České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Michal F u k s a

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Simulace chování leteckého dispečera při řazení letadel před vstupem do koncové oblasti

Pokyny pro vypracování:

1. Seznamte se s metodou řazení letadel před vstupem do koncové oblasti nazvané "Miles-in-trail" (MIT) a její implementaci na oblastních řídících pracovištích.
2. Prostudujte architekturu systému AgentFly, který umožňuje simulovat letecký provoz a také chování lidského operátora obsluhujícího pracoviště oblastního řízení.
3. Navrhněte algoritmus chování lidského operátora simulující implementaci MIT metody.
4. Tento navržený algoritmus implementujte v simulačním systému AgentFly jako další přídavný modul s možností konfigurace potřebných parametrů.
5. Implementovaný algoritmus pomocí simulace otestujte a demonstrujte jeho správnou funkcionalitu.

Seznam odborné literatury:

[1] Michael S. Nolan: Fundamentals of Air-Traffic Control, Thomson, 2004.
[2] Federal Aviation Administration: Air-traffic procedures
[3] Šišlák, D.; Pěchouček, M.; Volf, P.; Pavlíček, D.; Samek, J.; Mařík, V.; Losiewicz, P.: Agentfly: Towards Multi-Agent Technology in Free Flight Air Traffic Control. Defence Industry Applications of Autonomous Agents and Multi-Agent Systems. Birkhauser Verlag, 2008.

Vedoucí bakalářské práce: Ing. David Šišlák, Ph.D.

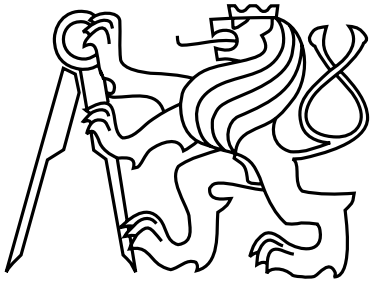
Platnost zadání: do konce zimního semestru 2012/2013

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 9. 1. 2012

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra kybernetiky



Bakalářská práce

Simulace chování leteckého dispečera při řazení letadel před vstupem do koncové oblasti

Michal Fuksa

Vedoucí práce: David Šišlák PhD.

Studijní program: Otevřená informatika, Bakalářský

Obor: Informatika a počítačové vědy

25. května 2012

Poděkování

Chtěl bych poděkovat především panu doktorovi Davidu Šišlákovi za trpělivost a pánům inženýrům Přemyslu Volfovi a Dušanu Pavlíčkovi za spolupráci a podporu. Pánům Hrstkovi, Hlavatému a Šálkovi z rozličných důvodů.

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

V Praze dne 24. 5. 2012

Fuksa

.....

Podpis

Abstract

Avionic transport became the most important and fastest-growing form of transport and justifiably gets a lot of attention. Its further growth forces the scientific research on new methods of handling avionic transport.

This bachelor thesis deals with the implementation of a module for the multi-agent simulation program called AgentFly, written in Java, and further adjustments needed to implement this functionality. The module extends the existing system of avionic transport simulation on a global scale by adding the ability to apply the method of flight control used to spread out peak hours on transportation bottlenecks – like e.g. airports – known as Miles In Trail. The results of the AgentFly project will serve as a base to new methods of air traffic control.

Abstrakt

Letecká doprava se stala nejdůležitější a nejrychleji rostoucí formou dopravy a dostává po právu mnoho pozornosti. Její další růst vynucuje vědecké zkoumání nových postupů jak vývoj letecké dopravy zvládat.

Tato bakalářská práce pojednává o implementaci modulu pro multiagentní simulační program AgentFly psaném v programovacím jazyce Java, a dalších úpravách nutných pro zavedení této funkcionality. Modul rozšiřuje dosavadní systém simulující leteckou dopravu v celosvětovém měřítku o schopnost aplikovat metodu řízení letového provozu sloužící k rozmělnění dopravních špiček u dopravních omezení, jakými jsou kupříkladu letiště, známou pod anglickým názvem Miles In Trail. Na výsledcích z projektu AgentFly budou založeny nové postupy pro řízení letecké dopravy.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Letecká doprava se v posledních desetiletích rozvíjela a stala jednou z nejdůležitějších dopravních metod pro přepravu zejména lídí, ale i nákladu. [1] V dnešní době se letecky přepraví více jak 9,5 miliónů lidí denně, což je 360 krát více než před 60ti lety. Vůbec největší systém letecké dopravy leží ve spojených státech. Nad 15 tisíci letišti se zde v každou chvíli vznáší až 4 tisíce letadel přenášející 61 tisíc cestujících. Takto velké množství letadel je třeba systematicky koordinovat tak, aby nedocházelo ke kolizím, ale aby zároveň délka letu se příliš neprodloužila a aby důležité části letového provozu, jakými jsou letiště, byly co nejlépe využity. O toto vše se snaží a zaručuje v každém státě zdejší organizace zajišťující řízení letového provozu. Pro Spojené státy americké tím je Americký úřad pro letectví (FAA Federal Aviation Administration). Jednou z funkcí řízení letového provozu je vyvažování zatížení důležitých vzdušných dopravních uzlů. Přirozeně nemůže například letiště zvládnout množství letadel, jaké je schopno letovým prostorem k němu přiletět. Tomuto problému čelí řízení letového provozu několika metodami, z nichž některé budou probrány v druhé kapitole a právě vyvažování hustoty letového provozu je širším a obecnějším předmětem této práce. Jednou z těchto metod je metoda Miles In Trail (MIT)[2], která slouží k rozmělnění dopravních špiček na letištích ve vytížených oblastech tak, že omezí množství přilétajících letadel na zvladatelnou míru. Za pomoci sousedních sektorů[1], které efekt MIT aplikují, nedojde k přehlcení cílového sektoru. Jakým způsobem toho okolní sektory docílí není pojato v definici MIT a je individuální. MIT a jeho implementace je zde předmětem. Letecká doprava, které je statisticky nejbezpečnější dopravou vůbec, každoročně roste a od vzniku koordinovaného řízení letového provozu je potřeba stále používat modernější techniku a vymýšlet nové postupy k zefektivnění tohoto druhu dopravy.

Jeden ze způsobů jak navrhovat a testovat nové postupy a metodiky v řízení letového provozu je počítačová simulace a jedním z takových systémů je AgentFly. [3]AgentFly je systém konstruovaný pro americký úřad pro letectví právě za účelem výzkumu řízení letového provozu, pro jeho následné vylepšení. Aby bylo možné výsledky práce nad simulačním systémem brát jako relevantní pro reálnou aplikaci je potřeba vytvořit co nejpřesnější a nejvěrohodnější model reality. AgentFly používá za tímto účelem multiagentní přístup. Fyzikálním model,

[1] Sektor je jednotka administrativního dělení letového prostoru. Letový prostor je vymezen letovými hladinami a horizontální hranicí

založený na datech z BADA projektu od Eurocontrol[4], definuje letecké vlastnosti všech letadel v simulaci a skrze něj se přes fyzické letové trasy letadel dopočítává vše důležité pro nahrazení reality. Každý subjekt v letovém provozu je v simulaci nahrazen agentem, malým kouskem kódu který ovládá na základě vstupů z fyzikálního modelu a od ostatních agentů přidělené letadlo. Největší pozornost však dostává implementace virtuálního ekvivalentu pro letové středisko. Každá relevantní osoba na letovém středisku je implementována jedním agentem zodpovědným za napodobení j funkce opravdového osoby. V této práci se zabýváme jednou z funkcí agenta, který simuluje činnost řídícího letového provozu.

Jednou z činností řídícího letového provozu je zmíněný Miles In Trail. Podle uživatelsky editovatelné konfigurace agenti obdaření MIT funkcionalitou provádějí podobné příkazy, jaké provádějí opravdoví řídící letového provozu. Rozsah funkcionality MIT je pouze v rámci jednoho sektoru.

Každý agent operuje skrze akce a reakce. Každá událost uvnitř simulovaného světa může evokovat reakci agenta, jehož činnost může a většinou působí jako akce pro některého jiného agenta. Řetězením těchto akcí a reakcí spolu s ubíhajícím časem vzniká simulace. Složitější agenti, právě jako agent řídícího se skládá ze základní implementace agenta schopného začlenit se do simulace a rozhraní pro aplikaci funkčních modulů, které jsou mezi sebou nezávislé a dodávají agentovi jeho funkcionalitu.

Tato práce pojednává o implementaci modulu pro řídícího agenta, který bude vykonávat funkčnost za MIT a jeho testování v systému AgentFly. Hlavním problémem bylo začlenění a správná kooperace s ostatními moduly. Některé důležité součásti systému také nebyly připraveny na zavedení funkčnosti, jako například agenti jednotlivých letadel a rádiový komunikační protokol, byl nekompatibilní s potřebnými manévry.

V poslední kapitole jsou uvedeny výsledky testů na dvou scénářích, jednom ukázkovém a jednom založeném na reálném provozu.

Kapitola 2

Popis problému, specifikace cíle

- Popis řešeného problému, vymezení cílů DP/BP a požadavků na implementovaný systém.
- Popis struktury DP/BP ve vztahu k vytyčeným cílům.
- Rešeršní zpracování existujících implementací, pokud jsou známy.

2.1 Hustota civilní dopravy

Jak bylo řečeno v úvodu, letecká doprava narostla a dál roste do proporcí, kde je nutné předvídat a reagovat na situace dříve než nastanou. Hustota dopravy se projevuje na místech s nejvíce omezeními, letištích, proto všechny praktiky zabývající se problematikou hustoty dopravy vychází z maximální kapacity letišť. Maximální kapacita letišť se zvyšuje pouze s vysokými náklady, takže řešení problému se většinou orientuje na co nejefektivnější využití letištní kapacity.

V dnešní době se potíže se špičkami v hustotě dopravy řeší několika způsoby. Některé z těchto metod, souhrnně nazývány Traffic Flow Management (TFM), jsou:

- "Time Based Metering"na strategické úrovni
- "Holding patterns"pro sektor
- "Miles-In-Trail"na taktické úrovni.

2.1.1 Metering

Time-Based Metering (Metering)[5] je teoreticky nejefektivnější způsob vyvažování hustoty dopravy s minimálním dopadem na spotřebu paliva. Tato metoda simuluje předpokládaný průběh dopravy v celosvětovém měřítku s velkým předstihem. Vyhledává ideální odletové časy pro všechna letadla tak, aby nedocházelo k dopravním špičkám jak na cílovém letišti, tak v sektorech po trase. Dále během samotného letu je dopočítáváno aktuální zpoždění nebo

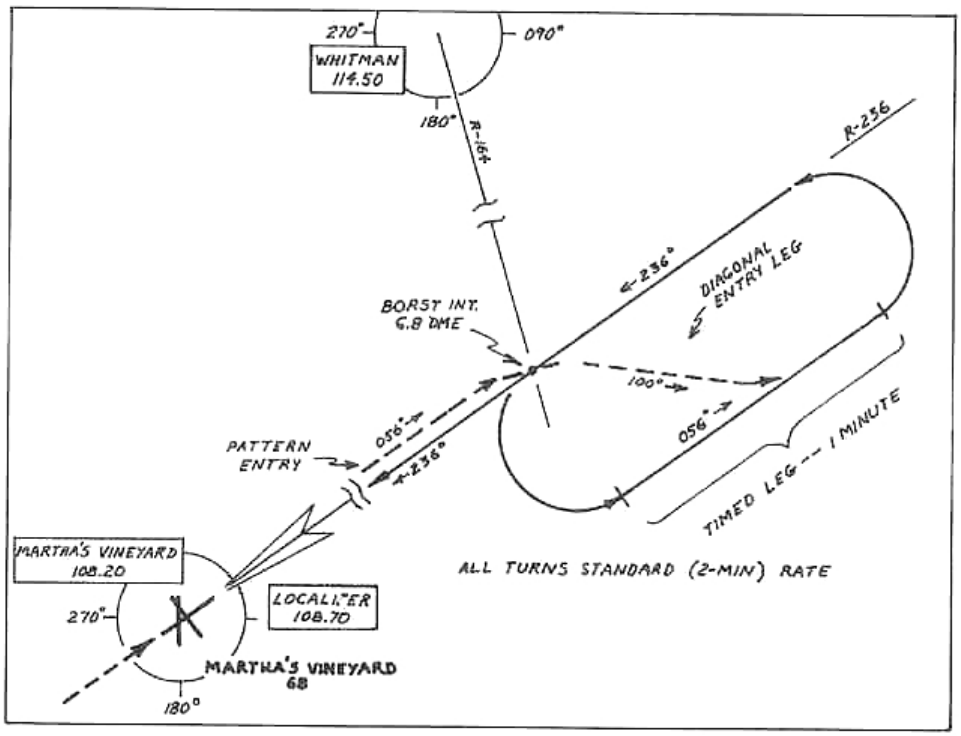
předstih podle dosavadního letu.S cílem zaručit limity přetížení důležitých sektorů může být let daného letadla upraven v souladem se zpožděním či předstihem.

Při Ground Delay Programu [6] dále podle odhadů zatížení letového prostoru dochází ke zpoždění odletu letadla na počátečním letišti.

Problémem je značně obtížná implementace a závislost na statistických datech a přesnosti výpočtu předpokládané trasy. Vše je samozřejmě silně ovlivněno individuálními událostmi, povětrnostními podmínkami, viditelností a dalšími.

2.1.2 Holding Patterns

Holding Patterns (HP)[7] je bezprostřední způsob jak pozdržet letadla, většinou ve finálním sektoru před přistáním přistáním. Nevyžaduje žádné plánování ani data. HP jsou pro každé letiště dopředu definované a známé, což ulehčuje zátěž pro rádio na pouze dvě instrukce. Letadlo zpomalí na definovanou rychlost a opakuje obraty o 180 °, většinou rovnoběžné s dráhou a mezi obraty letí přímo po danou dobu. Mezi nevýhody HP patří zbytečná spo-



Obrázek 2.1: Příklad definice Holding Pattern [8].

třeba. HP neřeší problém s množstvím letadel v sektoru. K druhé situaci, při níž se využívá Holding Patterns, dochází méně často a značí problémy se zvládáním letového provozu. Pokud v sektoru operátor má potíže s koordinací letadel, začne odmítat další letadla letící do jeho sektoru. Operátor sousedního sektoru řeší tuto nečekanou situaci právě pomocí holding

patterns. Stačí nadefinovat nový prozatimní Holding Pattern nad některým z FIXů v sektoru a může letadlo pozdržet. Na obrázku 2.1 je vidět definice Holding Pattern na přistávací dráhu na letiště Martha's Vineyard. Takovéto definice jsou dostupná pilotům před letem a bývají po dlouhou dobu nezměněny.

2.1.3 Miles In Trail

Miles In Trail (MIT)[2] je metoda vyvažování letecké dopravy svazující letadla letící do stejné oblasti, na jedno nebo více letišť. Používá se u větších, vytíženějších letišť. Vychází z faktu, že podle bezpečnostních předpisů musí mít letadla na vzletu i přistání dané časové rozstupy závisející na typu letadel. Cílový sektor se dohodne s každým z okolních sektorů na rozstupech mezi letadly letícími do stejné oblasti nebo na stejné letiště. Tato restrikce se dále propaguje do vzdálenějších sektorů s jinými daty (například jiné rozstupy) dokud je zapotřebí. Častý příklad, kdy na distribuci MIT restrikcí dalším sektorům záleží, je když se v sektoru spojují silné dopravní toky s předem známým,predikovaným počtem letadel v každém ze směrů v příštím časovém intervalu. MIT restrikce jsou pro tyto proudy nastaveny přesně nepřímo úměrně hustotě provozu. Dopravní špičky se tím rozmělní daleko od vytíženého sektoru. Myšlenku MIT naznačím:

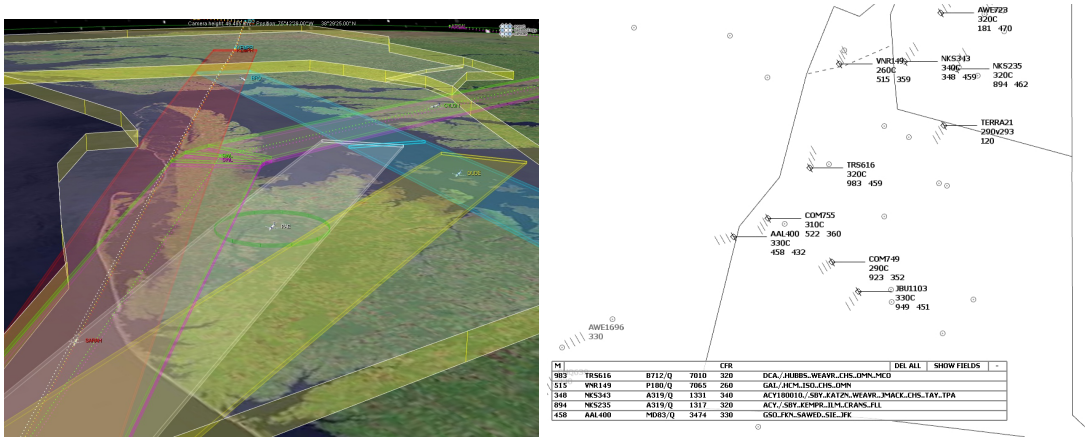
Pokud se do aktuálního sektoru blíží letadlo se záměrem přistát na některé z letišť, ale již pro něj není místo z důvodu množství letadel, musíme ho buď zdržet v našem sektoru nebo se musí zdržet v předchozím sektoru. Zdržení v našem sektoru bude znamenat jistou ztrátu paliva ale hlavně zátěž na sektorovou komunikaci. Není tedy výhodné letadlo vpouštět do sektoru pokud nemůže letět přímou trasou na přistání. Zajistěme si tedy minimální rozstupy dohodou s vedlejšími sektory, které zdržení provedou za nás s menší obtíží neboť jsou zpravidla méně vytížené.

V porovnání s Time-Based Meteringem je MIT jednodušší na implementaci a provedení. Kromě nepotřebnosti složitých CSP algoritmů nevyžaduje MIT letová data v takovém měřítku. Po stránce výkonu zaměstnává MIT pouze několik sektorů a je spolehlivý a efektivní.

2.2 AgentFly

AgentFly [3] je projekt, který vznikl v roce 2006 v Agent Technology Center (ATG) pod vedením Volf, Šišlák a Pěchouček. Jedná se o multi-agentní systém sloužící k rozsáhlým simulacím celosvětové letecké dopravy. Simulace používá přesné výkonové modely letadel a prostředí s maximální věrohodností. Všechny aspekty jako například povětrnostní podmínky, spotřeba paliva a bezpečné vzdálenosti letadel jsou brány v potaz. Systém také dovoluje měřit a vyhodnocovat množství parametrů a vlastností probíhající simulace, díky čemuž lze detailně analyzovat charakteristiky simulované letecké dopravy. AgentFly používá platformu AGLOBE, která vznikla v roce 2003 pod stejným vedením. Tento systém zajišťuje grafické rozhraní a vykreslování planety, letadel a dalších informací o simulaci. Na obrázcích 2.2 je vidět vizualizace AgentFly ve verzi 2.

Dále probíraná aplikace AgentFly se vztahuje k simulačním testům na systému řízení letového provozu(ATM) používaném National Airspace System (NAS) ve spojených státech



Obrázek 2.2: Reprezentativní obrázky grafického výstupu systému AgentFly. Render simulované reality vlevo a snímek obrazovky řídícího letového provozu s radarovým výstupem dopravy v sektoru.

[9]. Tento systém zahrnuje postupy a pravidla zajišťující bezpečný letecký provoz. Kapacita letové prostoru(LP) závisí na mnoha faktorech od schopností operátorů až po použité zařízení. Jelikož kapacita (LP) zůstává řádově stejná a letecká doprava dle očekávání, extrapolací z předchozího růstu nebo podle společnosti Boeing, která očekává ztrojnásobení počtu nákladních letů během příštích 20ti let, je potřeba ATM modernizovat. NextGEN (Council 1998)(Next-Generation Air Transportation Systems) je program ke koordinaci dalšího vývoje ATM ke schopnosti pojmout vzrůstající potřeby letecké dopravy. Další (spotřeba a efektivita)

Právě pro potřeby NextGENu je rozšiřován systém AgentFly. Pro použití k analýze konceptů NextGen.

Koncepty byly testovány skupinami řídících podle konceptu Human in the Loop, kde je řídícím představována simulace jejich práce a oni na ni reagují adekvátní činností. Výhodami použití reálných lidí jsou reálné reakce, které není třeba simulovat, ale kvůli náročnosti na lidskou pozornost lze HITL provádět pouze v omezeném rozsahu. V malém měřítku však nelze odhalit problémy složitých systémů. Relativně malý problém v malém měřítku může v rozšíření na globální měřítko vyrůst v relevantní problém. Tyto problémy vznikají tzv. kaskádovým efektem a právě ve velké simulaci jakou je AgentFly k ní nedochází.

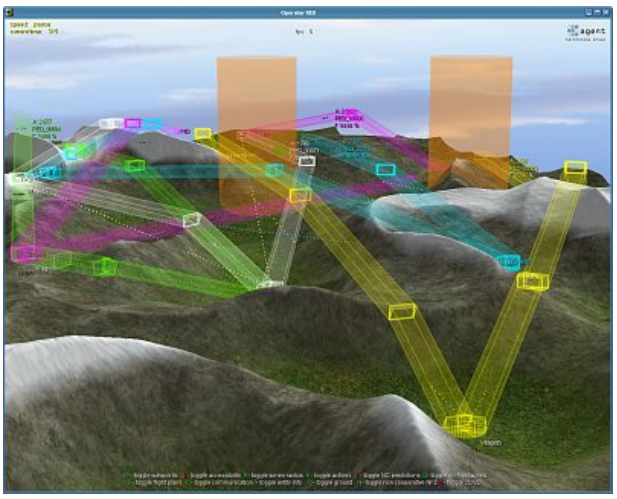
Náš simulační model se stává z GPS modelu reálného světa a modelu řídícího střediska.

2.2.1 Simulovaná realita

Představuje věrnou simulaci reálného světa ve 3D. Pozice letadel,sektorů ad. jsou reprezentovány v GPS souřadnicích. GPS model zahrnuje letadla, jejich plány, pozice a tvary sektorů, bezletové zóny a FIXy. Obrázek 2.3 ukazuje vizualizaci simulované reality. Barevné kvádry

představují letové plány jednotlivých letadel, červené tubusy jsou bezletové zóny[1]. FIXy samotné viditelné nejsou, jsou však zpravidla v kloubech letových plánů, v místech kde se mění směr letu.

GPS model funguje jako směrodatný základ zbytku simulace a je používán od první verze AgentFly.



Obrázek 2.3: Detail simulované reality s vykreslenými letovými plány.[3]

Nejdůležitější kámen počítačové simulace je fyzikální stroj (Physical Engine), zajišťující věrohodnou podobu simulace s realitou. Fyzikální stroj AgentFLy zajišťuje správný výpočet rychlosti, spotřeby paliva a výsledků činností, ke kterým dochází uvnitř systému. Jelikož fyzikální model letícího letadla je složitý systém diferenciálních rovnic, nelze se vyhnout numerickému řešení. Jak je známo, chyba numerického řešení vyplývá z metody a délky kroku. AgentFly může najednou pracovat s tisíci letadly a řešit jejich pohyb pomocí numerických metod řešících dynamické rovnice jejich letu by bylo pomalé. A protože letadla zůstávají většinou během letu stejná a provádějí podobné manévry, je rozumné použít pro výpočet přepočítané tabulky.

Databáze BADA[4] (Base of Aircraft Database) obsahuje přepočítaná a naměřená data pro většinu typů letadel. BADA byla vytvořena a je spravována společností Eurocontrol. Umožňuje rychlejší a přesnější výpočty letových drah, nežli by bylo možné pomocí AD-HOC výpočtů skrze diferenciální rovnice. Další použití je při on-fly predikci letového provozu a analýza konstrukčních designů při výrobě letadel.

Bada obsahuje:

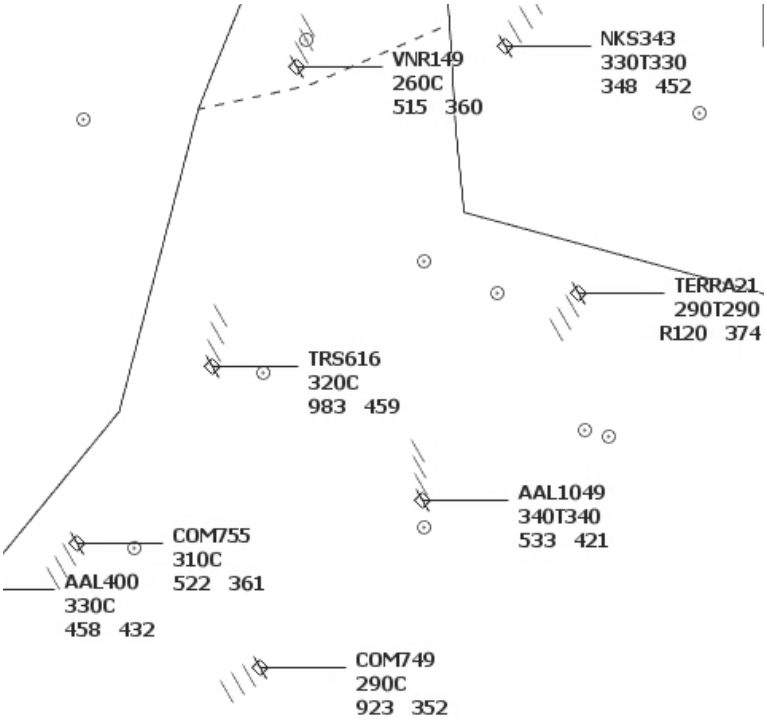
- Polynomiální výkonnostní popisy letadel umožnující výpočty.
- Data ke konkrétním letadlům, koeficienty,vlastnosti do výpočtů.

[1] Oblast, nad kterou se nesmí nacházet žádné letadlo.

Bada pokrývá 99.99% používaných typů letadel v civilní přepravě v Evropě a je základ výpočtů uvnitř fyzikálního stroje AgentFly, i když je schopen fungovat i bez BADA dat. BADA je v dnešní době nepřesnější databáze, kterou lze pro tyto účely použít.

2.2.2 Model řídícího letového provozu

Tento model byl vytvořen pro potřeby FFA2[9] a reprezentuje radarového řídícího v letovém středisku a jeho nástroje. Na obrázku 2.4 je vidět vzhled obrazovky radarového řídícího v obrácených barvách. Tento obrázek byl pořízen na systému AgentFly, který se snaží imitovat vzhled obrazovky na opravdovém letovém středisku.



Obrázek 2.4: Detail obrazovky řídícího letového provozu v převrácených barvách.

Každý radarový řídící letového provozu (dále jen 'řídící') je reprezentován jedním, moduly rozšiřitelným, agentem. Agentův interface se skládá z vizuálního vjemu obrazovky před ním, vstupem ze sektorového rádia, datovým spojením s okolními sektory a z kontaktu s dalšími operátory na stejném středisku (například Point-Out)[10]. Výstupními operacemi agenta je klávesnice a rádio. Agent je implementován jako smyčka lidského vnímání běžící v reálném čase.Takzvaný VCAP [9] model simuluje člověka pomocí čtyř zdrojů. Zdroje jsou vizuální, kognitivní, sluchový a psychomotorický. Akce, které agent koná, využívají tyto zdroje, každá akce vyžaduje jiné a jeden zdroj může najednou být používán pouze jednou akcí. Několik akcí tvoří tzv. aktivitu nebo činnost a každá činnost, kterou může agent provádět, je reprezentována pomocí jednoho modulu.

Propojení mezi Simulovanou realitou a Modelem řídícího letového provozu je rádio a zpětné zobrazení simulované reality do dvoudimenzionálního pohledu na přehledovém radaru řídícího. Rádio slouží ke komunikaci s agenty, kteří ovládají letadla v 1. GPS modelu. Zpětné zobrazení představuje výstup z radaru.

V základu agent zajišťuje událostní rozhraní pro moduly, které mezi sebou komunikují skrze toto rozhraní. Pro jakoukoli akci na agentově vstupu existuje ID události, které je společně detailními informacemi distribuováno modulům schopných kromě iniciování jiných událostí pro mezi-modulovou komunikaci zacházet se všemi výstupy agenta.

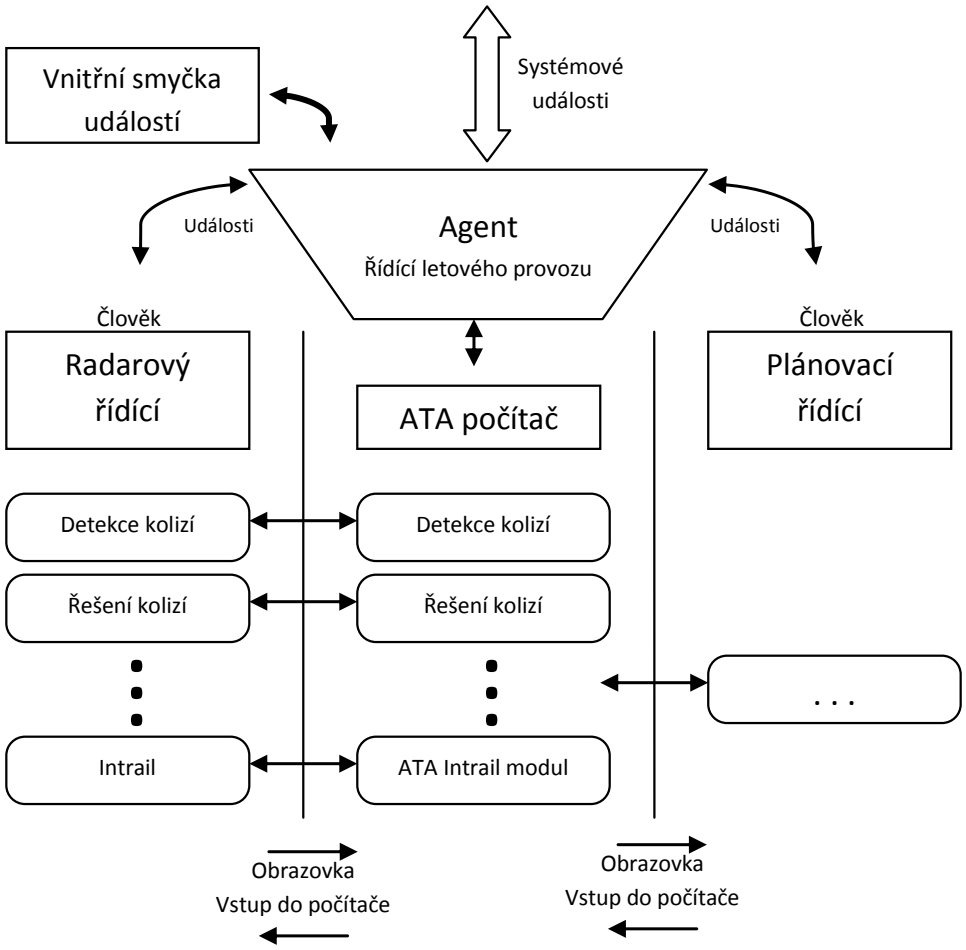
Na obrázku 2.5 je vidět detailní relace uvnitř modulového agenta. Agent samotný tvoří pouze mezivrstvu mezi simulací a moduly. Samotná funkčnost je uvnitř modulů. Jeden agent zaobstarává nyní činnost jednoho člověka, ale jak je vidět na tomto obrázku je založené rozhraní pro přidání druhého řídícího. Tito dva dispečeři zajišťují funkčnost jednoho sektoru. Jeden agent vždy zaopatřuje jeden sektor. Uvnitř agenta existuje vnitřní smyčka událostí, která slouží k událostní interakci mezi moduly uvnitř agenta. Nejdůležitější součást agenta je ATA, nebo ATA počítač. Do této skupiny patří všechny moduly pojmenované s předponou Ata*.

ATA je simulační ekvivalent ERAM(En Route Automation Modernization) [11] [12] počítače sloužícího ke správě sektoru. Tento počítač je nejdůležitějším nástrojem řídícího letového provozu, umožňuje mezi sektorovou komunikaci, spravuje data o letech, umožňuje vizualizaci informací o letech, zpracovává data z radaru a dalších lokalizačních zařízení, zobrazuje přehledně výstup s radaru s doplňujícími informacemi, nabízí řídícímu řadu užitečných nástrojů a je také plným záložním systémem v případě rozličných selhání systému.

2.2.3 Event-based agentní simulace

V simulacích založených na událostech probíhá celé dění jako chronologická sekvence událostí. Každá událost - event nastane v daný simulační čas a změní stav systému. Všichni reaktanti jsou navíc v agentní simulaci agenti.Na systému záleží, které události mohou zaznamenat a na agentovi záleží, na které události bude reagovat. A při reakci agenta agent, jak tomu u událostních simulacích bývá, změní svůj stav a/anebo naplánuje pro systém další událost. Událost musí být samozřejmě naplánována na pozdější čas. Uměle zaváděné události bývá systémem generovaná počáteční událost při vytvoření pro každého agenta a tik simulačního času. Řetězením těchto akcí a reakcí vzniká celá simulace. Event-based simulace jsou používány v situacích, kdy přesné matematické vyčíslení není možné. A oproti vícevláknovým simulacím reálného času jsou daleko úspornější jelikož se zachovává nezávislost jednotlivých agentů i při běhu celého systému na jediném vlákně, právě kvůli chronologickému spouštění event v závislosti na jejich naplánovaném čase.

Jedna událost na úrovni kódu představuje čas, kdy má být spuštěna, identifikátor, který definuje charakter události, jestli se jednalo agentem spuštěnou událost nebo časový tik, a doplňující data, v kterých může být zaznamenáno cokoliv v závislosti na podstatě události, tedy v závislosti na identifikátoru. Identifikátor musí být jeden z předem v kódu definovaných



Obrázek 2.5: Architektura modulové implementace řídícího agenta.

identifikátorů. V systému musí být unikátní identifikátor pro každou možnou událost, která může nastat. Poslední charakter události, který však není v ní přímo uložen, je kde se událost propaguje. V systému může existovat více "událostních kanálů"a když vyvstala událost v jednom, nemusí nutně být zachytitelná ve druhém.

Další složkou událostní simulace je generování pseudonáhodných čísel, která dotváří dojem neurčitosti. Pseudonáhodný generátor může, a v případě AgentFly tomu tak je, být stejný pro stejnou simulaci. Tím pádem při dvou běžích identických simulacích skončí simulace stejně, jsou deterministické.

Omezené kognitivní schopnosti lidského operátora, přenosová kapacita rádia přenášející hlas a klávesnicový vstup do počítače dál prodlužuje dobu každé reakce. Z toho důvodu má každá akce v závislosti na pravděpodobnostním rozdělením trvání přidělený čas. Hodnoty těchto rozdělení jsou nastaveny na základě měřen na skutečných řídících letového provozu v FAA [2]. Akce také používají některé ze čtyř psychicko-fyzických zdrojů VCAP modelu. Agent může například pozorovat obrazovku a zároveň mluvit do rádia nebo hledat řešení

[2]Tyto časy byly poskytnuty Americkým úřadem pro letectví ve Washingtonu. Za pomoci dodaných sta-

kolize ve stejnou dobu když poslouchá rádio, ale nemůže například "přemýšlet"o dvou věcech najednou. Tyto čtyři zdroje jsou zrakové, sluchové, rozhodovací a psychomotorické. Akce, které by využívaly stejné zdroje, musí běžet sekvenčně. Pokud využívají různé zdroje, mohou být prováděny paralelně. Například řídící může najednou sledovat obrazovku a u toho mluvit do rádia. Kdykoliv začne provádět akci patřící do jedné ze dvou zmíněných skupin nastaví tím čas, kdy se nejdříve může provádět další akce z této skupiny a čas nastaví přesně jako součet aktuálního času s dobou trvání prováděné akce.

tistik a výpovědí dispečerů s mnohaletou praxí bylo ve spolupráci s FAA rigorózně rozhodnuto o hodnotách, nebo spíše rozděleních časů pro jednotlivé akce.

Kapitola 3

Analýza a návrh řešení

AgentFly je psán v programovacím jazyce Java a jako grafické knihovny používá Javovskou nástavbu pro OpenGl JOGL [13]. Zvolený jazyk je tedy Java.

V Agenfly bude v rámci MIT implementačně zasaženo do pilot-agenta který ovládá letadla a do operátor-agenta. Operátor bude rozšířen o dva funkční moduly a pro potřeby funkčního MIT bude třeba naimplementovat několik funkcí.

Analýza a návrh implementace (včetně diskuse různých alternativ a volby implementačního prostředí).

Jak bylo řečeno v předchozí kapitole tak MIT pro jeden sektor znamená vypouštět ze sektoru dotčená letadla s požadovanými rozestupy dle příslušného MIT pravidla. Rozestupy měříme na výstupu ze sektoru.

Pokud definujeme σ jako maximální množství letadel, které je cílové letiště nebo letištní dráha schopna zvládnout za jednotku času a σ_n jako množství letadel letící na zmíněné letiště nebo letištní dráhu z n-tého sektoru za jednotku času pak:

$$\sigma < \sum_{n=0}^{k} \sigma_n$$

Rozstupy mezi letadly těmito letadly letícími z n-tého sektoru musí být:

$$d_n = \frac{v_n}{\sigma_n}$$

Tato vzdálenost je společně s dalšími informacemi o MIT předána n-tému sektoru, který ji po té musí implementovat a dodržovat.

3.1 Metodika

První letadlo samo podmínky MITu neporuší. Každé další letadlo je potřeba pozdržet natolik, aby se požadované rozstupy dodržely. Zpožďovat lze letadla dvěma způsoby, změnou rychlosti nebo změnou trasy. Změna rychlosti je efektivní z hlediska spotřeby paliva, ale zpoždění,které

je možno vytvořit pomocí změny rychlosti,je malé a přímo úměrné délce trasy letu v sektoru. Pokud změna rychlosti nestačí, je potřeba měnit trasu. Nyní je potřeba udělat odbočku ke komunikačnímu protokolu na sektorovém rádiu a jakým způsobem může operátor upravovat trasu letadla.

Letový plán je ve skutečnosti lomená čára, definovaná jako spojnice mezi FIXy (bod definovaný jako GPS souřadnice), které jsou většinou reprezentací radio-majáků, sloužících k automatické strojové navigaci letadla. Posloupnost FIXů pro každý let je dopředu známa v systému a operátor si jí muže nechat graficky zobrazit. Operátor může letový plán měnit buď ve vertikálním nebo v horizontálním směru.

Vertikální změny letového plánu změní výšku letadla. Pokud letadlo letí podle svého původního letového plánu je zadání vertikální změny snadné a funguje na principu výškových omezení. Řídící sdělí letadlu výšku a bod na letovém plánu. Letadlo musí od chvíle zadání do chvíle průletu bodem změnit svojí výšku tak, aby v definovaném bodě se nacházelo na přikázané letové hladině. Body na letovém plánu se zadávají vzdáleností relativně k některému FIXu na dráze letadla. Řídící musí počítat s tím, že v časovém intervalu od zadání vertikální změny do momentu, kdy letadlo prolétá referenčním bodem, se může letadlo nacházet v libovolné letové hladině mezi jeho původní přikázanou výškou.

Horizontální změna letového plánu je neurčitější než vertikální, jelikož zahrnuje více proměnných, které mohou její průběh ovlivnit. V rádiové komunikaci může operátor přikázat letadlu udělat vybočení na jiný kurz, respektive vybočení z kurzu o zadaný úhel. Tento příkaz může operátor opakovat neomezeně mnohokrát. Druhým příkazem je příkaz pro návrat k původnímu plánu, a to buď k příštímu FIXu, který ještě nebyl proletěn, nebo pokud je explicitně řečeno tak k libovolnému dalšímu FIXu v letovém plánu.

Pomocí těchto manévrů jsme schopni splnit MIT.

Metodika bude následující:

1. Po vletu letadla do sektoru odhadneme čas, kdy opustí sektor.
2. Porovnáme čas s letadlem, které poslední vyletí ze sektoru dříve než nové letadlo.
3. Odhadneme rozstup mezi letadly ve chvíli kdy druhé letadlo opouští sektor.
4. Pokud je rozstup menší než požadovaný, rozstupem musíme upravit trasu nového letadla.
5. Postup opakujeme pro každé další letadlo, které vylétá po právě zpracovaném letadle.

3.2 R-Side operátor

Popis architektury agenta řídícího sektor je v kapitole 2.2.3 a znázorněno na obrázku 2.5. R-Side operátor neboli radarový řídící je hlavní ze dvojice radarový a plánovací řídící. Povinnosti radarového řídícího jsou přijímání a předávání kontroly nad letadly[1], který je potřeba

[1] Takzvaný Incoming Handoff nebo OutgoingHandoff

komunikovat s řídícími z okolních sektorů. Pokud je v letovém plánu letadla zaznamenáno stoupání či klesání musí pokud možno zajistit splnění letového plánu. Detekovat a řešit kolize ve svém sektoru je však nejdůležitější a nevytěžující činnost rádiového řídícího. Zajišťovat plnění meteringu a Miles In Trail restrikcí. Pro každou z funkcí je v AgentFly napsán modul, který je za ni zodpovědný.

R-Side operátor také komunikuje a využívá pomoci D-Side[2] operátora, který provádí dlouhodobé strategické analýzy a operace ve větším časovém i prostorovém měřítku.

Dalšími povinnostmi a privilegiem R-Side řídícího je komunikace po sektorovém rádiu, kontrola SOP pravidel (Standard operating procedure) **??**, aplikace Point-Outu a operace s radarovou obrazovkou a klávesnicí počítače ERAM [12][3].

3.2.1 Intrail moduly

Funkcionalita MIT je v systému rozdělena systematicky do dvou modulů. Podle architektury agenta z obrázku 2.5 jsou funkce spojené s myšlením a činností řídícího prováděny uvnitř modulu na `Intrail` na straně `radarového řídícího`. Řídící používá funkci uvnitř modulu `ATAIntrail`, který reprezentuje potřebnou funkcionalitu počítače pro problém MITu.

Pro vytvoření modulů a přiřazení k agentovi je potřeba zavést dvě nové řádky do konfigurace sektoru. Uvnitř konfiguračního XML souboru popisujícího sektor se mimo jiné nachází párové tagy `<InitAtaModule/>` a `<InitRSideModule/>`, oba uzavřeny do párového tagu `<SectorConfiguration/>`. Tyto tagy definují, které moduly budou načteny. V této sekci bude mezi patřičné tagy přidán řádek:

`<Module ClassName="atc.faa.atm.enroute.ata.EomAtaIntrail"/>`, který v sekci ATA vytvoří Intrail modul. Pro sekci radarového řídícího to je

`<Module ClassName="atc.faa.atm.enroute.rside.EomRSideIntrail"/>`.

Samotná funkčnost MIT bude vyžadovat popis, jakým způsobem se má MIT aplikovat. Za tímto účelem budou vytvořeny XML soubory, pro každý sektor jeden, ve kterých budou popsány příznaky letadel patřících do daného MIT, aplikované rozstupy a další nutné informace.

3.2.2 Kooperace s dalšími moduly

Pro úpravu letového plánu letadla potřebujeme komunikační kanál, kterým je v letovém provozu rádio. Každý sektor má vlastní kanál a při přeletu letadla mezi sektory letadlo přeladí na frekvenci nového sektoru. Komunikační kanál rádia je poloviční duplex[4] a vysílání je časově náročné. Přesně takto je rádio v AgentFly modelováno.

MIT modulu rozhodneme o podobě letového plánu letadla a ten uložíme do instance třídy SgPlan. SgPlan představuje operátorův záměr, jak chce vést letadlo sektorem. Základní tvar SgPlanu je let letadla po FIXech přesně jak bylo definováno předem - provedení tohoto

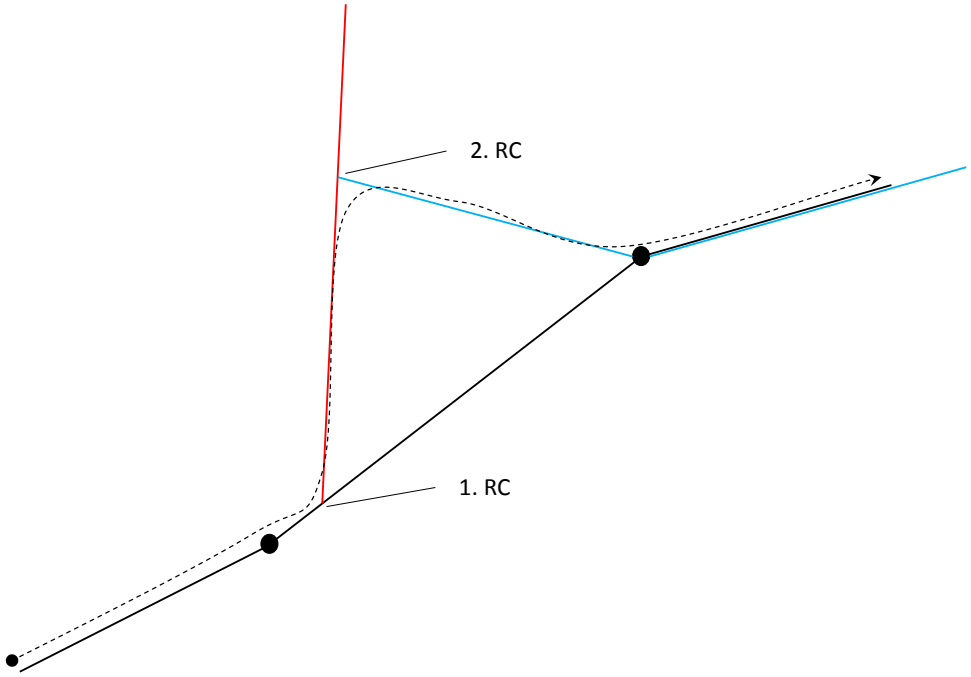
[2] Na obrázku 2.5 je D-Side operátor plánovací řídící.

[3] ERAM je detailněji popsán v kapitole 2.2.2.

[4] Half-Duplex : Při poloviční duplexním přenosu dat může najednou vysílat pouze jeden subjekt. Vysílání mohou slyšet všechny subjekty na stejné frekvenci.

plánu nevyžaduje použití rádia. Ostatní SgPlany vyžadují v pravý moment příkaz přes rádio. SgPlany jsou sdíleně uloženy v paměti operátora. Stačí zde SgPlan upravit a o ostatní se postará modul ovládající rádio.

Obrázek 3.1 ukazuje znalost pilot agenta v jednotlivých částech přikazované zatáčky. Pokud řídící naplánoval letadlu jiný nežli předem domluvený plán, zná jeho podobu zatím jenom on. Při normálním běhu nakonec letadlo poletí podle jím vymyšleného plánu. Na obrázku přerušovaná čára značí *SgPlan*, tedy plán pro letadlo, který je pod správou řídícího. Černá trasa značí původní letový plán, známý dlouho dopředu pilotovi i řídícímu. Pilot si myslí, že letí po tomto plánu až do bodu 1.*RC* (první rádiová komunikace), kdy je mu sdělen nový plán. Mezi body 1.*RC* a 2.*RC* se pilot řídí podle červeného plánu, který se dozvěděl během první radiokomunikace. V momentě když letadlo přelétá nad 2.*RC* řídící nahlásí letadlu novou trasu a to návrat k původnímu plánu. Letadlo pokračuje po modré trase.



Obrázek 3.1: Rozdíl mezi znalostmi řídícího agenta a pilot agenta. RC značí rádiovou komunikaci mezi řídícím a pilotem.

MIT moduly musí být naprogramované tak, aby nekolidovaly s ostatními funkcemi operátora. A to jest především modul vyhledávající a řešící kolize.

Tento modul je v aktuální verzi implementován tak, že pří každém Incoming handoffu (IH) [5] se aktivuje a porovná letový plán příchozího letadla s letovými plány ostatních letadel

[5]Proces, kdy sousední sektor oznámí blížení se nového letadla do aktuálního sektoru. Soudední sektor nemůže předat letadlo pokud řídící aktuálního sektoru nepotvrdí, že je schopen letadlo přijmout. Incoming hand-off = příchozí předání.

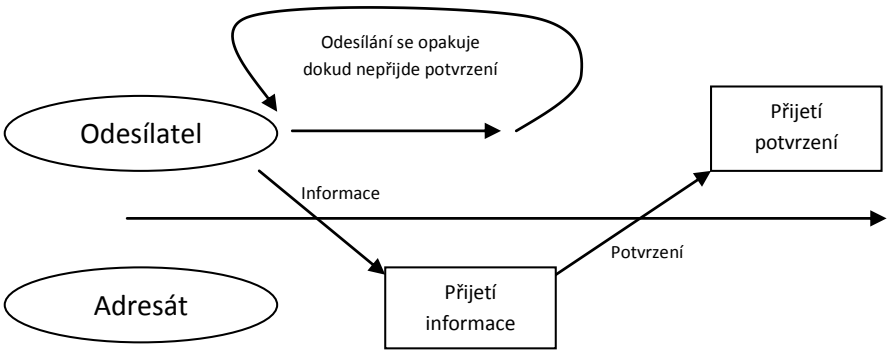
v sektoru a provede takové akce aby ke kolizi nedocházelo, a stav letadel uvnitř sektoru byl opět bezkolizní po příštích několik minut simulace. S předpokládaným bezkolizním stavem letadel po několik příštích minut simulace před tímto procesem se indukčně zajistí, že nedojde ke kolizi pokud se po zbytek času podaří navádět letadla podle jejich upravených letových plánů. Detekce kolize se kromě situace po IH provádí periodicky.

Problém společného fungování MIT modulu a CD-CR modulů (Collision Detection - Collision Resolution) je, že oba mění letové plány s jinou heuristikou. Pokud bychom řešili tento problém užší spoluprací, porušili bychom myšlenku modulového přístupu. Řešení tedy bude sekvenční použití obou modulů. Pokud první se provede řešení kolizí a následně MIT, může MIT vytvořit nové neřešené kolize. MIT by sice mohl po úpravě letového plánu znovu volat CD-CR, ale dle mého názoru toto řešení povede k cyklickému volání MIT a CD-CR a bude výpočetně náročné.

I když je výše uvedené řešení korektnější z hlediska úplnosti tak volím druhou možnost. MIT se provede první, provede potřebné úpravy a nový stav přenechá CD-CR. Problém který zde nastává je když by CR rozhodl upravit letový plán letadlu, které patří do MIT. Dostačují mnou navrhované řešení je přiřazení priority letovým plánům podle toho, jestli například náleží do MIT. Plány náležící do MIT budou označené jako prioritní a k jejich úpravě dojde až tehdy když neexistuje jiné řešení, což je v pořádku, jelikož prevence kolizí má větší prioritu nežli MIT.

MITu tedy přidělíme vyšší prioritu nežli hledaní kolizí (CD).

3.2.3 Rádio



Obrázek 3.2: Předávání informací na sektorovém rádiu.

Jak již bylo v předchozí kapitole řečeno, na sektorovém rádiu při předávání zprávy zatní nejdříve informace a následně odpověď adresáta. Pokud nejsou problémy s komunikací, slyšitelností nebo pokud nemluvil adresát ve stejnou chvíli jako odesílatel, na předání jedné informace budou dva vstupy do rádia. Zde popíši jedno předání informace pomocí rádia.

Celá popisovaná činnost je zobrazena na obrázku 3.2. Komunikace je iniciována odesílatelem. Kvůli podstatě rádia musí odesílatel počkat na chvíli, kdy nikdo jiný nevysílá. Po

chvilce rádiového ticha začíná vysílat. Nejdříve uvede adresáta, následně předá informaci a po dokončení přenosu uvolní rádio. Nyní odesílatel vyčkává na odpověď adresáta. Pokud adresát neodpovídá, odesílatel akci opakuje. V případě, že některý z adresátů se na frekvenci vůbec nehlásí řeší se situace jinak v závislosti na její podstatě.

Odesílatel opakuje vysílání informace do té doby, dokud mu adresát neodpoví formou: jméno adresáta, opakování informace. Tímto je vysílaní ukončeno. Jediná další možnost je, že si adresát požádá o zopakování zprávy. V tom případě se celá operace opakuje.

Na konci předání adresát zná předanou informaci. Zda odesílatel ví, že adresát informaci dostal může adresát vyvozovat pouze z toho, že podezřele dlouho odesílatel nežádá o potvrzení.

Kapitola 4

Realizace

Následující kapitola popisuje podobu a funkčnost MIT funkcionality v první verzi.

4.1 Intrail moduly

Nově založené moduly jsou EomAtaIntrail a EomRSideIntrail.

Myšlenka rozdělení MIT funkčnosti do dvou modulů ctí rozdělení všech modulů na činnosti svázané s přístrojovou technikou a na nezávislé na nástrojích. Po vytvoření obou modulů si RSideIntrail najde v tabulce modulů AtaIntrail a uchová si na něj odkaz a volá jeho funkce přímo. V původní architektuře byla komunikace ATA a RSide modulů zprostředkována pomocí eventů, ale nyní je použito přímého volání pro zrychlení komunikace.

EomRSideIntrail je registrovaný jako posluchač agentových eventů. Event, na který se čeká kvůli MITu je `HHO_RQ_ACC_KBD`, která představuje příchozí handoff od sousedního sektoru. Tento event proběhne vždy jednou pro každé příchozí letadlo jakmile je z vedlejšího sektoru iniciativa ho předat do aktuálního sektoru. Pokud existuje modul EomAtaIntrail, k čemuž by mohlo dojít nesprávnou konfigurací nebo jinou chybou v systému, tak je ID letadla předáno modulu EomAtaIntrail funkcí `processIntrail(AirplaneId id)`. Třída *AirplaneId* obsahuje název letadla a pomocí *AirplaneId* , pomocí kterého jsou indexovány uložené letové plány a informace o letu.

Ostatní použité eventy:

`AIRPLANE_LEFT_THE_SECTOR_NOTIFICATION` tento event vyvstane tehdy, když letadlo opustí operátorův sektor, ať horizontálním nebo vertikálním handoffem. Neplatí to tedy v situaci, kdy je letadlo odstraněno ze simulace násilně. Tento typ opuštění simulace a tím i sektoru spustí event `FORCED_AIRPLANE_REMOVAL`.

Při odouch těchto eventech musí být letadlo odstraněno z vnitřních MIT modulů. Při `AIRPLANE_LEFT_THE_SECTOR_NOTIFICATION` navíc je v debug-módu pořízena statistika o vzdálenosti letadel sloužící k zpětné kontrole efektivity a přesnosti MIT. S touto funkcí se setkáme v následující kapitole 5. Experimenty.

4.2 Používané funkce

Zde je uveden stručný popis nových a některých používaných existujících funkcí na straně jak řídícího agenta, tak letadlového agenta.

4.2.1 Nové funkce - Intrail moduly

- `processIntrail` primární metoda `EomAtaIntrail` modulu volaná z `EomRSideIntrail` modulu. Parametrem je identifikátor nového letadla, které se blíží k sektoru. Po ověření, jestli patří do některých z aplikovaných MITů, je dále zpracováno uvnitř třídy `Intrail`.
- `belongsToIntrail` je metoda třídy `Intrail`, která provádí zmíněné ověření příslušnosti letadla do MITu. O příslušnosti rozhoduje shoda několika posledních FIXů letového plánu s některých cílových FIXů definovaných v konfiguraci MITu. Mezi letadly a MITy platí vztah n:1, letadlo náleží maximálně do jednoho MITu, ale k MITu bývá přiřazeno více letadel.
- `addPlaneToIntrailApproach` Po ověření příslušnosti letadla do MITu je zavolána tato metoda, která obstará správné zařazení mezi ostatní letadla a provede nutné úpravy v letových plánech pomocí funkce `delay`.
- `delay` Ověří nutnost opoždění aktuálního letadla a spočte minimální čas, o který se let musí opozdit. Nalezne vhodné, dostatečně dlouhé segmenty[1] v aktuálním sektoru, na které bude spouštět funkci `delayFor` s časem, o který se musí pozdržet, jako argument.
- `delayFor` pro daný let na daném sektoru hledá vhodné zpožďovací parametry. Vrací instanci třídy `DelayResult`, do které buď uloží informaci o nezdaru nebo výsledek výpočtu.
- `processLeavingSector` se stará o úklid po zbytku funkcí. Když letadlo opustí sektor, je již nedůležité pro další výpočty, a tak je vyřazeno ze seznamu a v debug módu je spočten rozestup s dalším letadlem pro zpětnou vazbu efektivity modulu.

4.2.2 Nové funkce - Pilot agent

Agent řídící jedno letadlo v simulaci. Většina funkcí patří pod třídu GpsFlightPlanWrapper-Polyline(GFPW), která zaobaluje informace o letu, letovém plánu a vypočtené trajektorii letu pro vlastnícího agenta.

- `setHorizontalTurn` je funkce uvnitř GFPW sloužící k zavedení odbočení z aktuálního kurzu. Parametry obsahují směr(pravý, levý) a úhel zatočení. Obrat je do plánu zaznamenán za aktuální bod v čase plus rozhodovací čas. Datově se provede záloho zbytku plánu který nebyl proveden a naplánuje se let na daný kurz po příštích 300 kilometrů. Po zavolání této funkce se očekává brzké volání funkce `setHorizontalReturn`, jelikož pokud letadlo opustí sektor v tomto stavu, bude zpravidla násilně odstraněno ze simulace.

[1] Segment je část letového plánu mezi dvěma FIXy, spojnice.

- `setHorizontalReturn` další funkce GFPW sloužící k návratu letadla k původnímu plánu. Zbytek původního plánu byl uložen funkcí `setHorizontalTurn` do proměnné `backupWaypoint` a bude je použit jako zbytek plánu. Spojnice mezi aktuální pozicí a zálohovaným plánem je spojena několika body, jejichž výpočet bude popsán později. Uvnitř proměnné `backupWaypoint` je vždy úplně původní plán, protože při opakovaném volání funkce `setHorizontalTurn` se znovu záloha nepřepisuje.

- `createTurn` slouží k výpočtu průchozích bodu v prostoru při průletu zatáčkou. Tato funkce se používá v obou předchozích funkcích právě k výpočtu podoby zatáčky. Pro tuto funkci jsou důležité parametry úhel, směr, rádius obratu a původní situace, ve které se nachází letadlo. (Pokryto pomocnou třídou `Situation`)

- Jelikož let na konkrétní azimut na planetě není přímka ani kružnice, je pro potřeby reprezentace letového plánu,který je sérií jednoduchých křivek, aproximovat takovýto let. K tomu slouží funkce `pointInDirection`, která vrací prostorový bod, který je od bodu předaného jako argument, v zadaném směru a vzdálenosti. Směr a vzdálenost jsou také argumenty této funkce.

4.2.3 Pomocné třídy

- `Situation` je třída zaobalující zjednodušený stav letadla v některý moment. Je nadefinovaná uvnitř GFPW a je použita během hledání správného tvaru zatáčky a letu podle azimutu. Na obrázku 4.2 každý pomocný bod zobrazený modrou barvou byl nalezen pomocí třídy `Situation`. Obsahuje směr, pozici v GPS souřadnicích a rychlost letadla.

- `HorizontalDiversionParams` obsahuje parametry směrové diverze pro pilot agenta. Používá se v simulaci komunikace po rádiu mezi agentem řídícího a pilot agentem.

- `DelayResult` je na straně Intrail modulu pomocná třída obsahující parametry pro pozdržení letadla. Funkce jednotlivých metodik pozdržení jsou dle priorit volány a každé volání vrací tuto třídu, která kromě informace o úspěchu či nezdaru obsahuje informace dostačující pro aplikaci pozdržení. V aktuální verzi je použito pouze diverzní řešení, v budoucnu ale bude přidána minimálně změna rychlosti.

- `Intrail` třída zaobalující jednu definici MIT. Obsahuje příznaky k rozhodnutí o příslušnosti letadla do tohoto MITu a ostatní parametry jako rozstupy. Obsahuje spojový seznam třídy `IntrailPosition`. Všechny důležité metody pro provedení MIT jsou obsaženy právě v této třídě.

- `IntrailPosition` reprezentuje doplňující informace k letadlu, které bylo zařazeno do MIT, tedy prošlo třídou `Intrail`. `IntrailPosition` také tvoří jeden prvek spojového seznamu uvnitř třídy `Intrail`. Obsažené proměnné jsou příští a předchozí `IntrailPosition`, cílový čas opuštění sektoru a postižené letadlo.

4.3 Konfigurace

Jak bylo řečeno, Miles In Trail je v první verzi řešen statickým přidělením pravidel pro každý sektor a pro každý MIT zvlášť. K tomu slouží konfigurační soubory formátu XML, pojmenované intrail[KÓD_SEKTORU].xml. Pro každý sektor existuje maximálně jeden soubor definující všechny aplikované MITy. Zde je příklad jednoho krátkého konfiguračního souboru použitého ve scénáři, který bude použit jako ukázkový v kapitole 5. Experimenty:

```
<?xml version="1.0" encoding="UTF-8"?>
<Intrails xmlns="atc/faa/ontology"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="atc/faa/ontology ../../../
          atc.faa2/src/ontology_src/sectorConfiguration.xsd">

          <Intrail Name="JFK_approach"
                   Destination="JFK"
                   OptimumDistanceNM="13"
                   MinimumDistanceNM="10"/>
</Intrails>
```

Aplikované intraily uzavírá tag *Intrails*. Vnitřní tag *Intrail* se opakuje pro každý MIT v sektoru a obsahuje tyto parametry:

- Name : Slouží pouze k identifikaci konkrétnímu MIT. Později bude sloužit k svazování MIT napříč sektory, k dynamické úpravě MIT pravidel.
- Destination : Obsahuje informace sloužící k přidělení letadel do MIT. V jednom řetězci jsou zakódovány všechny cílové FIXy, které musí letový plán obsahovat, aby byl do tohoto MIT zařazen, oddělené dvojtečkou.Pro uvedený příklad budou do MIT zařazeny všechny lety, jejichž letové plány končí FIXem 'JFK'. Sekvence FIXů se značí pomocí dvou teček : například FIX..FIX.
- OptimumDistanceNM : Obsahuje vzdálenost, která by měla jednotlivá letadla v MITu rozdělovat v ideálním případě, v námořních mílích.
- MinimumDistanceNM : Obsahuje minimální vzdálenost, kterou musí mezi sebou letadla mít, aby se jimi operátor nezabýval. Pokud mají vzdálenost menší, tak bude letový plán druhého letadla upraven. Vzdálenost je v námořních mílích.

V pozdějších verzích budou přidány další parametry jako například maximální zdržení, které je teď definované pouze v kódu.

4.4 Funkce detailně

4.4.1 Funkce EomAtaIntrail.processIntrail

Další použité třídy (Všechny následující třídy jsou privátní třídy modulu EomAtaIntrail) :

- Intrail : Reprezentuje právě jednu definici a paměť o již přijatých letech k jednomu MITu.
- IntrailPosition : Vnitřní třída třídy Intrail. Obsahuje data o jednom do MITu přijatém letu. Mimo to je IntrailPosition také jeden element dvojně propojeného spojového seznamu, ve kterém jsou všechny IntrailPosition uchovány.

Po nahlášení příchozího handoffu a odchycení této události v *EomRSideIntrail* je volána funkce *EomAtaIntrail.processIntrail*(*AirplaneId*). Nejdříve se ověří, zda letadlo vůbec existuje a zjistí se skrze funkce operátora, jaký je jeho letový plán. Pro MIT je důležité několik posledních FIXů v letovém plánu. Názvy posledních FIXů se porovnají s koncovými FIXy každého aktivního MITu a maximálně jednomu MITu se nové letadlo přiřadí. K ověření náležitosti letadla do intrailu slouží funkce *Intrail.BelongsToIntrail*(*List*<FixProjection> *fixes*), která porovná zmiňované FIXy.

Když *BelongsToIntrail* metoda vrátí *true*, ukončíme hledání a letadlo přidáme pouze do tohoto MITu zavoláním jeho metody *AddPlaneToIntrailApproach*(*AirplaneIdid*).

Uvnitř třídy *Intrail* jsou jednotlivé zahrnuté lety uloženy ve dvojitě spojovaném Linked-Listu [2] a jsou seřazeny podle pořadí, v jakém budou opouštět sektor. Jeden prvek kolekce reprezentuje třída *IntrailPosition*. Proměnné *IntrailPosition*(IP) jsou :

```
public AirplaneId airplaneId; // ID letadla.
public IntrailPosition after; //IP letadla, které vyletí ze sektoru
                              //až po aktuálním letadle.

public IntrailPosition before; //IP letadla, které jako poslední vyletí ze
                                  //sektoru dříve než aktuální letadlo.

private boolean delayedAlready = false; //Letový plán letadla již byl upraven
                                     //kvůli MIT.
private double delayedOnTime = Double.NaN; //Pokud letový plán nebyl
                                  //upraven kvůli MIT, obsahuje NaN.
                                  //V opačném případě obsahuje Absolutní čas
                                  //simulace, ve kterém se odhaduje,
                                  //, že letadlo opustí sektor.
```

V takto uloženém seznamu se udržuje přesné pořadí letadel a odhady času pro opuštění sektoru.

[2] Linked list [14], neboli spojový seznam, je seznam prvků uschován tak, že existuje reference na první prvek a reference na každý další prvek je vždy uschována v předchozím. Výhodou je snadné přidání elementů, nevýhodou přístupová doba k n-tému prvku je n. Prvek ve dvojitě spojované spojovém seznamu navíc obsahuje odkaz na předchozí prvek.

4.4.2 Algoritmus

Algoritmus se skládá ze tří kroků :

1. Zařazení letadla do fronty správně podle odhadovaného času opuštění sektoru a stavu ostatních letadel podle konfigurace a modelu řídícího.
2. Ověření potřeby pozdržení nově přidaného letadla a letadla následujícího.
   - Nalézt správné parametry pozdržení, je-li třeba.
3. Je-li třeba, pozdržet ostatní letadla, co přiletí později oproti novému letadlu.
   - Nalézt správné parametry pozdržení.

Druhá část je odstraňování letadel ze seznamu ve dvou situacích : násilného odstranění letadla ze simulace nebo opuštění sektoru letadlem.

1. Zařazení letadla do fronty.

   Počínaje posledním letadlem ve frontě směrem dopředu se postupně porovnává odhadovaný čas opuštění sektoru s novým letadlem. Nové letadlo se zařadí za aktuální letadlo buďto jestliže aktuální letadlo již vylétá ze sektoru dříve nebo když aktuální letadlo již bylo pozdrženo.

2. Ověření rozstupů.

   V druhé fázi se pořínaje novým letadlem ověřuje, jestli nechává letadlu před sebou dostatečný náskok. V případě potřeby se nové letadlo pozdrží. Stejné ověření a pozdržení se provede na dalším letadle v pořadí.

3. Ostatní letadla.

   Pro každé další letadlo platí. Pokud bylo pozdrženo letadlo přede mnou, provedu ověření rozstupu. Pokud je rozstup menší než minimální, pozdržím letadlo a pokračuji dalším letadlem v pořadí.

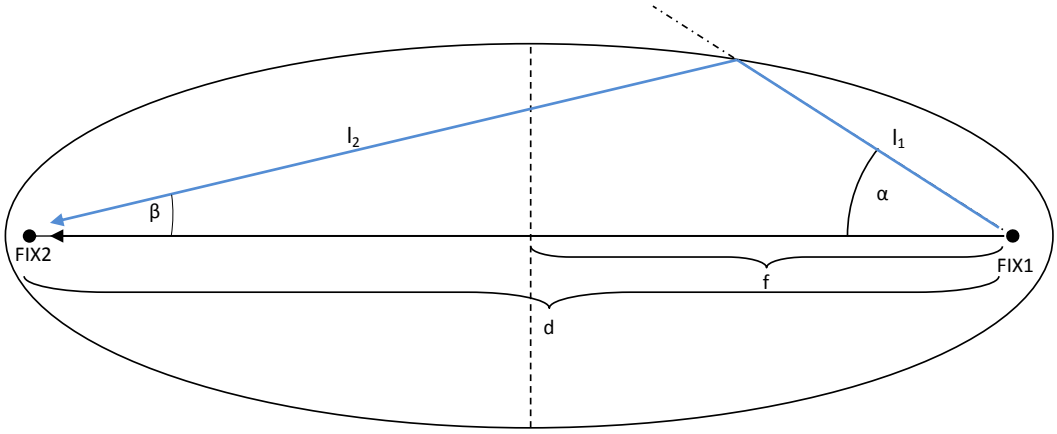
4.4.3 Hledání parametrů pozdržení letadla

Vstupem jsou dva letové plány dvou letadel, která budou ze sektoru vylétat příliš blízko u sebe. První letadlo, které vyletí dříve, necháme beze změny plánu. Výchozí letový plán je zpravidla nejkratší možný a zrychlení není implementováno. Řešením je pozdržení prvního letadla. Důležitá hodnota pro další výpočet spojená s prvním letadlem je časový rozdíl opuštění sektoru mezi oběma letadly, kterou označíme jako τ. Z konfigurace také víme dva důležité údaje. δ_{min} představující minimální vzdálenost mezi letadly (kvůli tomu, že $\delta_{min} > \tau$ jsme se dostali k pozdržování letadla) a δ_{opt} jako optimální rozstup letadel. δ_{opt} použijeme jako cílovou vzdálenost mezi letadly. Cílový časový rozstup vypočteme z rychlosti v pomocí:

$$\tau_{opt} = \frac{\delta_{opt}}{v}$$

Víme o kolik je potřeba pozdržet druhý let. Provedeme to prodloužením trasy a to jediným dostupným implementovaným způsobem : kombinací operátorových povelů vybočení na kurz a příkazem návratu k původnímu letovému plánu. Řídící smí ovlivnit trasu letadla pouze v případě, že se letadlo pohybuje jeho sektorem a zároveň nad ním má kontrolu. To nemusí platit na začátku, když sice letadlo už je v kontaktu, ale stále se pohybuje v cizím sektoru. Stejně tak je řídící povinen předat letadlo řídícímu v dalším sektoru s předstihem před opuštěním sektoru letadlem. Dále jsme omezeni na pouhé dva příkazy, které se aplikací musejí vejít mezi dva FIXy. Letadlo je podle letového plánu zadaného před vzletem povinno proletět v blízkosti všech zadaných radio-majáků, FIXů, a tím se správně držet na středisky očekávané letové dráze. Pokud se na této dráze cokoliv změní, je to pouze po dohodě se všemi zúčastněnými, minimálně tedy sektorem. Příkaz `Vybočení o úhel` nutí letadlo dle povelu změnit letový kurz o zadaný počet stupňů do zadané strany. Příkaz `Návrat k letovému plánu` říká letadlu, aby pokračovalo na předchozí FIX, který ještě neproletělo. Jestliže se chceme vyhnout kaskadérským obratům v našem sektoru, které my vytvořila situace, kdy letadlo odletělo daleko za svůj další FIX a snaží se obrátit a vrátit se, tak provedeme příkaz vybočení a návrat v takovém sledu, aby finální trasa byla bez ostřejších zatáček. Druhou možností je přepsat letový plán a přeskočit některé FIXy a poslat letadlo na další FIX v pořadí. Tenhle způsob většinou zahrnuje spolupráci dalšího sektoru, je složitější, zatím neimplementovaný a nad rámec této práce. Budeme se tedy držet pouze trojúhelníkovitých úhybů mezi pouze dvěma FIXy.

Na obrázku 4.1 je naznačená trasa letadla mezi FIX1 a FIX2. Přímá spojnice představuje původní plán, původní předpokládanou trasu letadla. Modrá lomená čára představuje novou trasu. Podle trojúhelníkové nerovnosti bude nová trasa vždy delší. Původní délku trasy mezi FIXy, mezi kterými provedeme manévr, označíme jako d. Délka nové trasy bude $l = l_1 + l_2$, kde $l_1 a l_2$ jsou délky dvou ramen trasy od prvního FIXu a směrem ke druhému FIXu (na obrázku 4.1 modrá čára).



Obrázek 4.1: Nákres výpočtu změny trasy

Nová trasa musí být delší o:

$$\partial l = v.(\tau_{opt} - \tau)$$
$$d + \partial l = l_1 + l_2$$

Společně s původní trasou o délce d tvoří $l_1 a l_2$ trojúhelník. Hledáme množinu bodů, pro které platí, že:

$$|X, FIX1| + |X, FIX2| = d + \partial l$$

Takováto množina bodů je podle definice elipsa s ohnisky ležících na obou FIXech. Ve volbě diverzního úhlu α máme volnost a kvůli snadnému dorozumění a aplikovatelnosti pro letadlo volíme celý úhel, dělitelný deseti nebo alespoň pěti. Velikost úhlu α bude rádiem předáván letadlu ve chvíli, kdy se letadlo bude pohybovat nad bodem FIX1.

Interface `SgPlanu` potřebuje úhel α a β. Jakmile jsme vhodně zvolili úhel α, úhel β dopočítáme přes polární vzorec pro elipsu a sinovou větu. Polární rovnice pro elipsu, relativní k ohnisku:

$$r(\phi) = \frac{a.(1-e^2)}{1-cos(\phi)}$$

Kde a je hlavní poloosa a e excentricita. Následující vzorec ukazuje výpočet excentricity v naší situaci:

$$e = \frac{f}{a} = \frac{d/2}{(d+\partial l)/2} = \frac{d}{d+\partial l}$$

,kde f je ohnisková vzdálenost,tedy polovina vzdálenosti FIXů, a a hlavní poloosa, polovina cílové vzdálenosti $d + \partial l$. Následující vzorec vychází z rovnic elipsy aplikovaných pro popisovanou situaci:

$$l_1 = \frac{\frac{d+\partial l}{2}.(1-(\frac{d}{d+\partial l})^2)}{1-cos(\alpha)}$$

Máme délku všech stran a dopočítat úhel β je snadné. Sinová věta:

$$\frac{sin(\alpha)}{a} = \frac{sin(\beta)}{b} \;..>\; \frac{sin(\alpha)}{l_2} = \frac{sin(\beta)}{l_1} \;..>\; sin(\beta) = \frac{l_2.sin(\alpha)}{l_2} \;..>$$
$$..> \beta = arccos(\frac{l_2.sin(\alpha)}{l_2})$$

Získali jsme oba potřebné úhly.

Úhel α můžeme zvolit nekonečně mnoho způsoby. Pokud vybíráme úhly dělitelnými pěti máme zpravidla deset možných úhlů. Problém s volbou je, že čím je úhel větší tím je složitější jeho aplikace a pokud zvolíme úhel α malý tak nutně bude β velká. V mé implementaci z tohoto důvodu volím úhel co nejpodobnější situaci, kdy by diverzní trojúhelník byl rovnoramenný. Taková situace by nastala pro $l_1 = l_2$ a diverzní trojúhelník lze rozdělit na dva pravoúhlé. Použijeme trigonometrické vztahy:

$$\alpha \geq arccos\left(\frac{d}{d+\partial l}\right) = arccos\left(\frac{d/2}{l_1}\right)$$

Úhel α volím jako nejbližší větší dělitelný pěti.

Výše uvedený výpočet provádí v kódu funkce:
`void delay()`
`void delayFor(double time, SgPlan own)`
`getDiversion(int index, SgPlan own, double startDistance, double time)`
,které jsou vnořeně v tomto pořadí volány.

Jeden z problémů přesnosti je komunikační médium, tedy rádio. Můžeme sice přesně naplánovat, ve který moment zadat letadlu příkaz k vybočení a stejně tak k návratu, ale vlastnosti rádia zabraňují záruce přesnosti. Nelze dopředu předpokládat, že v potřebný čas bude kanál volný. Zamykání kanálu na určité časy také neexistuje. Možné řešení by bylo kromě parametrů manévru zadat po rádiu dotčenému letadlu také čas nebo pozici, ve kterých má příkaz provést.

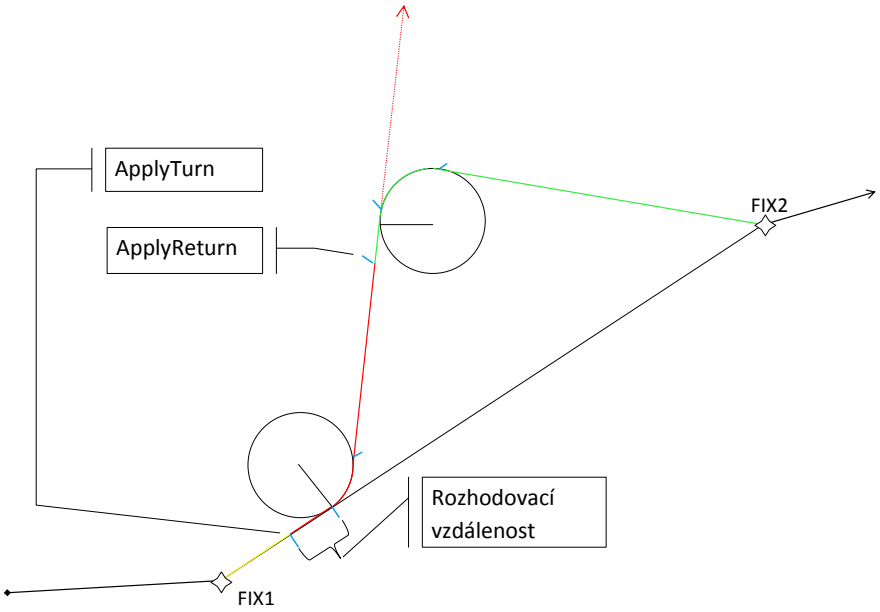
4.5 Plán trajektorie letadla

Agent ovládající letadlo si informace o naplánované trase drží ve třídě `GpsFlightPlanWrapperPolyline`. Zde je popsán celý letový plán ve formě funkce času. Dopředu jsou fyzikální vlastnosti započteny a po částech spojené přímky, spirály a kružnice definují celý plán, jak napovídá název. Do funkčního repertoáru letadlového agenta byla za účelem MITu připsána funkce zajišťující odbočení na kurz a návrat k původnímu plánu.

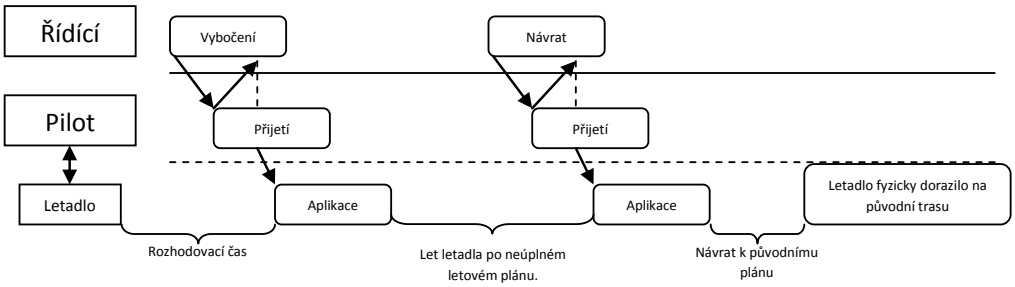
Celý proces je započat eventy `TURN_RECEIVED` a `RETURN_RECEIVED`, které jsou vytvořeny v po přijetí rádiové zprávy s příslušným pokynem. Na základě těchto dvou eventů a dvojitém handshaku jsou volány nové metody *processPilotApplyTurn* a *processPilotApplyReturn*. Při první diverzi je uložen zbytek původního plánu, protože při návratu bude znovu použit. K plánování regulérního a správného plánu se používá funkce `GpsFlightPlanWrapperPolyline.replan( )`. Jako vstup pro tuto funkci je nutná posloupnost bodů v prostoru a funkce nalezne plán, který prochází těmito body. Úhly mezi segmenty spojujícími jednotlivé body nesmí být příliš malé.

Při vytváření obou manévrů se nejdříve zanese do plánu simulovaná, deterministicky náhodná vzdálenost, která představuje vzdálenost, kterou pilot urazí, než se mu podaří změnit směr. Další body v zatáčce jsou plánovány, aby rádius zatáčky nebyl menší, než podle rychlosti dovoluje konstrukce letadla podle BADA parametrů [4].

Stejným způsobem je vytvořena rozhodovací vzdálenost u návratu k původnímu plánu. A za vložené průletové body je navázán zálohovaný zbytek původního plánu. Pomocné průletové body jsou vidět na obrázku 4.2 modrou barvou. V zatáčkách se letadlo pohybuje po kružnici s poloměrem daným konstrukcí a rychlostí letadla. Tečkovaně červeně je naznačena trasa, kterou by se vydalo letadlo pokud by se nepodařilo doručit příkaz pro návrat k původnímu plánu. Obě situace zahrnují rozhodovací vzdálenost, kterou letadlo urazí než simulovaný lidský pilot začne provádět zadaný manévr.



Obrázek 4.2: Grafický nákres výpočtu pozice pomocných řídících bodů pří vybočení z kurzu (ApplyTurn - červená trasa), a následný návrat na původní plán (ApplyReturn - zelená trasa).



Obrázek 4.3: Komunikace řídící - pilot.

Obrázek 4.3 popisuje protokol, jakým je na sektorovém rádiu komunikována diverze letové trajektorie. V předchozí kapitole byla stručně popsána pravidla sektorového rádia. Jak bylo řečeno, celý manévr se zakládá z vybočení z trajektorie a návratu na trajektorii. Pilot aplikuje příkaz vždy až po dokončení komunikace odpovědí radarovému řídícímu.

Během tohoto procesu se letadlo nachází ve třech stavech, různě náchylných na neschopnost komunikace s pilotem. Nejdůležitější je situace mezi dvěma pokyny kdy letadlo letí mimo svůj plán a čeká na pokyny řídícího. V ostatních stavech se letadlo pohybuje po kompletním letovém plánu. Pokud k přerušení komunikace dojde právě během prostředního stavu, může letadlo nedovoleně vyletět ze sektoru nebo vletět do kolize, se kterou se v případě funkčního rádia nepočítalo.

Kapitola 5

Experimenty

Nová funkčnost systému AgentFly byla testována podle tzv. scénářů. Jeden scénář uvnitř AgentFly představuje složku obsahující soubory s daty, které popisují počáteční stav simulace a vlastnosti systému, jakými jsou například definice Miles in Trail pro každý sektor. Bezpodmínečně musí složka se scénářem obsahovat soubor atc.xml, který je vstupní soubor pro načítání konfigurace. Ostatní soubory jsou v něm odkazovány. Obsahuje také základní vlastnosti celé simulace.

Další sobory definují sektor. Jsou v atc.xml uvedeny kódem:

```
<ConfFile Name="RSideSectorZDC34"
FilePath="../scenario_basic/visioSectorZDC34.xml"/>
```

Mezi tagy ohraničující definici sektoru uvádíme moduly k použití a individuální nastavení sektoru. Uvnitř atc.xml, v definici scénáře je nutné uvést parametr:

```
<Param Name="FlightsData1" Value="<!--#path-->AgentFlyDemo.fpx.xml"/>
```

, který odkazuje na soubor se specifikací letových plánů.

Každý letový plán, uzavřený v párovém tagu `<Flight/>`, obsahuje obsahuje informace o jednom letu. Vyjadřuji se zde pouze k těm důležitějším.

Tagy popíši na výstřižcích z konfigurace hlavního scénáře Intrail, na letadle SARAH.

- <TgfStartTime>00:00:00</TgfStartTime> : Uvádí čas vzniku, relativní k čase simulace. Letadlo SARAH tedy vznikne hned při počátku simulace.
- <Acid>SARAH</Acid> : Identifikátor, anglicky callsign, je krátká sekvence znaků, písmen a číslic, pomocí které se letadlo identifikuje řízení a na které během komunikace slyší. Identifikátor je jednoznačný a nerozlišují se v něm malá a velká písmena.
- <AcType>B757</AcType> : Typ letadla.
- <Beacon>2235</Beacon> : SQUAWK kód [15], je čtyřmístný oktální identifikátor sloužící k lokalizaci letadla. Letadlový respondér se radarovým přístrojům hlásí pomocí SQUAWK kódu.

- <TargetSpeed Type="TAS"Units="Knots»510</TargetSpeed> : Rychlost letadla v uzlech.
- <Route>ILM..ISO..KEMPR..WARNN..HPN..JFK</Route> : Trasa jako posloupnost jmen FIXů oddělených dvěma tečkami.
- <AssignedAltitude Units="HundredsOfFeet»330</AssignedAltitude> : Výška letu ve stovkách stop. V angličtině pod názvem Flight level [15].
- <Cid>128</Cid> : Computer ID - unikátní trojmístný kód v rámci jednoho centra pro každý aktuální let. Jakmile letadlo opustí sektor může být použit pro jiné letadlo.

Ostatní parametry jsou pro všechna letadla v uvedených scénářích neměnná, proto je neuvádím. Poslední typ souboru obsahuje konfigurace MITů, tyto konfigurace jsou popsány v kapitole 4.3.

5.0.1 Popis experimentu

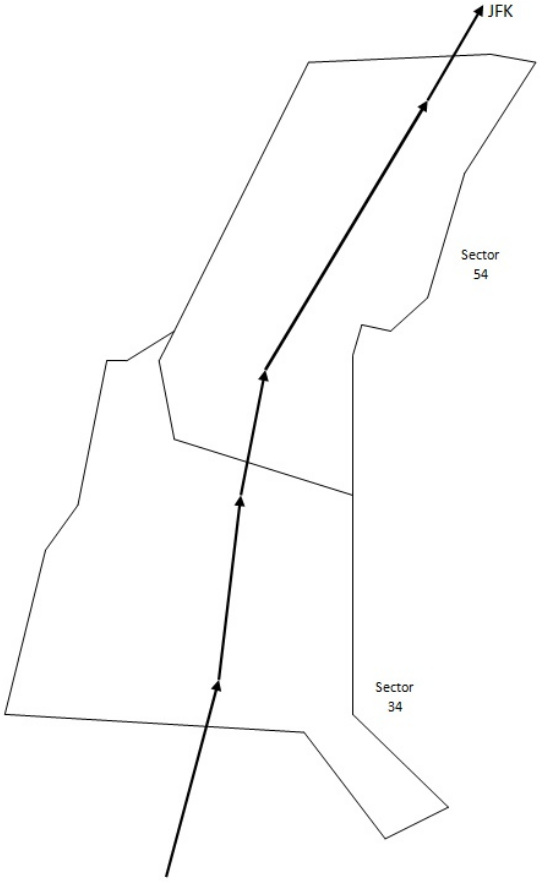
V každém scénáři se pro jednotlivé sektory provede několik různých nastavení MIT. Spustíme systém a průběh simulace porovnáme s činností opravdového operátora ve stejné situaci. Výsledné časy a vzdálenosti na výstupu ze sektoru porovnáme se vzdálenostmi v konfiguraci MIT. Ve vybraných scénářích navíc provedeme některé z těchto činností:

- Změna parametrů MIT a pozorování změny výsledků simulace, především vzdálenosti před výstupem ze sektoru.
- Pořízení videozáznamu obrazovek řídících agentů. Takto pořízená videa budou dostupná na přiloženém CD.

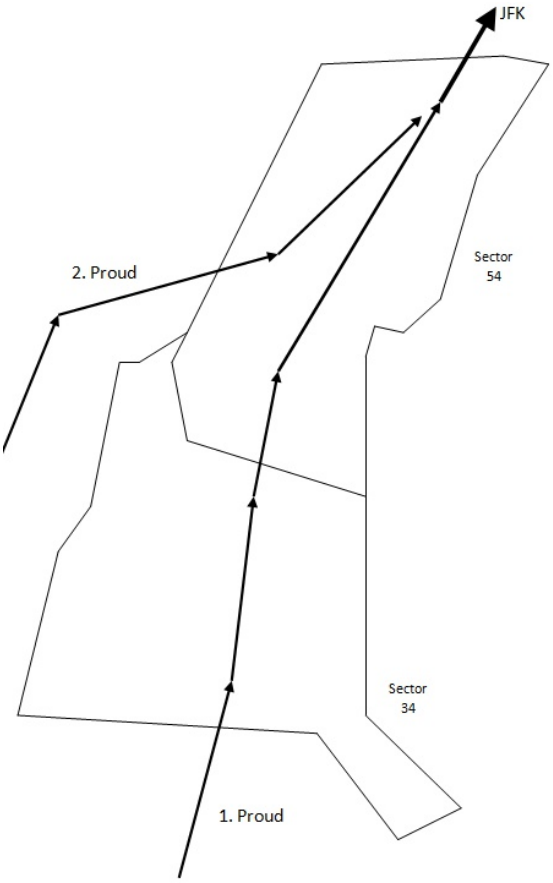
5.1 Scénáře

5.1.1 Scenář IntrailBasic

Scénář IntrailBasic bude první a nejkompaktnější scénář ukazující průlet n letadel se stejnými letovými plány a se stejným nebo podobným počátečním časem. V tomto scénáři je snaha odstínit vliv všech ostatních modulů k co nejčistšímu otestování nových modulů. Proto každé letadlo má jinou letovou hladinu, tím pádem nebude spuštěn dekonflikční modul. Jediné ostatní moduly, které by měly zapůsobit jsou moduly zajišťující vstup a výstup letadla do/ze sektoru. Na obrázku 5.1a je vidět kde proud letadel začíná a kudy zhruba přes oba sektory prolétá. Jelikož všech n letadel bude letět v jedné velké skupině a nebudou mít MITem definované rozestupy řídící letového provozu se pokusí skupinu roztáhnout. Simulace bude provedena pro 2 až 5 letadel. Pro oba sektory lze určit různé cílové vzdálenosti, aby byl MIT aplikován v obou sektorech.



(a) Popis scénária IntrailBasic.



(b) Hlavní scénář Intrail. K základnímu proudu po průchodu prvním sektorem se připojí z boku sekundární proud letadel a dispečer druhého sektoru se pokusí spojit dopravní toky.

Obrázek 5.1: Přes nákres dvou používaných sektorů 34 a 54 jsou naznačeny směry letových proudů v jednotlivých scénářích.

5.1.2 Scenář Intrail

Scénář Intrail je hlavní testovací scénář této práce. Obsahuje stejný letecký proud jako předchozí scénář, ale navíc se v druhém sektoru 54 připojí druhý proud z jihozápadu. Tento scénář simuluje standardní situaci z reálného světa, ve kterém se postupně více a více letadel spojuje do jednoho masivního proudu u koncového sektoru[15]. Náhodné skupinky letadel budou vpouštěny do těchto dvou proudů tak, aby docházelo k porušení MIT restrikcí. Pro druhý proud to znamená opoždění v čase cca 10 minut. V tabulce **??** jsou vidět odletové časy testovacích letadel.

V poslední verzi tohoto scénáře bude vytvořeno letadlo letící proti použitému proudu a bude testováno, zda-li funguje kooperace mezi Intrail modulem a moduly pro hledání a řešení kolizí. Pokud scénář proběhne se správným vyřešením kolize, vyhnutím neprioritního letadla nezařazeného v MIT, měl by systém obstát i u simulace se scénářem 34-54.

Tabulka 5.1: Základní nastavení scénáře Intrail se dvěma proudy. Pro každé letadlo je zaznamenán čas vytvoření.

Jméno letadla	SARAH	PAE	DUDE	GALACT	BRO	CYLON
Čas vstupu do simulace	00:00	00:00:30	00:01:50	08:30	10:20	15:00
Číslo proudu	1.	1.	1.	2.	1.	2.
Jméno letadla	EFG	COLIDER				
Čas vstupu do simulace	16:15	14:00				
Číslo proudu	2.	*				

Tabulka 5.1 uvádí nastavení konfiguračního souboru popisujícího lety pro scénář Intrail. Jednotlivá letadla jsou rozdělena do dvou proudů podle obrázku 5.1b. Poslední z letadel, pojmenováno patřičně COLIDER, nepatří do žádného MITu a tedy ani no žádného proudu, slouží pouze k otestování řešení kolizí mezi prioritním letadlem a letadlem bez priority.

5.1.3 Scenář 34-54

Největší scénář používaný v AgentFly2. Obsahuje kompletní letecký provoz nad sektory 34 a 54, to je na Washingtonském letovém středisku sektory Norfolk a Salisbury[16] po dobu jedné hodiny. Data byla poskytnuta Americkým úřadem pro letectví ve Washingtonu. Zde je použit jeden MIT pro letadla letící směrem na New York a okolí.

5.2 Měření

Měření spočívá z výpočtu vzdálenosti dvou bodů uvnitř 3D simulované reality. Vzdálenost je zaznamenána vždy při odletu letadla ze sektoru v porovnání s dalším letadlem v pořadí za ním. Všechny jednotky jsou v námořních mílích. Pro scénář IntrailBasic je pořízeno pouze měření v prvním sektoru, jelikož v druhém sektoru jsou letadla již roztažena.

5.2.1 IntrailBasic scénář

Tabulka 5.2: Scénář IntrailBasic se čtyřmi letadly vylétajícími najednou. Vztah mezi nastavenými rozstupy a naměřenými rozstupy

MIT parametry(NM)		Rozstup na výstupu ze sektoru (NM)		
min. rozstup	ideální rozstup	1.	2.	3.
10	13	10.85	12.66	13.85
9	11	10.23	9.40	10.73
7	10	8.84	9.07	8.35
6	9	7.63	7.97	7.52
5	8	6.99	7.16	4.76
4	7	6.11	5.63	6.41
4	6	4.81	5.25	4.45
3	5	4.04	3.95	3.55
2	4	3.21	3.54	2.82
1.5	3	2.45	1.85	1.63

Tabulka 5.3: Měření scénáře IntrailBasic na pouze dvou letadlech.

MIT - opt. rozstup	20	18	16	14	12	10	8	6	5
Naměřený rozstup	19.18	17.65	13.77	12.43	9.83	7.66	5.8	4.17	3.62
Použitý diverzní úhel	45	40	40	40	35	35	30	25	25

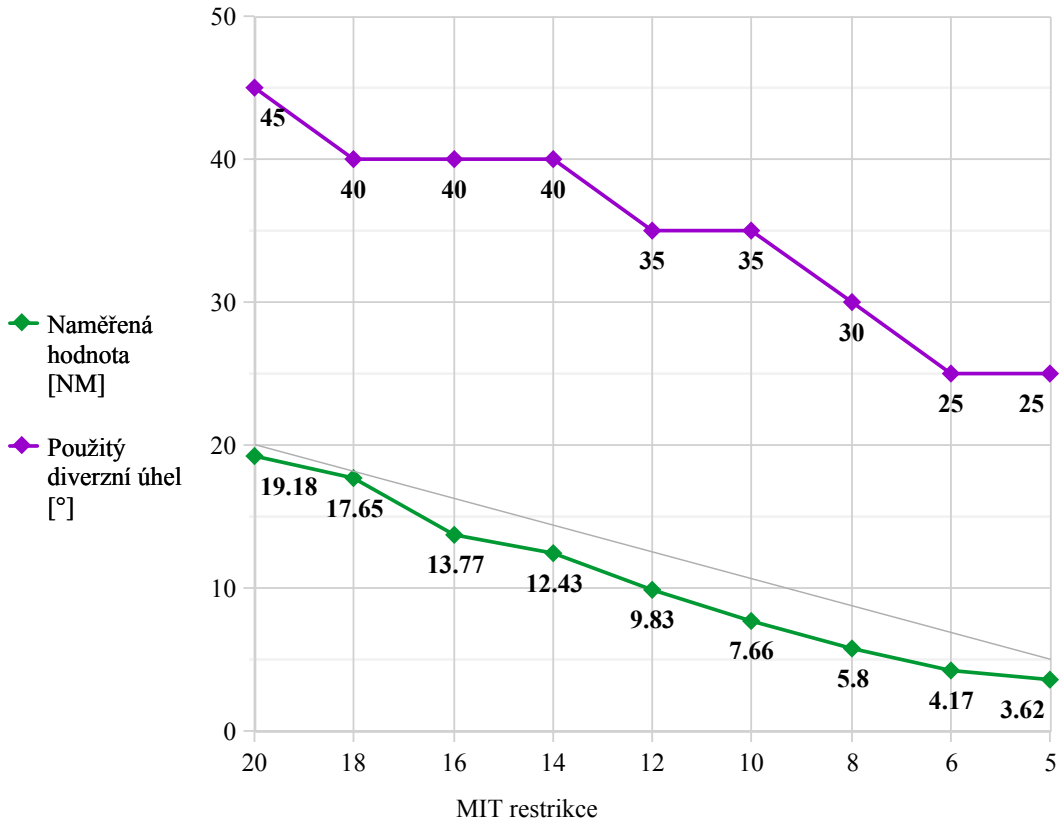
Ve všech měřeních, jak lze vidět v tabulkách 5.2 5.2, jsou vzniklé rozstupy menší nežli cílové. Na grafu 5.2 je znázorněna chyba v závislosti na cílové vzdálenosti. Tento graf vychází ze scénáře IntrailBasic se dvěma letadly, tzn. tabulky 5.3. Jelikož v tomto scénáři začínají všechny letadla ze stejné pozice v téměř stejném čase, zpoždění vždy vychází přesně z nastavení MIT rozstupů.

5.2.2 Intrail scénář

Průběh scénáře Intrail je dobře viditelný na videu, které se nachází na CD přiloženém k této bakalářské práci.

V tabulce 5.4 jsou pro oba sektory vypsány naměřené rozstupy, po řádcích mezi prvním a druhým, druhým a třetím atd.. V dvou levých sloupcích jsou údaje ze sektoru 34.. Na stejném řádku je vždy kód letadla a rozstup k dalšímu letadlu v pořadí. V tabulce není vidět údaj o letadle BRO, které bylo poslední. Ve scénáři došlo k hladkému splynutí dvou proudů. Došlo i k zipování ve smyslu, že mezi seřazená letadla ze sektoru 34. se přiřadilo nově příchozí letadlo.

Kromě stejných systematických chyb jako ve scénáři IntrailBasic docházelo pro některé konfigurace k selhání hledání správného manévru z důvodu, že sektor 54. je úzký a manévry



Obrázek 5.2: Grafická reprezentace tabulky 5.2. Tenká čára nad hodnotami naměřeného rozstupu orientačně ukazuje zamýšlený rozstup.

Tabulka 5.4: Výsledné rozstupy scénáře Intrail.

Naměřené rozstupy [NM]					
			Sektor 34	Sektor 54	
		SARAH	11.4	12.5	SARAH
		PAE	11.12	10.65	PAE
		DUDE	62.3	16.24	GALA
MIT restrikce[NM]				35.3	DUDE
Optimální rozstup:	13			11.64	CYLO
Minimální rozstup:	10			16.2	EFG

by sahaly mimo sektor. Test na dekonflikci se vydařil. Při standardním nastavení se modul pro řešení kolizí postaral o letadlo COLIDER změnou jeho letové hladiny. Při zakázané vertikální dekonflikci použil správnou horizontální dekonflikci na letadlo bez priority (COLIDER). Poslední chyba nastávala pouze při některých konfiguracích. Dvě z letadel v různých proudech se naskytla v situaci, že byla po aplikaci MIT v kolizi a neexistovalo vertikální řešení. Jakmile se však modul pokusil vytvořit úhybný manévr selhal, protože se pokusil provést manévr na stejném segmentu, který byl použit pro manévr v MITu. Implementace zatím není schopna řešit tento druh problému.

5.2.3 Reálný scénář

Pro reálný scénář byl vytvořen jeden MIT, který je podobný jednomu reálnému. Zaobírá se letadly letícími směrem na New York a okolí (Hlavní FIX je JFK). Dále byly použity další MITy, které se ale neprojevují tak jako JFK. Použitá konfigurace:

```
<Intrail Name="JFK_34" Destination="JFK:JFK..BOS" OptimumDistanceNM="11"
 MinimumDistanceNM="7"/>
<Intrail Name="MIA_34" Destination="MIA" OptimumDistanceNM="11"
 MinimumDistanceNM="7"/>
<Intrail Name="FLL_34" Destination="FLL" OptimumDistanceNM="11"
MinimumDistanceNM="7"/>
```

Výsledky zde nebyly dobře měřitelné. V několika případech modul zasáhl na malé skupinky letadel se stejným cílovým letištěm. Letové plány ale opouštějí sektor v různých bodech a hlavně v různých vzdálenostech od cílového letiště. Data vzdáleností letadel nepředstavují věrohodně vzdálenosti letadel vzhledem k cílovému sektoru. Efekt MIT je dobře vidět na přiloženém videu.

5.3 Analýza výsledků

Na přesnosti se nejvíce projevují dva problémy. Zaprvé to je nepřesný výpočet vhodné diverzní trajektorie. V použitém výpočtu se zanedbává kruhová podstata zatáček. Jednak kvůli tomu, že chyba není řádově velká a kvůli zjednodušení prvního algoritmu. Jak je vidět na obrázku 4.2, který popisuje podobu plánu v simulaci reality, trasa není trojúhelníková ale kratší. Takováto chyba je pro malé diverzní úhly velmi malá ale projevuje se v závislosti na délce segmentu od 15° až 30° vybočení a návratu. V algoritmu je, jak je vidět na obrázku 4.1, použito zjednodušení pomocí trojúhelníku. Lze tedy očekávat,že skutečné zpoždění bude kratší. Ale díky tomu, že při zpoždění letadla je cílový čas, na který se zpožďuje, explicitně uložen a další výpočty jsou odvozeny od této hodnoty nebude nikdy tato chyba zanesena řetězově.

Druhá chyba je zanesena rádiem. Jelikož v mnoha těchto scénářích vlétala letadla ve velkých skupinách najednou, a tím pádem musela všechna letadla být vybočena ve stejný moment. Rádio bylo přetíženo jednak povely od řídícího tak odpověďmi od pilotů, a jelikož

pilot nebude měnit kurz, dokud se mu nepovedlo úspěšně odpovědět sektoru, tak čekáním na rádiové ticho si dále zmenšuje trasu, kterou v manévru poletí. Omezení rádiem zanáší tedy úplně stejný druh chyby : Trasa, kterou letadlo výsledně poletí, je kratší, než jakou zamýšlel řídící letového provozu původně.

Poslední typ chyby je náhodný čas, po který se pilot rozhoduje a aplikuje příkazy. Pro prevenci této chyby byl použit algoritmus, který volí v trojúhelníku diverzní trasy minimální úhly a maximální délku trasy, kterou letadlo vynechá.

Dohromady chyby tvoří celkovou chybu, která nepřímo úměrně stoupá s úhlem vybočení. V produkčním kódu je tato systematická chyba řešena lineárním zvětšením cílové vzdálenosti.

Zde uvádím některé dříve nastíněné problémy a jak se s nimi systém vypořádal:

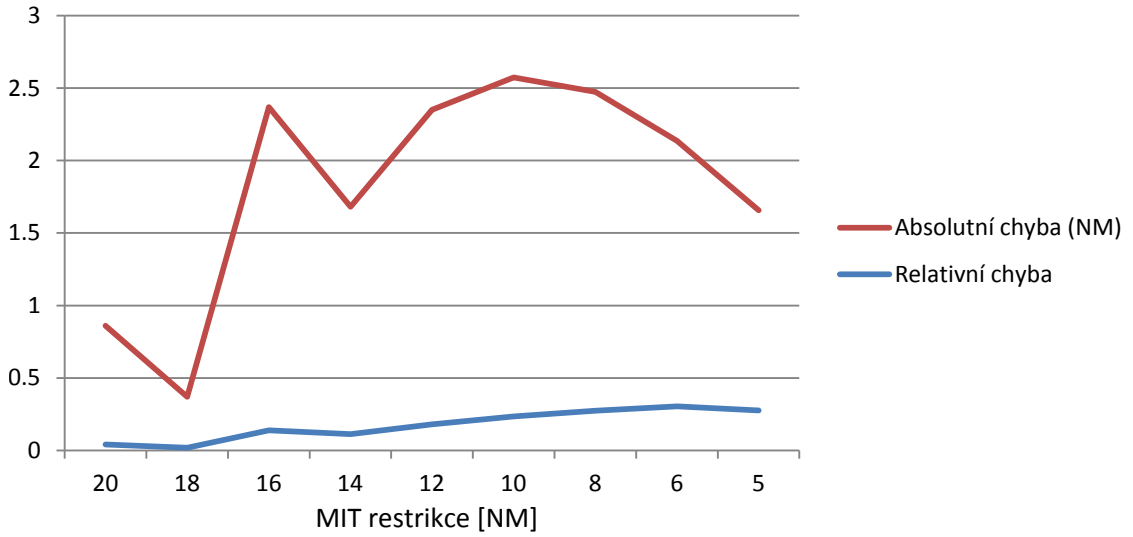
- Řešení kolizí bylo otestováno v několika případech a je i ve scénáři Intrail. V situacích, kdy došlo ke kolizi jednoho letadla, na které byl aplikován MIT, a druhého bez příslušnosti k některému z MITů, proběhla dekonflikce správně a to úpravou trajektorie letadla bez priority. Ostatní případy dopadají dle očekávání. Dekonflikce se nezdařila pouze pokud neexistovalo platné řešení pomocí, které by bylo implementované.

- Obrat mimo sektor : Letové plány použité v měření procházejí dvěma sektory, viz obrázek 5.1a. Řídící nesmí dovolit letadlu opustit sektor, aniž by letadlo předal jinému sektoru. Proto všechny manévry, které používáme pro účely MITu nesmí překročit hranici sektoru. Jižnější sektor 34 má lepší tvar pro pro zpožďovací manévry. Jelikož tvar sektoru 34 je více kruhový a vzhledem k 1. proudu velmi široký mohl by řídící provádět manévry, které by více než zdvojnásobily vzdálenost v sektoru. V sektoru 34 nebyla funkce řídícího v aplikaci MIT nijak omezena podobou sektoru.

  V sektoru 54 plánovač manévrů narážel na úzký tvar. V aktuální implementaci lze použít pouze prostor mezi dvěma FIXy. V jediném použitelném segmentu v sektoru 54 při nutnosti většího zpoždění nešlo najít přibližně rovnoramenný trojúhelník s potřebným obvodem tak, aby nezasahoval mimo sektor. Algoritmus použitý v této práci funguje na podobném principu. V aktuální verzi plánovač manévrů nenašel řešení, i když existovalo. Jedinou cestou ze je změna algoritmu pro hledání tvaru manévru, použití jiné metody nebo kombinací více metod, jako je například změna rychlosti.

- Příliš vysoké úhly : V umělých scénářích s extrémními špičkami dopravy, ke kterým v obvyklých situacích nedochází, byla snaha pozdržet letadlo neúměrně délce segmentu, na kterém se opoždění provádělo. Použitý zpožďovací algoritmus se chová rozumě pro až 150% prodloužení délky segmentu. S rostoucím úhlem se více projevovalo zanedbání opravdového tvaru zatáčky 4.2. Proto je v kódu v aktuální verzi zavedeno omezení na maximální velikost úhlu. Po přepracování algoritmu pro výpočet správných parametrů bude tento problém odstraněn.

- Přesnost : Jak je vidět z výsledků všech měření existuje zde vysoká chyba. Měřená vzdálenost je ve většině případů menší než cílová vználenost, na kterou jsme chtěli letadlo opozdit a to průměrně o 15% 5.3. Kupodivu chyba roste směrem k menším diverzním úhlům. To může být způsobeno tím, že stejná nebo podobná chyba se vzhledem k menší



Obrázek 5.3: Graf popisující závislost relativní a absolutní chyby vzhledem k optimální vzdálenosti v MIT restrikci naměřené z dat ve scénáři IntrailBasic se dvěma letadly.

cílové vzdálenosti bude zdát větší. Pokud se podíváme na absolutní chybu uvidíme že ta alternuje ve mílovém rozsahu okolo průměrné hodnoty 1,65 NM. Alternování absolutní chyby si vysvětluji zaokrouhlováním diverzního úhlu na hodnoty dělitelné pěti. Při spojité změně MIT parametrů bude se absolutní chyba nespojitě měnit v bodech, kde plánovač manévru použije jiný úhel.

Na grafu 5.3 je vidět, že absolutní chyba zůstává řádově stejná. To v dopadu na relativní chybu působí jejím vzrůstem směrem k menším cílovým vzdálenostem (17,5% pro 5-ti mílové rozstupy). Tento graf je vytvořen z dat v tabulce 5.3.

Na scénářích se také projevuje rychlost letadla. V použitých scénářích jsou letadlům nastaveny vysoké rychlosti, na kraji konstrukčních doporučení. Vysoká rychlost se projevuje dvěma směry. Poloměry zatáček letadel stoupají úměrně s rychlostí, čím je tedy vyšší rychlost tím se více projevuje chyba ve zjednodušeném výpočtu přes trojúhelník. Druhý dopad na přesnost je zvýšení chyby zanesené rozhodovacím intervalem. Jelikož ten je udán v časových jednotkách tak se do letový plán projevuje násobený rychlostí jako vzdálenost. Z toho plyne, že chyba je lineárně závislá na rychlosti letadla. Pokud by rychlost byla velmi malá, k čemuž v letecké dopravě nemůže dojít, tak se jednak v porovnání se vzdálenostmi uvnitř sektoru ztratí vliv kruhového charakteru zatáčky ale i komunikace bude relativně k letu instantní.

Poslední efekt na přesnost, který zde zmíním je aktuální výška letadla. Ve vyšších sektorech se letadla pohybují ve výšce od 24 000 do 34 000 stop. Velké rozdíly tlaků v těchto výškách způsobují rozdíly v rychlostech a spotřebě letadel. Hlavní rozdíl mezi trasami v různých hladinách je však jejich délka. Čím nižší letová hladina tím kratší trasa, a tím vyšší rychlost relativně k zemi. Měřením na systému byl naměřen rozdíl vzdáleností o jednu míli u letadel vylétajících ze stejného bodu, letící přes jeden sektor

stejnou rychlostí, ale v různé výšce a to pouze o 1000 stop. A mílovou chybu může zanést do MIT zásah dekonflikčního modulu, který změní některému z letadel v MITu jeho letovou hladinu.

Kapitola 6

Závěr

Cílem této práce bylo zprovoznit funkcionalitu vyvažování letového provozu v sektorech pro systém AgentFly. Největší výzvou v implementační části byla úprava ostatních komponent. Systém nenabízel značnou část funkcí, kterou používá nyní. Některé nové funkce byly popsány v této práci, jako například horizontální změny v pilotově plánu. Podle architektury řídícího agenta jsou přidány dva nové moduly, jeden reprezentující činnost spojenou s radarovým řídícím a druhý s počítačem. Navržený algoritmus používá pro aproximaci vybočení z letového plánu a návrat téměř rovnostranný trojúhelník. Pomocí sektorového rádia je letadlo navigováno podél ramen tohoto trojúhelníku se základnou na původním plánu.

Výsledkem je konfigurovatelný nezávislý modul, který v běžných situacích funguje bez problému. Za nebezpečné situace označuji situace, ve kterých dochází ke kolizím mezi letadly náležícími do některých MIT, které nemají jiné nežli horizontální řešení. Dále například nedostatek prostoru pro provádění řídícím koordinovaných manévrů. Jeho funkčnost byla otestována na dvou vymyšlených scénářích a na jednom reálném, kde se potvrdila jeho funkčnost a vyskytly některé chyby, jejichž doporučené řešení nastíním v následující sekci.

Funkcionalita Miles in Trail je prozatím založena pouze na pozdržování letadel změnou tras dvěma zásahy řídícího v horizontální oblasti. Během experimentů s první verzi návrhu MIT algoritmu se ukázalo, že výpočet pomocí aproximace je nedostatečně přesný. Kvůli zanedbání opravdového tvaru letecké trasy při vybočení letadla z kurzu a návratu na původní trasu, která v zatáčkách opisuje kruh, byla do výpočtu správných úhlů zanesena velká chyba. Tím že trasa je ve skutečnosti kratší, nežli trojúhelníková aproximace, která byla při výpočtu použita, byly dosažené rozstupy menší než požadované.

Všechny zdroje nepřesností lze opravit úpravou algoritmu a úpravou SgPlanu, který představuje naplánovanou trasu letadla ze strany řídícího. Pokud bude rozhraním SgPlanu možné rozhodovat o návratu z vybočení parametricky nebo funkcí, místo přesného plánu dopředu bude možné dynamicky reagovat na neurčitosti v průběhu letového plánu. Úprava algoritmu pokryje další část chyby. Nepřesnost algoritmu považuji za hlavní důvod chybných vzdáleností. Ale protože je chyba vždy podobná a předvídatelná, v aktuální implementaci jsou zavedeny konstanty pro úpravu parametrů před výpočtem algoritmu.

6.1 Možnosti rozšíření

Další rozšíření MIT modulů by měla být implementace změny rychlosti, a to jak na straně GpsPlanu[1] tak SgPlanu[2]. Změnou rychlosti bude možné dosahovat přesnějších rozestupů pomocí malé změny rychlosti.

Po dokončení MIT bude na řadě implementace dalších metod pro úpravu hustoty provozu, například Holding pattern jako nejjednodušší ze skupiny metod kontroly hustoty letecké dopravy.

Plánovač trajektorie mimo letový plán hledá k pozdržení jednoduchý, téměř rovnoramenný trojúhelník, s délkou ramen podle požadovaného zdržení. V konvexních sektorech je tento postup dostačující, ale u nekonvexních a úzkých sektorů má potíže s přesahem do vedlejšího sektoru. V dalších verzích by měl být zaveden všestrannější plánovač, který zváží více možných tras v závislosti na sektoru a bude počítat reálnou, neaproximovanou trasu počítající s kružnicí v zatáčkách.

Samotné konfigurování MIT by se mohlo přiblížit reálné podobě zavedením vyjednávání mezi sektory o podobě MIT restrikcí dynamicky tato nastavení měnit v průběhu simulace.

[1] Plán trajektorie letadla uvnitř pilot agenta.

[2] Plán trajektorie letadla z pohledu řídícího. SgPlan je omezen na sektor a zahrnuje základní plán a úmysly řídícího.

Literatura

[1] C. I. Agency, "Transportation factbook," *https://www.cia.gov/library/publications/the-world-factbook/geos/us.html*.

[2] R. Synnestvedt, H. Swenson, H. Erzberger, and A. R. Center, *Scheduling logic for miles-in-trail traffic management*, vol. 4700. National Aeronautics and Space Administration, Ames Research Center, 1995.

[3] D. Šišlák, M. Pěchouček, P. Volf, D. Pavlíček, J. Samek, V. Mařík, and P. Losiewicz, "Agentfly: Towards multi-agent technology in free flight air traffic control," *Defence Industry Applications of Autonomous Agents and Multi-Agent Systems*, pp. 73–96, 2008.

[4] A. Nuic, "User manual for the base of aircraft data (bada) revision 3.7," *Atmosphere*, vol. 2010, p. 001, 2010.

[5] S. Shresta and R. Mayer, "Benefits and constraints of time-based metering along rnav star routes," in *Digital Avionics Systems Conference, 2009. DASC'09. IEEE/AIAA 28th*, pp. 2–B, IEEE, 2009.

[6] FAA, "Ground delay program„"

[7] FAA, "Holding patterns."

[8] ContrailScience.com, "Holding pattern definition," *http://contrailscience.com*.

[9] M. P. David Šišlák, Přemysl Volf, "Multi-agent simulation of en-route human air-traffic controller," p. 6.

[10] FAA, "Point-out in air traffic control,"

[11] FAA, "En route automation modernization (eram),"

[12] H. Erzberger, "Transforming the nas: The next generation air traffic control system," 2004.

[13] J. Chen and C. Chen, *Foundations of 3D graphics programming: using JOGL and Java3D*. Springer-Verlag New York Inc, 2008.

[14] N. I. of Standards and Technology., "Linked list description," *http://nist.gov/dads/HTML/linkedList.html*, 8 2004.

[15] M. Nolan, *Fundamentals of air traffic control.* Delmar Pub, 2010.

[16] W. A. R. T. C. Center, "Washington center sectors,"

Příloha A

Obsah CD

```
ROOT
...bakalar.pdf
...\Videa -
...\...\IntrailBasic-ZDC.avi
...\...\IntrailBasic-visio.avi
...\...\Intrail.avi
...\...\Intrail-ZDC34.avi
...\...\Intrail-ZDC54.avi
...\...\Intrail - Real traffic.avi

...\Zdrojové kódy -
...\...\Moduly řídícího -
...\...\...\EomAtaIntrail.java
...\...\...\EomRSideIntrail.java
...\...\...\... etc ....

...\...\Ostatní zdrojové kódy -
...\...\...\GpsFlightPlanWrapperPolyLine.java
...\...\...\SgPlan.java
...\...\...\HorizontalDiversionParams.java
...\...\...\... etc ....

...\Konfigurační soubory -
...\...\Globální konfigurace -
...\...\Scénáře -
...\...\...\IntrailBasic -
...\...\...\IntrailBasic2 -
...\...\...\Intrail -
...\...\MIT konfigurace -
...\...\...\IntrailBasic.xml -
...\...\...\Intrail34.xml -
...\...\...\Intrail54.xml -
...\...\...\Real34.xml -
```