který je předjížděn, neboť osobní vlaky zastavují ve většině stanic a z toho důvodu je u nich předjetí prioritou, nehledě na jejich maximální rychlost. Tato část je libovolně editovatelná a je spíše otázkou na odborníka z oboru řízení železniční dopravy. Vyhodnocení se provádí vždy nad dvojicí po sobě následující, a v případě splnění podmínek je tato dvojice vlaků uložena do seznamu předjížděných.

Poslední otázkou, kterou zbývá zodpovědět, je, kdy bude toto vyhodnocení nad seznamem provedeno. Pokud bychom se spokojili s identickým intervalem, v jakém získáváme zprávy od vlaků, tedy každou sekundu reálného času, pak v případě zrychlené simulace 60x by tato analýza probíhala pouze jednou za hodinu simulačního času. Simulace je téměř vždy používána ve zrychleném režimu. Z tohoto důvodu je zapotřebí zacházet pouze se simulačním časem. Lze provádět vyhodnocení v periodě jedné sekundy simulačního času, to ovšem nezajistí, že již přišly zprávy od všech vlaků pohybujících se po simulovaném úseku trati. A aby nebylo vyhodnocení prováděno po větším časovém úseku a byly vždy všechny přijaté poziční zprávy adekvátně využity, je stanoveno počítadlo vlaků v simulaci, které využívá zpráv *trainCreated* a *trainDeleted* pro provádění inkrementace a dekrementace počítadla. Dříve zmíněný seznam zpráv vždy zaručující jedinečnost v rámci vlaku pak při dosažení velikosti počítadla provádí funkci vyhodnocení. Tento seznam je následně smazán, tím je zaručeno, že vyhodnocení se bude průměrně provádět každou sekundu simulačního času a ne po příchodu nové zprávy, kterých může přijít několik v rámci jedné sekundy.

5.3.3 Provedení předjetí

Předjetí lze separovat na množinu úkonů, které je nutné provést. Pro lepší představu a pochopení problematiky mějme vlak A, který má nižší maximální rychlost než vlak B jedoucí na stejné trase jako vlak A nebo vlak A může být osobním vlakem. Tyto vlaky se vzájemně přiblížily na detekční vzdálenost, kterou lze editovat podle potřeby. Jediným omezením tohoto kritéria je vzdálenost zkoumaného úseku trati. Nejbližší následující dopravnou na trati je stanice X.

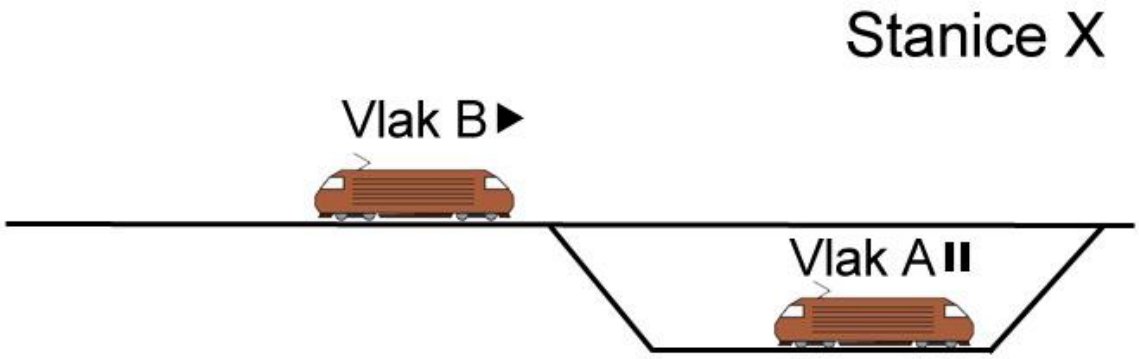


Obrázek 8: Detekce předjetí

Za využití detekce a vyhodnocení tedy došlo k rozpoznání potřeby předjetí vlaků a nyní je zapotřebí provést úkony:

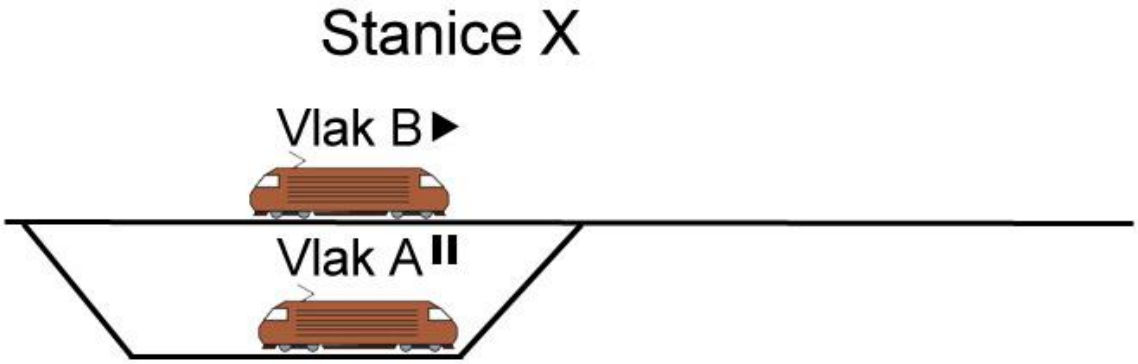
- zastavit vlak A v nejbližší následující dopravně,
- provést zaznamenání o příjezdu vlaku A do stanice X,
- zaznamenat průjezd vlaku B stanicí X,
- povolit vlaku A opuštění stanice.

Pro zastavení vlaku A je využito zprávy *setWaitForDepartureCommand,* kdy vlak zastaví a neopustí stanici X, dokud neobdrží příslušnou zprávu s povolením. Po příjezdu vlaku A do dopravny stanice X simulátor odešle zprávu *trainArrival*, jejíž součástí jsou identifikační údaje o vlaku a stanici současně s časem příjezdu a případným zpožděním spoje. Je zapotřebí tato data adekvátně zaznamenat, ve které dopravně vlak A čeká na předjetí od vlaku B. Tato informace je připojena k již neseným datům o předjetí.



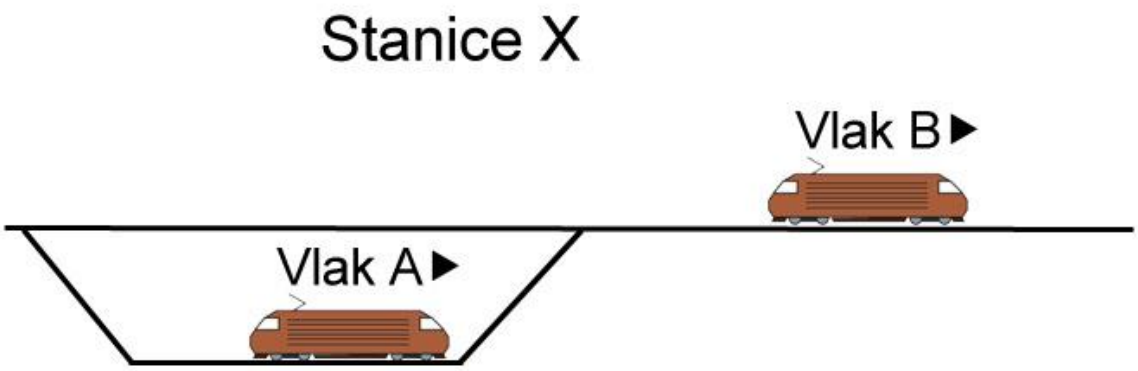
Obrázek 9: Vyčkávání na předjížděcí vlak

Nyní se vyčkává na vlak B, který může ve stanici zastavovat v takovémto případu zašle simulátor již zmíněnou zprávu o příjezdu, nebo vlak ve stanici nestaví a pouze projíždí, v takovém případě odešle zprávu *trainPass,* která nese identické informace jako *trainArrival*. V případě pouhého projetí vlaku B stanicí X může vlak A bez odkladu obdržet povolení k odjezdu zaslanou zprávou od API *setDepartureCommand*. To lze díky tomu, že když vlak B projíždí stanicí bez zastavení, rezervuje rovnou i úsek za dopravnou. Vlak A, který již obdržel povolení k odjezdu, musí vyčkat, než bude trať uvolněna a bude si ji následně moci rezervovat.



Obrázek 10: Předjetí

V případě, že vlak B ve stanici X staví, je zapotřebí počkat, než tento vlak stanici opustí. To zaručuje zpráva simulačního programu *trainDeparture*. Po obdržení zprávy, že vlak B již opouští stanici X, vlaku A může být povolen odjez ze stanice. Tím je proces předjíždění ukončen.



Obrázek 11: Dokončení procesu předjetí

5.4 Použité technologie

V této kapitole budou popsány stěžejní použité technologie z oboru informatiky, které byly uplatněny v této práci. Důležitým prvkem pro lepší pochopení této kapitoly je Obrázek 12: Sekvenční diagram, který zjednodušeně zobrazuje jednotlivá vlákna společně s klíčovými datový strukturami.

5.4.1 Vlákna

Během implementování práce bylo uplatněno vláknové programování s cílem získat nejvyšší možnou odezvu. Provést realizaci AN, která by dynamicky souzněla se simulátorem a korektně prováděla komunikaci a následné vyhodnocování situací potřebných pro zvládnutí zvolené problematiky. Byla využita celkem tři vlákna vždy realizována v separátní třídě.

Třídou realizující první vlákno je třída *ServerThread*, ta je potomkem třídy *Thread*, díky tomu lze překrýt metodu *run* a provést tak vlastní implementaci vlákna. Vlákno využívá několik atributů, jako jsou datová struktura fronty *IAbstrFifo*, semafor třídy *Semaphore* z balíku *java.util.concurrent* pro řízení přístupu do kritické sekce, kterou je zmíněná fronta. Posledním atributem je proměnná typu boolean sloužící pro cyklení vlákna. V klíčové metodě *run* je cyklus typu while, jehož podmínkou postupu je zmíněná proměnná. Ta je defaultně nastavena na hodnotu true a pouze v případě zavolání metody *vypnout* dojde k překlopení proměnné a tím k ukončení celé metody *run*.

V metodě *run* dochází k realizaci serveru na zvoleném portu a následnému naslouchání. Podrobný popis realizace je uveden již v kapitole 4.1.3 Vlastní server. Po získání kompletní zprávy SOAP zbývá poslední část, kterou je metoda *zpracuj*, ta má za úkol vyseparovat klíčovou část z přijaté zprávy. Přesněji se jedná o obsah těla obálky zprávy. Tuto získanou informaci vkládá do fronty a následně prování inkrementaci semaforu. Metody třídy *AbstrFifo* jsou realizovány s klíčovým slovem *synchronized*, tím je tedy zaručena jejich atomicita. Semafor slouží ke kontrole odebírání prvků z fronty. Fronta i semafor jsou předány v konstruktoru, neboť s nimi bude zacházet i další třída implementující vlákna, kterou je *FiltrMessageThread*.

Její náplní je zprávy nejprve zpracovat a následně filtrovat a na tomto základě adekvátně reagovat. Třídě *FiltrMessageThread* je v konstruktoru předána identická fronta se semaforem jako byla předána v konstruktoru třídy *ServerThread*. Uvnitř metody *run* je zde identicky použitý while cyklus s proměnnou pro ovládání ukončení vlákna. Zprávy jsou postupně odebírány z fronty, současně probíhá dekrementace semaforu voláním metody *acquire*. Semafor zabraňuje odebírání prvků z prázdné fronty tím, že vlákno je uspáno. Nad zprávami odebranými z fronty se provádí nejprve volání lexeru, ten slouží pro převod vstupního řetězce na množinu tokenů a následně parser, který vytváří objektovou podobu zprávy. Za využití jednoduchého větvení jsou pak podle druhu zprávy provedeny adekvátní kroky. Třída je stěžejním prvkem aplikace a obsahuje hned několik atributů, mimo fronty zpráv a semaforu jsou to hashovací tabulka sloužící pro ukládání pozičních zpráv o vlacích, tabulka předjížděných vlaků, parser a klienta realizovaného v třídě *ClientSendMessage* sloužícího pro vytváření a odesílání zpráv simulačnímu nástroji OT.

V případě, že je z fronty zpráv odebrána poziční zpráva vlaku a má být provedeno vyhodnocení, zda se má na některé dvojici vlaků na trati provést předjíždění, dochází k vytvoření nového vlákna implementovaného v třídě *OvertakingThread*. Je mu předána za pomoci konstruktoru tabulka všech vlaků na trati a mimo tu i tabulka sloužící pro evidování aktuálně předjížděných vlaků. Neboť před vložením do této tabulky je zapotřebí ověřit, zda se tento záznam již v tabulce nevyskytuje. Pokud je dvojice vlaků již v tabulce, neprovádí se nic, stejně tak při variantě, kdy se vlaky neomezují. Když se však záznam v tabulce nevyskytuje, je zapotřebí tuto dvojici předjížděných uložit. Tím končí náplň tohoto vlákna. Následuje ukázka vyhodnocování předjíždění nad seřazenou tabulkou *zpravy*, která nese poziční zprávy vlaků v identickém pořadí, v jakém se pohybují po trati. Z toho důvodu postačí provádět kontrolu vždy dvojice vlaků následující v seznamu přímo po sobě.

```
private void detectDistance() throws Exception {
    TrainPositionReport trainA = null;
    TrainPositionReport trainB = null;
    int speedA = 0;
    int speedB;
    Iterator iter = zpravy.iterator();
    if (iter.hasNext()) {
        trainA = (TrainPositionReport) iter.next();
        speedA = getMaxSpeed(trainA.getTrainID());
    }
    while (iter.hasNext()) {
        trainB = trainA;
        trainA = (TrainPositionReport) iter.next();
        speedB = speedA;
        speedA = getMaxSpeed(trainA.getTrainID());
        if (speedA > -1 && speedB > -1 && speedA < speedB &&
        (speedA < DETEKCNI_MAXIMALNI_RYCHLOST || trainA.getTra-
        inID().substring(0, 2).equals("OS"))) {
            int rozestup = getDistance(trainA, trainB);
            if (rozestup > 0 && rozestup < DETEKCNI_ROZESTUP)
                overtake(trainA, trainB);
            }
        }
    }
}
```

Ukázka kódu 3: Detekce předjíždění

5.4.2 Datové struktury

Součástí aplikace jsou i datové struktury. První použitou datovou strukturou je fronta sloužící pro ukládání příchozích zpráv ze strany simulátoru. Je využívána v pořadí prvním zmíněným vláknem implementovaným ve třídě *ServerThread*. Tato třída zastává realizaci serveru

a naslouchá příkazům na portu 4002. Poté co vstupní zprávy triviálním způsobem třída analyzuje jejich klíčovou část, vkládá získaná data v podobě řetězce do fronty příchozích zpráv. Jedná se o jednosměrně zřetězenou abstraktní datovou strukturu, která využívá dvou referencí na začátek a konec. Dvě reference jsou použity z důvodu zlepšení asymptotické složitosti oproti realizaci s jednou referencí. Metody *vloz* a *odeber* pak získávají shodnou konstantní složitost $O(1)$. Datová struktura fronty dále obsahuje metody *jePrazdny* sloužící pro ověření prázdnosti a metodu *zrus* pro smazání všech prvků ve frontě. Díky tomu, že programovací jazyk Java disponuje garbage collectorem, není nutné procházet prvky při mazání seznamu. Pouze stačí odstranit vstupní reference v tomto případě výše zmíněné na začátek a konec seznamu a díky tomu má i tato metoda konstantní asymptotickou složitost.

Dále je v práci použita hashovací tabulka. Na tuto datovou strukturu byly kladeny specifické požadavky z důvodů jejího uplatnění. Slouží pro předávání dat mezi dvěma vlákny. První vlákno implementované v třídě *FiltrMessageThread*, které odebírá zprávy z fronty a následně provádí jejich třídění podle druhu příchozí zprávy. Stěžejními informacemi jsou oznámení o poloze vlaku. Tyto zprávy jsou vkládány právě do hašovací tabulky. Nároky na tuto datovou strukturu byly:

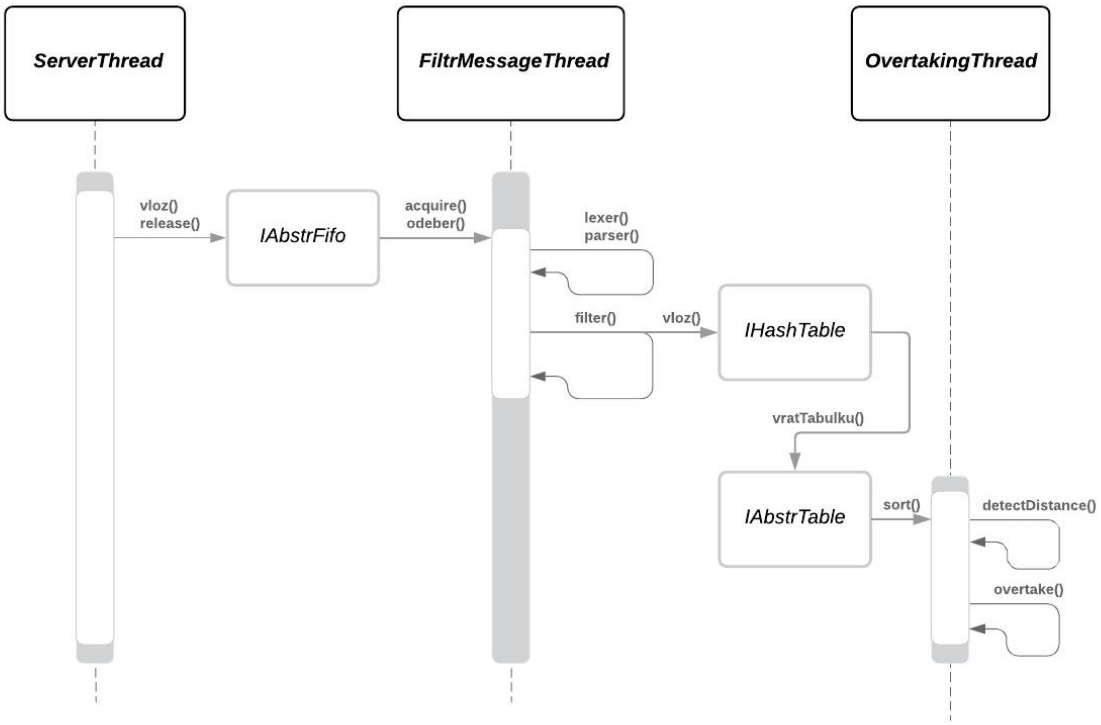
- udržet jedinečnost zprávy podle klíče,
- realizovat nejnižší možnou asymptotickou složitost pro vložení prvku,
- měla by umožňovat seřazení prvků.

Prvním požadavkem je myšleno, že v případě příchodu zprávy s identickým klíčem nebude zpráva uložena jako nová položka, ale pouze aktualizuje stávající. Díky tomuto mechanizmu nikdy nedojde k vytvoření duplicitního záznamu o tomtéž vlaku na trati. Druhý požadavek je kladen na nízkou složitost při vložení. Zprávy od vlaků přichází v nastavených intervalech, pro kvalitní detekci je tento interval stanoven na jednu sekundu. Četnost příchozích zpráv obzvláště při tradičním zrychlení simulace je ohromná, a právě z tohoto důvodu je kladen důraz na minimální operační náročnost algoritmu. Na základě těchto požadavků byla vybrána hashovaní tabulka. Poslední požadavek byl předán dál a tato struktura jej neřeší. Pouze umožňuje vytvořit tabulku realizovanou na seznamu. Tato metoda *vratTabulku* slouží pro získání hodnot z hashovací tabulky. Takto jsou data předána konstruktoru třídy *OvertakingThread*. Kromě zmíněných metod *vratTabulku* a *vloz* je součástí třídy i *velikost*, *zrus* a *jePrazdny*. Všechny tři zmíněné metody se vyznačují konstantní asymptotickou složitostí.

Poslední již zmíněnou datovou strukturou je tabulka definována na spojovém seznamu. Stejně jako předchozí se jedná o abstraktní datovou strukturu ovšem oproti frontě je realizována

s využitím obousměrně zřetězeného seznamu. Tabulka je uplatněna na dvou místech, prvním je seznam dvojic předjížděných vlaků a druhým pak je seznam pozičních zpráv vlaků. Na tabulku pozičních zpráv je požadována schopnost řazení a proto obsahuje třída *AbstrTable* metodu *sort* využívající comparator, který metoda obdrží jako vstupní parametr. Comparator *ComparatorMessageTrainPositionReport* slouží pro seřazení zpráv podle názvu úseků a při souladu úseku provádí porovnání na základě parametru *routeOffset* udávajícího vzdálenost čela vlaku od počátku daného úseku. Takto seřazená data jsou v tabulce předány pro závěrečné vyhodnocení, zda bude provedeno předjetí. Prvním zmíněným příkladem uplatnění tabulky je ukládání informace o předjížděných vlacích. Do tohoto seznamu je přidán záznam, pokud došlo ke kladnému vyhodnocení. K záznamu je následně vložena stanice, ve které první předjížděný vlak zastavil a čeká. Následně až při provedení předjetí druhým vlakem je tento záznam z tabulky smazán, a tím je proces předjíždění završen.

Níže je umístěn zjednodušený sekvenční diagram, který demonstruje průběhy jednotlivých vláken společně s klíčovými datovými strukturami. V diagramu jsou umístěny i základní metody popisující nejdůležitější kroky programu.



Obrázek 12: Sekvenční diagram

5.4.3 Scanner a parser

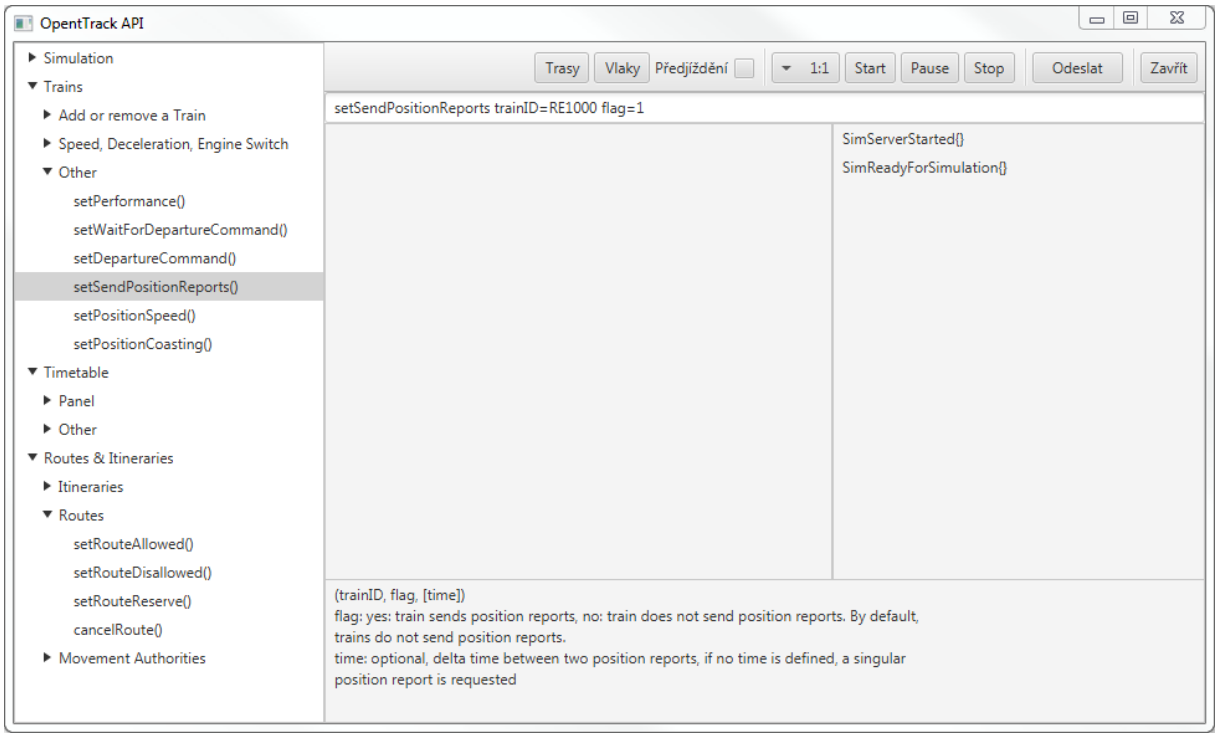
Zprávy je zapotřebí pro snadnější manipulaci převést do objektové podoby, a právě tato problematika bude nyní blíže popsána. Zprávy přijaté od simulátoru jsou nejprve z podoby vstupního proudu převedeny do řetězce znaků. Jedna zpráva obsahuje vždy pouze jednu nesenou informaci. Nikdy nedochází k situacím, že by v rámci jedné zprávy byla zaslána například poziční zpráva o vlaku společně s oznámením, že jiný vlak dorazil do stanice. Díky tomu lze snadným způsobem s využitím metody *indexOf*, ověřit zda zvolený řetězec znaků obsahuje daný podřetězec. Zprávy zaslané od simulačního nástroje OT mají kromě nesené informace v těle neměnnou formu a díky tomu lze snadno odstranit nedůležité části zprávy.

Nad tímto řádkem je provedena lexikální analýza. Je zde zapotřebí zdůraznit, že pro správnost by bylo vhodné provést analýzu celé zprávy a korektně tak kontrolovat a zpracovávat přijímané informace, nicméně cílem je navrhnout efektivní zbytečně nezatěžovaný algoritmus. Zprávy jsou navíc zasílány ze strany simulačního nástroje a nepředpokládá se syntaktická chyba v těchto oznámeních. Zpráv může být přejímáno až v řádů stovek za sekundu a tím pádem je v této části kódu nutné dávat efektivitě vyšší prioritu.

Tokeny, které vytvořil scanner, jsou předány syntetickému analyzátoru parseru. Ten přetvoří tokeny do objektů, kdy hlavní je třída *Message*, od té následně dědí ostatní třídy. Každá třída je určena pro jeden druh zprávy. Parametry zprávy se vždy shodují s atributy třídy. Platí, že ne všechny zprávy je zapotřebí zpracovávat, proto je součástí i obecná třída, pod kterou se skrývají ostatní pro aplikaci nezajímavé zprávy. Zpracované zprávy se ihned filtrují podle svého druhu a provádí se adekvátní reakce API.

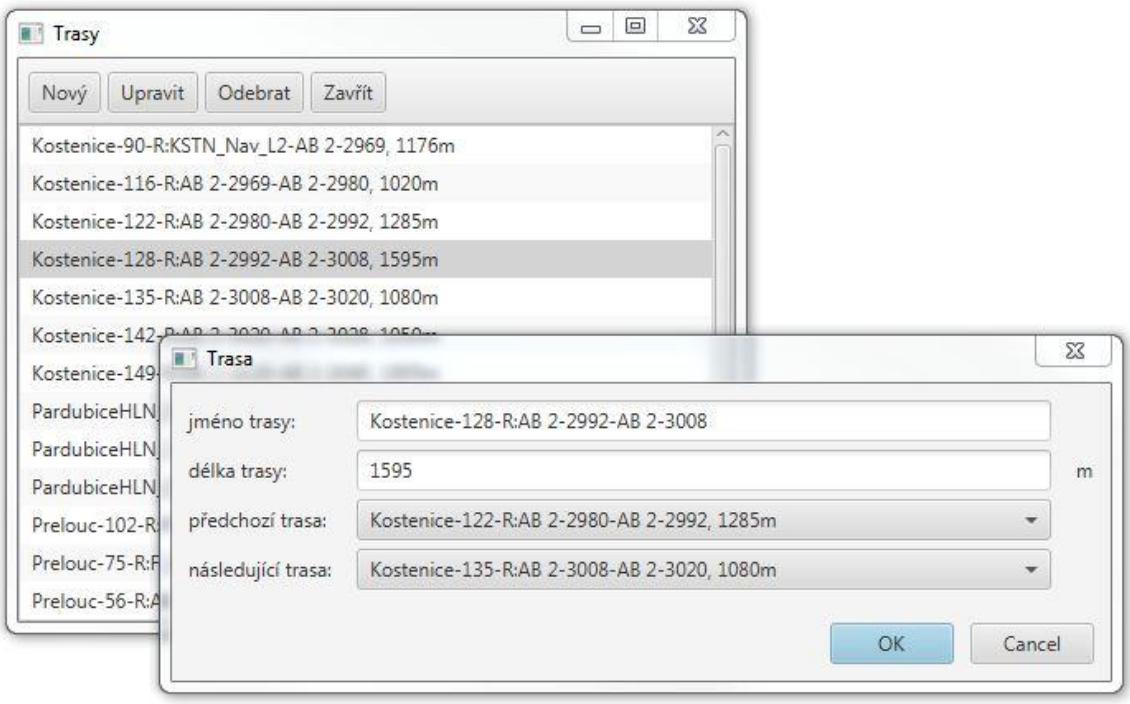
5.5 Grafické rozhraní

GUI slouží pro dva základní účely. Prvním je umožnit uživateli zasílat manuálně zprávy simulačnímu programu. Druhým pak je schopnost realizovat automatické předjíždění. V levé části hlavního okna aplikace jsou umístěny v hierarchické struktuře veškeré příkazy programu OT. Panel na pravé straně zobrazuje uživateli přijaté zprávy od simulátoru. Ve středové části jsou zobrazovány manuálně odesílané zprávy. Zprávy odesílané v rámci automatického předjíždění se zde nevyskytují. Spodní panel slouží pro zobrazování informací o příkazech. Poslední a nejvýznamnější částí je panel nástrojů umístěný v horní části okna. Mimo tlačítek pro zavření a odeslání zprávy z konzole obsahuje i tlačítka pro ovládání běhu simulace. Lze libovolně nastavovat rovněž rychlost běhu simulace.



Obrázek 13: Ukázka hlavního okna aplikace

Posledními prvky jsou nástroje sloužící pro realizaci automatického předjíždění. Při použití tlačítka „Trasy“ dojde k zobrazení dialogového okna, které slouží pro zadávání jednotlivých úseků na trati a jejich vzájemné navázání. „Vlaky“ pak slouží pro zadání maximálních rychlostí všech vlaků, u kterých vyžadujeme jakoukoli účast v procesu předjetí.



Obrázek 14: Ukázka editace trasy

5.6 Aplikování na modelu hlavního nádraží Pardubice

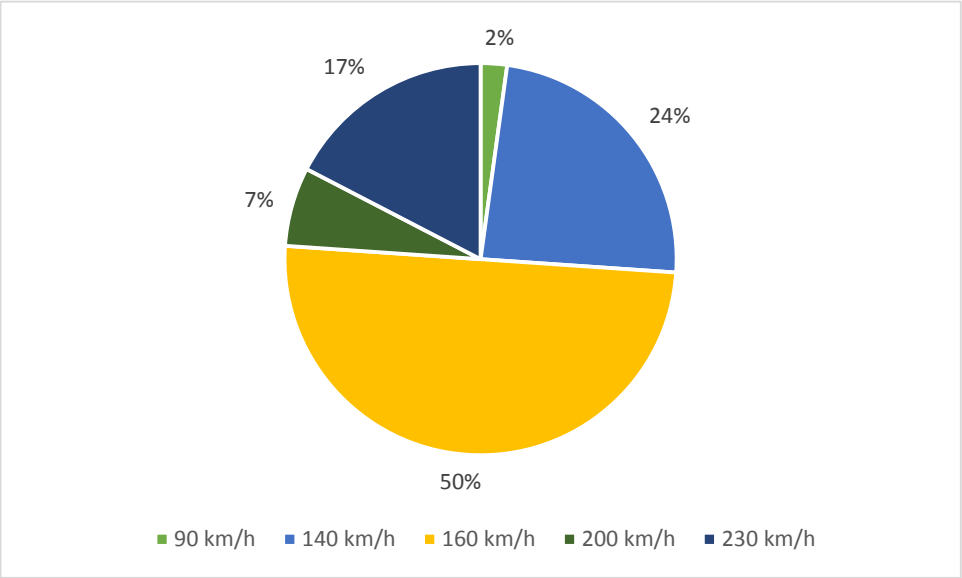
Model realizoval v rámci diplomové práce *Vyhodnocování odlišných variant železničního provozu s využitím počítačové simulace* toho času Bc. Jakub Zatloukal v programu OT. Je zde realizováno hlavní nádraží v Pardubicích s přilehlými traťovými úseky a zjednodušenými stanicemi Kostěnice, Přelouč a Pardubice-Rosice nad Labem.. Předjíždění bude použito pouze v rámci železniční stanice Pardubice hlavní nádraží a to ze směrů od Kostěnic a Přelouče. Simulace modeluje dopravu od 6:00 do 9:00.

Pro realizování předjíždění je zapotřebí zadat aplikaci tři základní vstupní data. Jsou jimi:

- struktura se vzdálenostmi úseků,
- maximální rychlost vlaků,
- detekční vzdálenost.

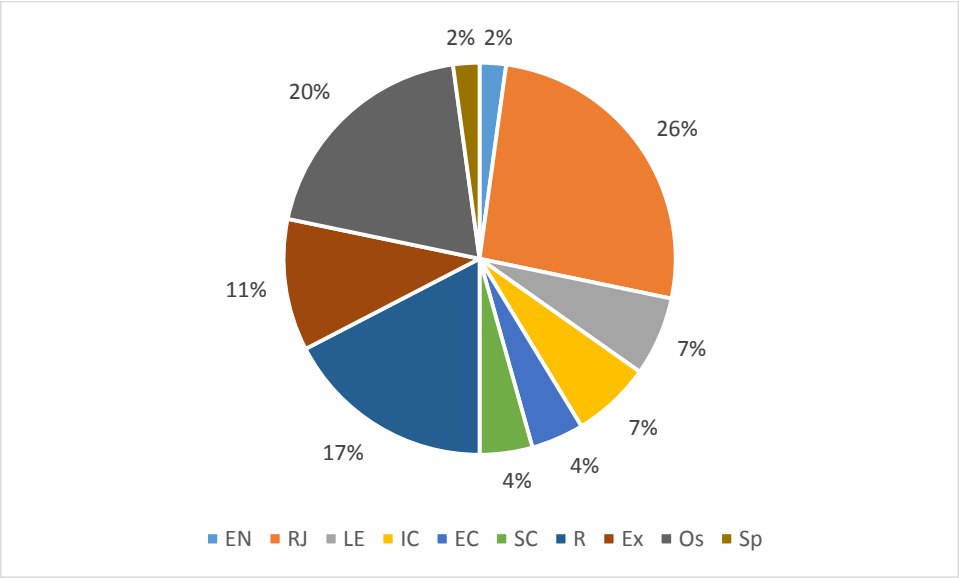
Jak mohou vypadat zadané úseky, zobrazuje Obrázek 14: Ukázka editace trasy. Zadané délky jednotlivých úseků slouží pro výpočet vzdáleností mezi vlaky. Součástí modelu je 46 vlaků, kdy se jedná o vlaky RJ, LE, IC, EC, SC, R, Ex, Os a Sp. Pro realizování předjetí je zapotřebí

znát vždy druh a maximální rychlost vlaku. Na níže uvedeném grafu je vyobrazena četnost maximálních rychlostí jednotlivých vlaků.



Graf 1: Maximální rychlosti vlaků

Z grafu je patrné, že v simulaci je prostor pro předjíždění, z důvodu maximální rychlosti vlaků. Alternativním důvodem pro realizaci předjetí je druh vlaku. Osobní vlaky staví ve všech stanicích i zastávkách, tedy je vhodné provést jejich předjetí, aby nedocházelo ke zdržování vlaků jiných druhů jedoucí za takovýmto osobním vlakem. Následující graf zobrazuje četnost jednotlivých druhů vlaků.



Graf 2: Druhy vlaků

Posledním nastavitelným kritériem je detekční vzdálenosti. V případě železničního úseku Kostěnice–Pardubice je vzdálenost cca 10 km a v opačném směru na úseku Přelouč–Pardubice je tato vzdálenost cca 13 km. Jedná se o omezení modelu, a tak maximální možná detekční vzdálenost vyplývá ze směru, ze kterého vlak přijíždí do stanice.

Na následujících tabulkách jsou zobrazeny hodnoty pro exponenciální rozdělení, podle kterého budou v modelu definována zpoždění jednotlivých vlaků.

	Mean	**Maximum**	**Lambda**
Os	270 s	7200 s	33 %
Sp	300 s	7200 s	45 %
ostatní	420 s	7200 s	50 %

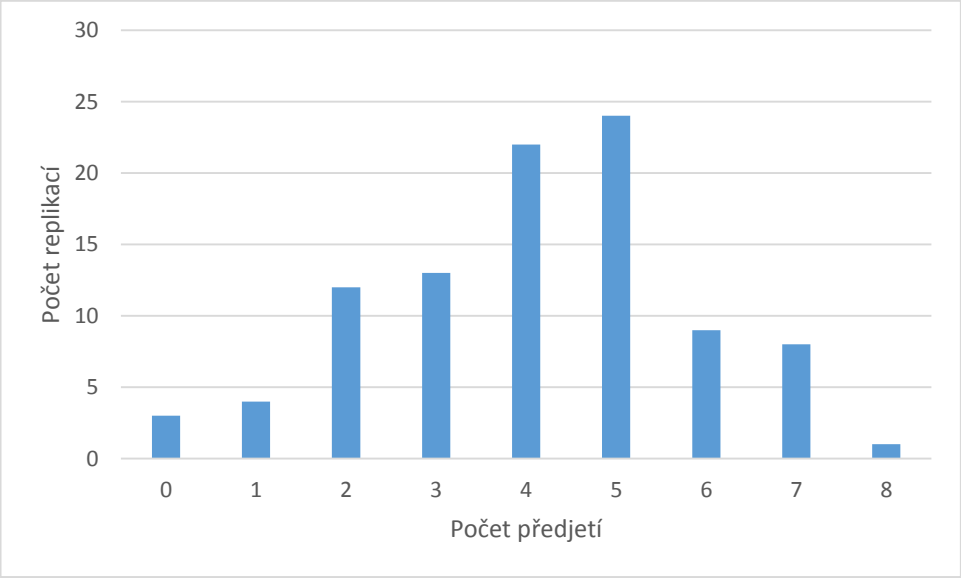
Tabulka 1:Vstupní zpoždění

	Mean	Maximum	Lambda
Os a Sp	60 s	300 s	5 %
ostatní	30 s	300 s	5 %

Tabulka 2: Zpoždění ve stanicích

5.6.1 Statistické vyhodnocení

Vyhodnocení je zaměřeno především na četnost realizovaných předjetí v rámci použitého simulačního modelu, které je zobrazeno na následujícím grafu. Z grafu vyplývá, že nutnost předjíždění není v žádném případu ojedinělou potřebou. Za pouhé 3 hodiny simulovaného času v rámci jedné replikace došlo v průměru k 4,15 předjetím kdy modem je 5 předjetí.



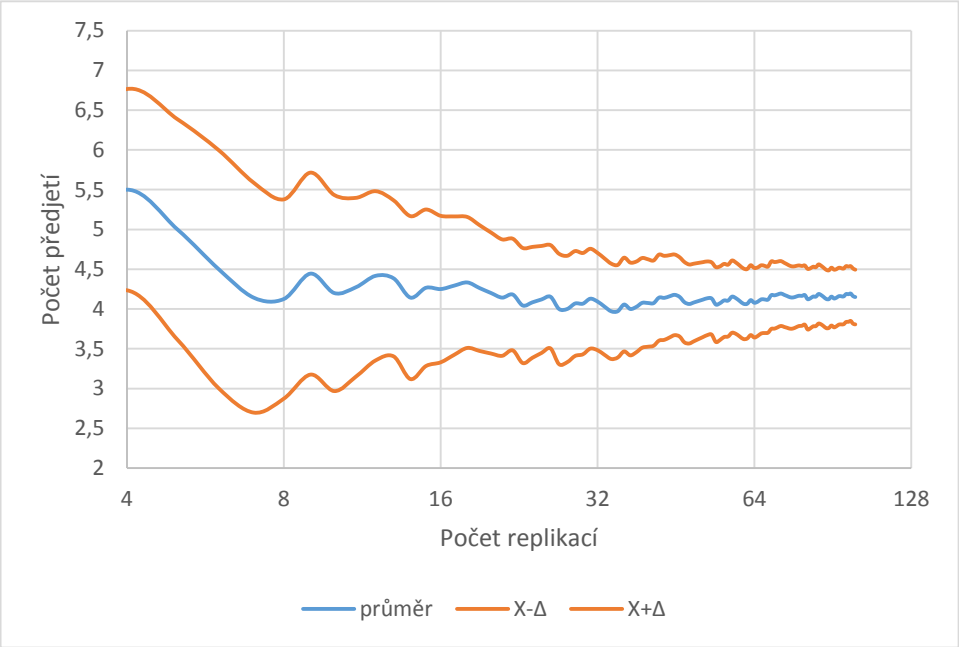
Graf 3: Četnost předjíždění

Následuje ukázka všech předjetí v rámci jedné replikace. V levé části tabulky jsou umístěny dvojice vlaků v pořadí, v jakém byli na vstupu do simulátoru. Pravá část tabulky pak zachycuje jejich pořadí po předjížděcím manévru. Poslední dvojice záznamů dokládá, že lze realizovat i dvojité předjetí jednoho osobního vlaku. Kdy vlak Os 5022 byl předjet vlakem R 894 Zlínský expres a následně i vlakem RJ 1002.

Pořadí vlaků na vstupu			Pořadí vlaků na výstupu	
Os 5002	EN 442 Slovakia	→	EN 442 Slovakia	Os 5002
Os 5003	Ex 121 Valašský expres	→	Ex 121 Valašský expres	Os 5003
Os 8655	Ex 113 Varsovia	→	Ex 113 Varsovia	Os 8655
Os 5022	R 894 Zlínský expres	→	R 894 Zlínský expres	Os 5022
Os 5022	RJ 1002	→	RJ 1002	Os 5022

Tabulka 3: Ukázka předjíždění v rámci jedné replikace

Následující graf zobrazuje vývoj průměru, který je ohraničený intervalem spolehlivosti, pro který byl zvolen kvocient 0,05. Díky tomu došlo k získání úrovně spolehlivosti 95 %.



Graf 4: Interval spolehlivosti

Bohužel jen těžko lze prokázat výslednou účinnost předjíždění, i když se na první pohled jeví jako přínosná. Specifikování vstupních parametrů pro předjetí je důležitou otázkou, která značným způsobem ovlivňuje výsledné hodnoty, ovšem autor práce není odborníkem na vlakovou dopravu, proto tuto problematiku přenechává pro případný další výzkum.

ZÁVĚR

V první části práce byl analyzován program OpenTrack s konkrétními příklady uplatnění a snahou odhalit prostor pro realizaci aplikační nadstavby. Následně byly popsány konkrétnější příklady použití API s ukázkami komunikace a popisem průběhu zpracování.

Dále byla popsána realizace vlastní API zaměřená na řešení problematiky předjíždění ve vlakové dopravě. Hlavním cílem bylo implementovat efektivní, pokud možno čistý kód. Autor se snažil uplatňovat vlastní datové struktury, a především vláknové programování s cílem realizovat výkonnou aplikaci, která nebude uživatele svými parametry nikterak omezovat. Výsledné statistické shrnutí v závěru práce potvrzuje předpoklad, že za pomoci aplikační nadstavby lze skutečně realizovat automatické předjíždění.

Autor tak splnil cíle, kterými bylo uplatnit aplikační nadstavbu pro realizaci konkrétní problematiky nad reálným simulačním modelem, který poskytl v rámci spolupráce Bc. Jakub Zatloukal. Rovněž bylo provedeno statistické vyhodnocení dokládající funkčnost a použitelnost této API společně se simulačním nástrojem OpenTrack.

Otázky nastavení vstupních parametrů jsou spíše směřovány na odborníka z oboru vlakové dopravy. Cílem této práce bylo především vytvořit efektivní, přenositelný nástroj, nežli realizovat z pohledu řízení dopravy bezchybnou API pouze nad konkrétním modelem.

POUŽITÁ LITERATURA

[1] ADAMEC, Tomáš. *Využití softwaru OpenTrack pro simulaci provozu metra* [online]. Pardubice, 2018 [cit. 2019-04-13]. Dostupné z: https://dk.upce.cz/handle/10195/71202. Bakalářská práce. Univerzita Pardubice, Dopravní fakulta Jana Pernera. Vedoucí práce Petr Nachtigall.

[2] Client–server model. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, ©2019. Dostupné z: https://en.wikipedia.org/wiki/Client%E2%80%93server_model

[3] FOJTŮ, Ondřej. *Modelování provozních parametrů posunovacích lokomotiv* [online]. Pardubice, 2018 [cit. 2019-04-13]. Dostupné z: https://dk.upce.cz/handle/10195/71237. Diplomová práce. Univerzita Pardubice, Dopravní fakulta Jana Pernera. Vedoucí práce Petr Nachtigall.

[4] HUERLIMANN, Daniel a Andrew B. NASH. *OpenTrack Manual: Simulation of Railway Networks*. Curych, 2019.

[5] *Java Platform, Standard Edition 8 API Specification* [online]. Redwood City: Oracle Corporation, ©2019 [cit. 2019-05-01]. Dostupné z:https://docs.oracle.com/javase/8/docs/api/index.html

[6] KAVIČKA, Antonín. *Vymezení základních pojmů z oblasti modelování a simulace.* Pardubice, ?2011.

[7] KLAUZ, Milan. Programy pro obvodové simulace. *DPS: Elektronika od A do Z* [online]. ©2019 [cit. 2019-03-18]. Dostupné z: www.dps-az.cz/cad-cam-cae/id:1709/programy-pro-obvodove-simulace.

[8] KRACÍK, Pavel. *Návrh zvýšení propustné výkonnosti tratě 501 v úseku Přelouč - Pardubice hl.n.* [online]. Pardubice, 2018 [cit. 2019-04-13]. Dostupné z: https://dk.upce.cz/handle/10195/71214. Diplomová práce. Univerzita Pardubice, Dopravní fakulta Jana Pernera. Vedoucí práce Petr Nachtigall.

[9] KUBÁT, Bohumil a Lukáš TÝFA. *Železniční tratě a stanice*. Vyd. 2., přeprac. Praha: Vydavatelství ČVUT, 2003. ISBN 80-01-02782-1.

[10] MOJŽÍŠ, Vlastislav a Tatiana MOLKOVÁ. *Technologie a řízení dopravy I: část železniční doprava*. Pardubice: Univerzita Pardubice, 2002. ISBN 80-7194-424-6.

[11] NOVÁK, Vladimír a Pavel ZÍTEK. *Praktické metody simulace dynamických systémů*. Praha: Státní nakladatelství technické literatury, 1982.

[12] *OpenTrack Railway Technology* [online]. Curych: OpenTrack, ©2016 [cit. 2018-12-07]. Dostupné z: http://www.opentrack.cz/opentrack_cz.html

[13] *OpenTrack: OTD-Version: Using the OpenTrack API.* Curych, 2019.

[14] *Porovnání řadicích algoritmů.* Algoritmy.net [online]. [cit. 2019-04-21]. Dostupné z: https://www.algoritmy.net/article/75/Porovnani-algoritmu.

[15] *Simulation software OpenPowerNet* [online]. Drážďany: IFB GmbH Dresden, 2018, 25. 9. 2018 [cit. 2018-12-07]. Dostupné z: http://openpowernet.de/

[16] TISCHER, Erik. *Simulace automatického provozu na trase metra B* [online]. Pardubice, 2017 [cit. 2019-04-13]. Dostupné z: https://dk.upce.cz/handle/10195/68797. Diplomová práce. Univerzita Pardubice, Dopravní fakulta Jana Pernera. Vedoucí práce Petr Nachtigall.

[17] VONKA, Jaroslav, Tatiana MOLKOVÁ a Jaromír ŠIROKÝ. *Technologie a řízení dopravy II.* - GVD. Pardubice: Univerzita Pardubice, 2000. ISBN 80-7194-286-3.

[18] *W3schools: XML Soap* [online]. Refsnes Data, ©2019 [cit. 2019-03-11]. Dostupné z: https://www.w3schools.com/xml/xml_soap.asp

PŘÍLOHY

PŘÍLOHA A – CD

Součástí CD je práce ve formátu PDF, simulační model v programu OpenTrack, aplikační nadstavba a statistické vyhodnocení v programu Excel.