SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3675

Poboljšanje sustava određivanja reputacije autonomnih sustava temeljeno na BGP prometu

Luka Matetić

Zagreb, lipanj 2014

Zahvaljujem se doc. dr. sc. Stjepanu Grošu na stručnom vodstvu i pomoći u izradi ovog završnog rada.

Sadržaj

1. Uvod

Internet danas čine međusobno nezavisne mreže ili skupine mreža koje se nazivaju autonomni sustavi. Davatelji pristupa Internetu posjeduju autonomne sustave i određuju njihov rad formirajući privatne politike usmjeravanja. S obzirom na decentraliziranost Interneta, davatelju pristupa Internetu se ostavlja sloboda biranja načina poslovanja što znači da nisu obavezni voditi računa o kvaliteti prometa koji stvaraju ili prosljeđuju. Ovakva sloboda je dovela do toga da pojedini davatelji svojevoljno pružaju pristup Internetu korisnicima koji stvaraju velike količine neželjenog prometa ili pak prosljeđuju neželjen promet radi zarade. Neželjenim prometom smatraju se podatci koji se prenose Internetom i opterećuju mrežne resurse tako da to ne koristi vlasnicima tih resursa. Neki primjeri takvog prometa bili bi napadi uskraćivanjem Internet usluge, virusi te neispravan promet koji se zbog pojave grešaka ne smije prosljeđivati.[15] Iako se veći davatelji pristupa Internetu ograđuju od propuštanja neželjenog prometa jer vode računa o kvaliteti i sigurnosti usluge koju pružaju svojim korisnicima, to nužno ne znači da vode računa i o prometu koji izlazi iz njihovog autonomnog sustava. Kao rješenje navedenom problemu, u sklopu Dejdarovog rada [1], predložena je ideja o računanju reputacije za autonomne sustave. Svaki davatelj pristupa Internetu koji bi imao implementiran sustav za računanje reputacije u svom autonomnom sustavu bi mogao prilagoditi svoju politiku usmjeravanja prometa te omogućiti bolju uslugu svojim korisnicima. Isto tako, davatelj bi trebao voditi računa kakav promet izlazi iz njegovog autonomnog sustava jer bi njegovi susjedi koji imaju implementiran sustav za računanje reputacije također vodili računa o usluzi koju pružaju svojim korisnicima. Ovakvim bi se rješenjem davateljima koji vode računa o svojim korisnicima povećala kvaliteta usluge, ali isto tako bi se prema autonomnim sustavima koji stvaraju velike količine neželjenog prometa povećalo filtriranje prometa što bi u konačnici poboljšalo kvalitetu i sigurnost Interneta. Ovaj rad se bavi daljnjom optimizacijom već postojećeg reputacijskog sustava koji funkcionira na temelju praćenja BGP prometa. Izvorni program napisao je T. Dejdar, a optimizirao M. Fabris.

U radu će se razmatrati odnosi između autonomnih sustava, njihova komunikacija putem BGP protokola te važnost i upotreba BGP-a u sklopu autonomnih sustava. Također će se prezentirati i model programskog ostvarenja reputacijskog sustava temeljenog na praćenju BGP prometa uz priložene podatke i mjerenja.

U drugom poglavlju govorit će se prvenstveno o ustrojstvu samog Interneta te o raznim organizacijama koje vode računa o njegovoj stabilnosti i sigurnosti. Spomenut će se razni modeli usmjeravanja, kako na Internetu, tako i lokalno, a potom će se prezentirati BGP i usmjeravanje korištenjem BGP-a na Internetu.

U trećem i četvrtom poglavlju analizirat će se konkretna implementacija reputacijskog sustava korištenjem programskog jezika Python 2.7 te GitHub repozitorija. U trećem poglavlju govorit će se o početnoj implementaciji sustava te koracima potrebnim da se program podesi za korištenje dok će se u četvrtom

poglavlju govoriti o mogućnostima optimizacije i konkretnim koracima poduzetim u tom pogledu.

U petom poglavlju navest će se provedena mjerenja i rezultati prema kojima će se zaključiti o daljnjem poboljšanju i mogućnosti korištenja sustava u praksi.

2. Usmjeravanje na Internetu

Internet danas čine zasebne mreže ili skupine mreža koje se nazivaju *autonomni sustavi* (*engl. Autonomous System, AS*). Po razinama AS-ova, Internet dijelimo na tri razine (*engl. Tier*), razina 1 AS-ovi, razina 2 AS-ovi i razina 3 AS-ovi. Razinom 1 se smatraju svi AS-ovi koji čine *okosnicu Interneta* (*engl. Backbone*). Oni se međusobno spajaju i ne naplaćuju razmjenu prometa među sobom (*engl. Peering*). Razinom 2 se smatraju svi AS-ovi koji koriste uslugu infrastrukture AS-ova razine 1 kao glavnog posrednika za promet te sami postaju posrednici (*engl. Transit*) i pružatelji Internet usluge drugim AS-ovima iste ili niže razine. AS-ovi razine 3 su najniže rangirani AS-ovi koji se uglavnom spajaju na AS-ove razine 2 i pružaju Internet uslugu krajnjim korisnicima. [11]

Autonomni sustav je skupina mreža (adresnih prefiksa) pod jurisdikcijom jednog *davatelja pristupa Internetu* (*engl. Internet Service Provider, ISP*) koji definira jedinstvenu, jasnu i tajnu politiku usmjeravanja u odnosu na druge autonomne sustave što čine Internet. Uobičajeno je da jedan ISP drži samo jedan AS pod sobom. Svi se AS-ovi mogu jedinstveno odrediti na Internetu uz pomoć 16-bitnog ili 32-bitnog cijelog broja (*engl. Autonomous System Number, ASN*) kako bi se moglo provesti usmjeravanje među njima. [1]

Internetski registrari (*engl. Regional Internet Registry, RIR*) esencijalni su za pravilan rad Interneta i bave se dodjelom ASN-a AS-ovima. Brojevi se dijele na temelju regionalnog područja, propisanog od strane međunarodne organizacije za dodjelu brojeva *Internet Assigned Numbers Authority* (*IANA)*. Postoji pet Internetskih registrara, a to su po područjima: Arin (Sjeverna Amerika), AfriNIC (Afrika), RIPE NCC (Europa, Bliski istok i Rusija), APNIC (Australija i jugoistočna Azija) i LACNIC (Južna i Srednja Amerika). Registrar novom autonomnom sustavu dodjeljuje slobodan broj iz skupa brojeva koji je dobio od IANA-e. Iako su RIR-ovi izvorno zamišljeni ne samo da dodjeljuju brojeve, već i da određuju pravila i nadziru AS-ove, politički i mrežno decentralizirana struktura Interneta dokaz je neuspjeha takve ideje i jedno je od temeljnih značajki Interneta danas. [1]

2.1. Usmjeravanje

Usmjeravanje se definira kao određivanje puta, odnosno niza usmjernika, kojima će se slati promet kroz mrežu, od izvorišta do odredišta. Kako bi usmjeravanje radilo, svaka mreža mora definirati načine unutarnjeg usmjeravanja. Načini unutarnjeg usmjeravanja mogu biti statički ili dinamički. *Statički načini unutarnjeg usmjeravanja* koriste se u malim mrežama gdje su susjedski odnosi usmjernika stalni te se u usmjernike unose putem konfiguracije. *Dinamički načini unutarnjeg usmjeravanja* implementiraju se u obliku protokola u usmjernicima u većini današnjih mreža zbog mogućnost neprestanog funkcioniranja mreže bez uplitanja ljudskog faktora unatoč promjenama topologije samih mreža.

Protokoli usmjeravanja na Internetu se dijele na *protokole vanjskog usmjeravanja* (*engl. Exterior Gateway Protocol, EGP*) i na *protokole unutarnjeg usmjeravanja* (*engl. Interior Gateway Protocol, IGP*). IGP-ovi se koriste za usmjeravanje prometa

unutar AS-a te AS ima izbor koji će IGP koristiti. Prema vrsti algoritma, protokoli unutarnjeg usmjeravanja na Internetu se dijele na protokole vektora udaljenosti i protokole stanja veze.

Usmjeravanje prema vektoru udaljenosti (*engl. distance vector routing*) dinamički je način usmjeravanja kod kojeg svaki usmjernik ima tablicu koja daje najkraću udaljenost za svako odredište i prvi korak prema njemu, pri čemu se ta poznata udaljenost stalno osvježava na temelju razmjene podataka s ostalim usmjernicima u mreži. *Usmjeravanje prema stanju veze* (*engl. link state routing*) dinamički je način usmjeravanja koji se temelji na razmjeni podataka o topologiji mreže i izmjerenih podataka o aktualnom stanju veza među usmjernicima. Razlika između zabilježenog i stvarnog stanja veza ovisi o periodu razmjene aktualnih podataka. Što je razmjena aktualnih podataka češća, to se više resursa troši na kvalitetu informacija, ali se manja količina prometa stigne usmjeriti korištenjem tih informacija.[2]

Protokoli unutarnjeg usmjeravanja ili IGP-ovo na Internetu su RIP, OSPF i IS-IS, dok je BGP (*engl. Border Gateway Protocol*) jedini EGP koji se koristi za usmjeravanje među različitim AS-ovima na cijelom Internetu te su ga sve mreže koje žele biti dio Interneta obavezne implementirati.

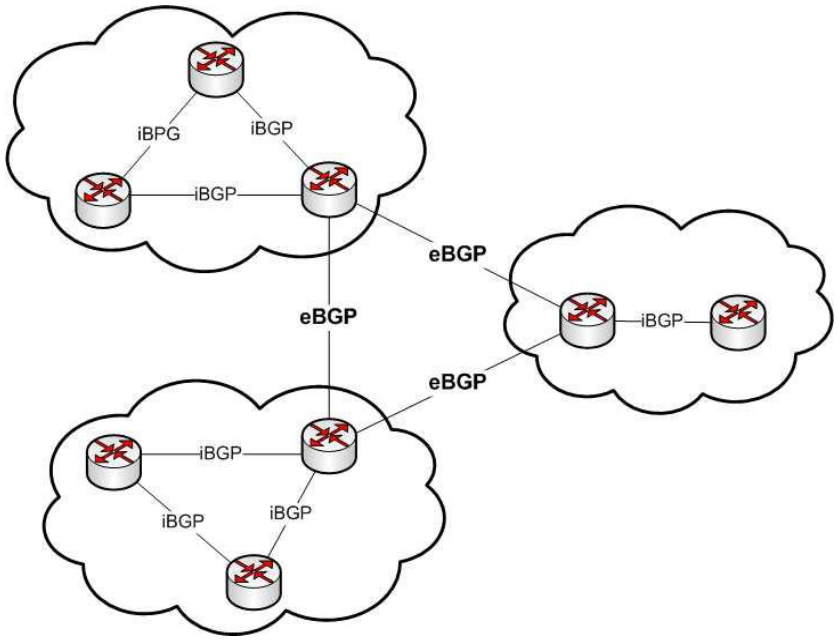
2.2. Usmjeravanje između autonomnih sustava

BGP je protokol usmjeravanja koji pripada četvrtom sloju TCP/IP (Internet) modela (aplikacijski sloj) i koristi spojnu uslugu TCP protokola između dva BGP usmjernika na vratima 179 kako bi formirao vezu potrebnu za ispravan rad. BGP se dijeli na dva dijela: unutarnji BGP (*engl. Internal Border Gateway Protocol, iBGP*) i vanjski BGP (*engl. External Border Gateway Protocol, eBGP*). *iBGP* omogućava vezu između BGP usmjernika (*engl. BGP speakers)* koji se nalaze unutar istog AS-a i koristi se za razmjenu i transport adresnih prefiksa dok *eBGP* omogućava vezu između BGP usmjernika koji se nalaze u različitim susjednim AS-ovima i koristi se za implementaciju politika usmjeravanja i komunikaciju među AS-ovima. Drugi naziv za BGP usmjernike jest rubni usmjernici ili granični usmjernici. Autonomni sustavi su osnovni razlog postojanja BGP-a.

BGP koristi *usmjeravanje prema vektoru puta* slično usmjeravanju prema vektoru udaljenosti kojeg koriste IGP protokoli, ali uzima u obzir put kojim putuje paket kao niz AS-ova od izvorišnog AS-a do odredišnog AS-a kako bi se izbjegle petlje u usmjeravanju prometa između AS-ova. Sukladno tome, dužina puta se mjeri prema broju AS-ova kroz koje paketi prolaze dok ne dođu do odredišnog AS-a (kod IGP protokola se mjeri broj usmjernika). Pri tome je također moguće odluku o usmjeravanju prilagoditi politici pojedinog AS-a.

2.2.1. Komunikacija BGP usmjernika

BGP usmjernik održava svoju tablicu usmjeravanja putem susjedskih odnosa, kao na slici 1, tako da saznaje adresni prefiks od unutarnjih BGP usmjernika putem iBGP-a ili od vanjskih susjednih BGP usmjernika putem eBGP-a. Ova dva načina rada BGP protokola razlikuju se samo po pravilima usmjeravanja, ali upravljačke poruke koje predstavljaju inkrementalne promjene su istog oblika i nazivamo ih *UPDATE porukama*. Za adresni prefiks koji je dobio primivši UPDATE poruku, BGP usmjernik izabire najbolju putanju na temelju tog i već postojećeg zapisa u svojoj tablici usmjeravanja te ažurira svoju tablicu usmjeravanja zapisom kraćeg puta (ako zapis već ne postoji u tablici, dodaje se). UPDATE porukama može se u isto vrijeme povući jedan ili više adresnih prefiksa ako postoje u tablici, a inače se takav zahtjev ignorira. Treba napomenuti da se ovdje promatra BGP usmjernike u stanju formiranih veza sa susjednim BGP usmjernicima te će se o inicijalnom stanju i automatima BGP veza govoriti u poglavlju 2.2.3.



Slika 1, Autonomni sustavi i BGP [1]

Ako govorimo o primanju UPDATE poruke izvana AS-a, odnosno od vanjskog susjednog BGP usmjernika, BGP usmjernik prosljeđuje UPDATE poruku svim ostalim unutarnjim BGP usmjernicima kako bi se na razini AS-a očuvao jedinstven pogled usmjeravanja.

Ako je UPDATE poruka stigla od unutarnjeg usmjernika, BGP usmjernik sukladno politici svog AS-a mijenja i prosljeđuje, prosljeđuje originalnu ili pak uopće ne prosljeđuje UPDATE poruku za primljeni adresni prefiks vanjskom susjednom BGP usmjerniku.

Politike usmjeravanja pojedinog AS-a se očituju u izboru najboljeg puta koji će se proslijediti susjedima. BGP usmjernici istog AS-a moraju biti potpuno povezani kako

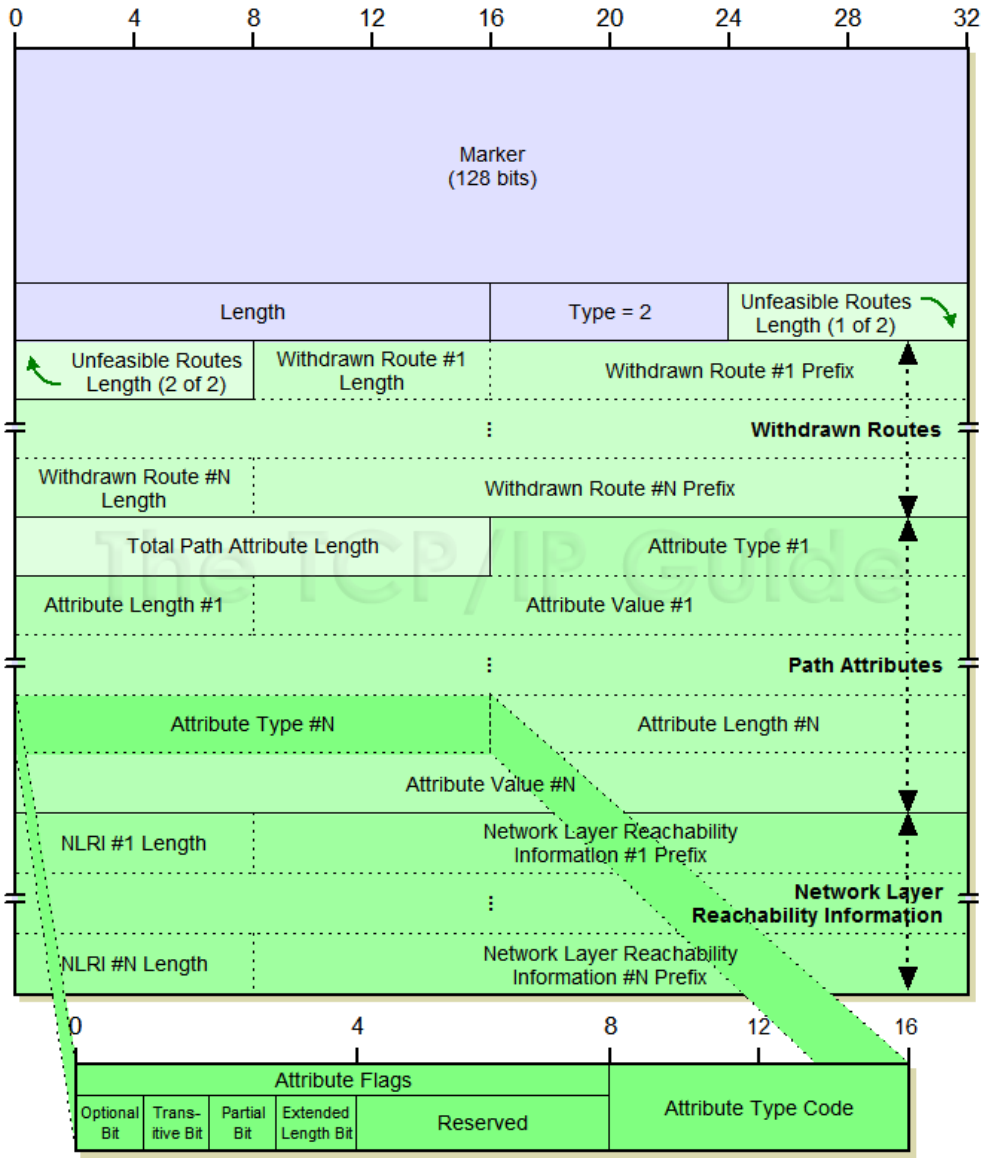
bi se informacija o novoj putanji propagirala od jednog do svih ostalih BGP usmjernika i kako bi pogled usmjeravanja u svakom trenutku bio jedinstven. Unutarnji BGP usmjernici ne prosljeđuju pristiglu informaciju o novoj putanji koju dobiju preko iBGP-a ostalim unutarnjim BGP usmjernicima iz razloga što su svi potpuno povezani te se pretpostavlja da je ta informacija već poslana i drugima pa daljnje prosljeđivanje nije potrebno. Treba napomenuti kako unutarnji BGP usmjernici ne moraju biti i uglavnom nisu direktno povezani na podatkovnom sloju, već koriste uslugu IGP-a na mrežnom sloju kako bi ostvarili svoju povezanost.

Kako bi BGP usmjernici funkcionirali, bitno je da znaju odrediti koji BGP usmjernici su im susjedi s kojima će moći komunicirati. Susjedi BGP usmjerniku su svi unutarnji BGP usmjernici istog AS-a te BGP usmjernici susjednog AS-a za čije je primanje poruka usmjernik „zadužen". Susjedski odnos se kod BGP usmjernika definira eksplicitno u njegovoj konfiguraciji. Prilikom uspostave unutarnjeg susjedskog odnosa, kod BGP usmjernika se razmjenjuju sve informacije o BGP rutama pohranjene na razini AS-a kako bi se omogućio jedinstven pogled usmjeravanja. Nakon inicijalne uspostave jedinstvenog pogleda usmjeravanja, provjere postojanosti susjedskog odnosa se vrše periodički, a daljnje promjene u BGP rutama inkrementalno u skladu sa informacijama koje BGP usmjernici prime izvana.

Dogovoreni način komunikacije BGP usmjernika obuhvaća 4 poruke:

- OPEN poruka se šalje nakon uspostave TCP veze na vratima 179 između dvaju usmjernika. Potvrdna poruka na tu poruku jest poruka KEEPALIVE kojom se potvrđuje primitak. Prilikom uspostave veze se razmjenjuju sljedeći parametri: ASN, verzija BGP-a koju koriste, BGP identifikator, najveći broj sekundi koji može proći između uzastopnih poruka KEEPALIVE ili UPDATE odnosno vrijeme zadržavanja, opcionalni parametri i duljina opcionalnih parametara.
- KEEPALIVE porukom se održava susjedski odnos između dvaju usmjernika periodičkim slanjem. U slučaju uzastopnog izostanka odgovora jedne ili druge strane, odnosno istekom vremena u polju vrijeme zadržavanja, veza se prekida.
- NOTIFICATION porukom se signalizira da je došlo do greške te se inzistira na ponovnom pokretanju veze. Primjeri grešaka uključuju istek vremena definiranog u polju vrijeme zadržavanja, primitak nepoznatog atributa ili primitak pogrešnog ASN-a.
- UPDATE poruka je temelj BGP funkcionalnosti jer se njome u inkrementima šalju nove informacije o stanju adresnih prefiksa. Te informacije nazivamo *informacije dostupnosti u mrežnom sloju* (*engl. network layer reachability information, NLRI*). Usmjernik svojim susjedima šalje jedan novi prefiks po UPDATE poruci. Istovremeno se može slati informacija o jednom novom dohvatljivom adresnom prefiksu i jednom ili više adresnih prefiksa koji se povlače jer su prestali biti dohvatljivi.

2.2.2. Format UPDATE poruke i atributi puta



Slika 2, format UPDATE poruke [12]

Namjena UPDATE poruke jest razmjenjivanje NLRI-a. Polja koja sadrži UPDATE poruka [1] mogu se vidjeti na slici 2, a važnija su:

- *Unfeasible Routes Lenght* - Ukoliko je vrijednost polja 0, *Withdrawn Routes* sekcija ne postoji u poruci.
- *Withdrawn Routes* – sekcija promjenjive duljine koja sadrži popis adresnih prefiksa (*Withdrawn Route #N Prefix*) odnosno IP prefiksa koji više nisu dohvatljivi te njihove duljine (*Withdrawn Route #N Length).*
- *Withdrawn Route #N Prefix* – IP adresni prefiks za put koji više nije dohvatljiv.

- *Withdrawn Route #N Length* – duljina polja *Withdrawn Route #N Prefix*.
- *Total Path Attribute Length* – duljina sekcije *Path Attributes*.
- *Path Attributes* – sekcija koja se sastoji od atributa puta za određene adresne prefikse u NLRI. Svaki atribut puta se sastoji od tri podatka: vrsta atributa (*Attribute Type #N*), duljina atributa (*Attribute Length #N*) i vrijednost atributa (*Attribute Value #N*).
- NLRI – sekcija koja se sastoji od popisa agregiranih putova (*NLRI #N Prefix*) ili puta (*NLRI Prefix*) koji se najavljuje za ažuriranje i na koji se odnose atributi puta. Veličina sekcije NLRI se određuje uspomoć polja *Length, Total Path Attribute Length* i *Unfeasible Routes Length*. Veličine pojedinih *NLRI #N Prefix* su zapisane u *NLRI #N Length*.

Atributi puta (*engl. path attributes*) dijele se u četiri skupine [4], a to su:

- Dobro poznati obavezni (*engl. Well-known mandatory*)
- Dobro poznati neobavezni (*engl. Well-known discretionary*)
- Izborni tranzitni (*engl. Optional transitive*)
- Izborni lokalni (*engl. Optional non-transitive*)

Svaki BGP usmjernik mora moći prepoznati dobro poznate atribute UPDATE poruke te UPDATE poruka mora sadržavati sve dobro poznate obavezne atribute ako sadrži NLRI. Svaki BGP usmjernik je također obavezan proslijediti dalje sve dobivene atribute u svojim UPDATE porukama prilikom prosljeđivanja adresnog prefiksa. Atributi puta UPDATE poruke su ORIGIN, AS_PATH, NEXT_HOP, MULTI_EXIT_DISCRIMINATOR, LOCAL_PREF, ATOMIC_AGGREGATE i AGGREGATOR. [4]

AS_PATH je dobro poznati obavezni atribut puta UPDATE poruke koji se koristi za detekciju ciklusa u usmjeravanju, primjenu politike usmjeravanja i konačan izbor rute usmjeravanja. AS_PATH u sebi sadrži niz ASN-a kojima je informacija o adresnom prefiksu putovala. Na Internetu danas nailazimo na problem veličine ASN-a koji se bilježe u AS_PATH atributu. Dok je 16-bitni ASN opće prihvaćen, 32-bitni ASN ne podržavaju svi AS-ovi te iz tog razloga AS_PATH može biti neupotrebljiv u pokušaju da AS iščita put kojim se adresni prefiks kretao.[18] Detekciju ciklusa u usmjeravanju vrši AS koji prima adresni prefiks te zaključi na temelju AS_PATH niza ASN-a da se njegov ASN nalazi u listi - u tom slučaju AS izbacuje adresni prefiks iz opticaja. Podsjetimo se, svaki atribut puta se sastoji od tri podatka ili polja: vrsta atributa, duljina atributa i vrijednost atributa. Vrijednost polja vrsta atributa za ovaj atribut puta je 2, vrijednost polja duljina atributa jednaka je broju AS-ova koje je informacija prošla, a polje vrijednost atributa sadrži popis ASN-ova kojima je informacija prošla.

NEXT_HOP je dobro poznati obavezni atribut puta UPDATE poruke koji služi usmjeravanju adresnih prefiksa. Postoje tri različita načina korištenja ovog atributa puta. Ako se UPDATE poruka šalje između dva AS-a (eBGP), tada polje vrijednost atributa poprima IP adresu BGP usmjernika koji je pošiljatelj. Ako se UPDATE poruka šalje unutar istog AS-a (iBGP), postoje dva scenarija. Prvi scenarij se odvija

ako je mreža čiji se prefiks prosljeđuje upravo mreža u kojoj se trenutno šalje UPDATE poruka i tada će opet IP adresa poprimiti vrijednost BGP usmjernika koji je pošiljatelj. Drugi scenarij jest u slučaju da je AS o čijem se prefiksu informacija prosljeđuje vanjski, onda će polje vrijednost atributa puta NEXT_HOP poprimiti IP adresu vanjskog BGP usmjernika koji je pošiljatelj i od kojega je stigla informacija o prefiksu u trenutni AS. Vrijednost polja vrsta atributa za ovaj atribut je 3.

ORIGIN je dobro poznati obavezni atribut puta UPDATE poruke, a definira porijeklo puta te ga generira usmjernik od kojeg put potječe. Njegovu vrijednost ne bi smjeli mijenjati drugi usmjeritelji. Polje vrijednost atributa može poprimiti: IGP, EGP ili *incomplete* i utječe na izbor najboljeg puta. Vrijednost polja vrsta atributa za ovaj atribut je 1. [16]

AGGREGATOR je izborni tranzitni atribut puta UPDATE poruke koji ne utječe na izbor najboljeg puta, a koristi se kod otkrivanja grešaka. Ovaj atribut sadrži ASN posljednjeg AS-a i IP-adresu BGP usmjernika koji je stvorio agregiranu rutu. Vrijednost polja vrsta atributa za ovaj atribut je 7. ATOMIC_AGGREGATE je dobro poznati neobavezni atribut puta UPDATE poruke koji omogućuje združivanje putova prema odredištu (objavljivanje više adresnih prefiksa pod jednim adresnim prefiksom). Mogu se združivati samo adresni prefiksi koji imaju iste atribute (pošto se združuju u istoj UPDATE poruci koja ima samo jedan popis atributa) te se atribut AGGREGATOR koristi kao pomoć pri ispravljanju grešaka i kao dodatna oznaka združenja putova. Vrijednost polja vrsta atributa za ovaj atribut je 6. [16]

MULTI_EXIT_DISCRIMINATOR je izborni lokalni atribut puta UPDATE poruke koji se koristi kod BGP usmjernika koji ima više vanjskih susjeda koji pripadaju istom susjednom AS-u i olakšava mu odlučivanje o prosljeđivanju poruke. Ovim se atributom provodi politika usmjeravanja odlaznog prometa pojedinog AS-a. Kada prema AS-u postoji više mogućih putova i odluka se ne može donijeti na temelju ostalih atributa, odabire se onaj put koji ima najmanju vrijednost atributa MED. Vrijednost polja vrsta atributa za ovaj atribut je 4.

LOCAL_PREF je dobro poznati neobavezni atribut puta UPDATE poruke koji koristi BGP usmjernik kako bi obavijestio unutarnje susjede o svom stupnju preferencije prema proslijeđenom adresnom prefiksu. Atribut se razmjenjuje između usmjernika istog AS-a i koristi za formiranje politike odlaznog prometa kada postoji više izlaznih putova iz AS-a (odabire se onaj put koji ima veću vrijednost atributa LOCAL_PREF). Vrijednost polja vrsta atributa za ovaj atribut je 5.

2.2.3. Rad BGP usmjernika

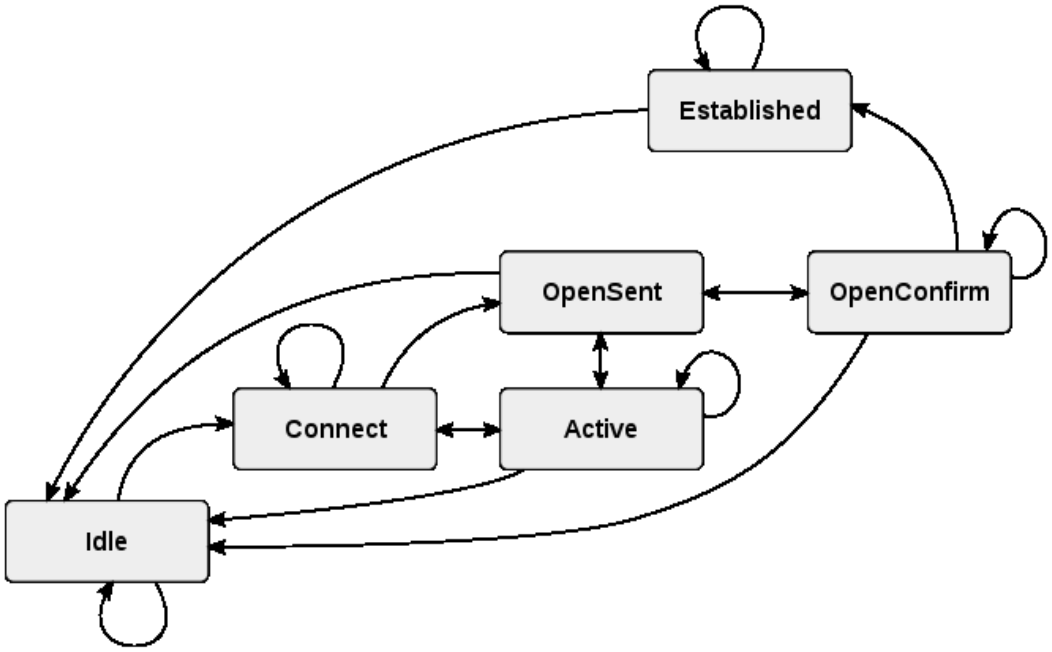
Konkretan put usmjeravanja paketa prema nekom odredištu odabire se na temelju parametara puta, dostupnosti puta i atributa puta, dodatnih pravila o prihvaćanju i propuštanju paketa te ugovora između AS-ova. Proces odluke (*engl. decision process*) se može podijeliti na dvije razine: algoritamsko i političko usmjeravanje.

Svaki usmjernik sadrži *bazu putova* (*engl. BGP Routing Information Base, RIB*) na temelju koje određuje putove usmjeravanja pojedinih paketa. RIB se sastoji od tri vrste popisa. `Adj-RIBs-In` je popis neobrađenih putova koji su primljeni od susjednih usmjeritelja, ali se uzimaju u obzir prilikom odabira puta te za svakog susjeda postoji zasebni `Adj-RIBs-In` popis. `Loc-RIB` je popis putova s lokalnim informacijama o usmjeravanju do kojih se dolazi primjenom vlastitih pravila usmjeravanja i provođenjem procesa odluke nad popisom neobrađenih ruta. `Adj-RIBs-Out` je popis putova koji se šalju susjednim usmjeriteljima slanjem UPDATE poruka. Kao i kod `Adj-RIBs-In` popisa, zasebni `Adj-RIBs-Out` popis se čuva za svakog susjeda. Informacija o politici usmjeravanja je pohranjena u PIB-u (*engl. Policy Information Base*) u svakom usmjerniku dok su podaci relevantni za algoritamsko usmjeravanje pohranjeni u RIB-u. [4]

Iako nisu obavezna, postoje pravila za dobar način podešavanja rada BGP usmjernika kojih bi se trebao pridržavati svaki AS prilikom formiranja politike usmjeravanja, a određuju rangiranje najboljeg puta na sljedeći način [4]:

- Adresni prefiks za koji nije poznat usmjeritelj u NEXT_HOP se ignorira
- Najveća težina usmjernika (nije vezana za BGP protokol)
- Najkraći AS_PATH
- Najmanji ORIGIN
- Najmanji MED
- Najveći LOCAL_PREFERENCE
- Preferiranje eBGP puta naspram iBGP putu
- Najmanja metrika do NEXT_HOP usmjeritelja

Temeljni rad BGP usmjernika, neovisan o konfiguraciji politike usmjeravanja, možemo opisati konačnim automatom sa šest stanja: *Idle*, *Connect*, *Active*, *OpenSent*, *OpenConfirm* i *Established*. Za svaku vezu koja se treba ostvariti između susjednih BGP usmjernika „stvara“ se po jedan automat stanja za svakog od njih kao na slici 3.



Slika 3, Automat BGP usmjernika [4]

Početno stanje automata BGP usmjernika jest *Idle*. U tom stanju BGP usmjernik odbija sve dolazne zahtjeve za uspostavom veze. Prilikom manualnog ili automatskog uključenja, BGP usmjernik prelazi u stanje Connect i pokušava uspostaviti spoj sa susjednim BGP usmjernicima. Prilikom uspješno uspostavljene TCP veze na vratima 179, usmjernik prelazi u stanje *OpenSen*t i šalje BGP poruku OPEN. U stanju *OpenSent* i poslavši poruku OPEN, BGP usmjernik je započeo proces rukovanja (*engl. handshaking*) sličan procesu rukovanja prilikom uspostave TCP veze. Poslavši BGP poruku OPEN, očekuje potvrdnu poruku KEEPALIVE. Primitkom poruke KEEPALIVE, BGP usmjernik prelazi iz stanja *OpenSent* u *OpenConfirm* i čeka poruku OPEN od susjednog BGP usmjernika s kojime je povezan TCP vezom, postavljajući pritom brojač vremenskog isteka. Susjedni BGP usmjernik odgovara porukom KEEPALIVE čime oba prelaze u stanje *Established* u kojem je dopušteno slati UPDATE poruke i susjedski BGP odnos je formiran.

Ako je BGP usmjernik u nemogućnosti uspostaviti TCP vezu, prelazi u stanje *Active* i pokušava još jednom uspostaviti TCP vezu – ako ne uspije, onda se vraća u stanje *Idle* i odustaje od daljnjih pokušaja. Ako u bilo kojem trenutku usmjernik pošalje ili dobije NOTIFICATION poruku, onda prelazi u stanje *Idle* čime se raskida TCP veza, a to može nastupiti ako poslana OPEN poruka nije dobra ili je nastalo zagušenje pa je brojač vremenskog isteka došao do nule.

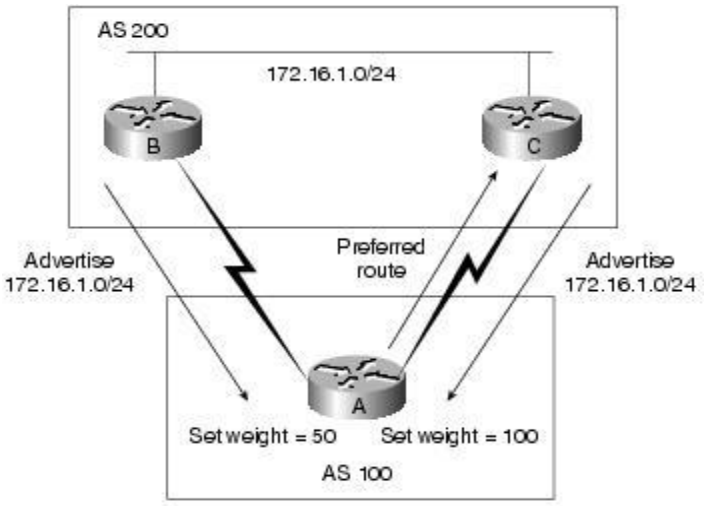
2.2.4. Primjeri rada BGP usmjernika

Prvo ćemo razmotriti primjer algoritma usmjeravanja BGP protokola koji se javlja u praksi [4]:

1. Provjerava se dohvatljivost usmjernika navedenog u atributu NEXT_HOP. Ako je dohvatljiv, provjerava se nalazi li se navedeni usmjernik u istom AS-u. Ako je i to točno, gleda se atribut LOCAL_PREFERENCE.
2. Ukoliko ima više putova za isto odredište, traži se onaj s najvećom vrijednosti u atributu LOCAL_PREFERENCE. Uzima se onaj s najvećom vrijednosti, a ako su vrijednosti iste, onda se kreće na korak 3.
3. Gledamo atribute AS_PATH, ORIGIN i MED tim redoslijedom. Traži se put sa najkraćim AS_PATH atributom, zatim put sa najmanjom vrijednosti u ORIGIN atributu, a posljednje (ako se ne može još uvijek donijeti odluka) se gleda put sa najmanjim MED atributom.

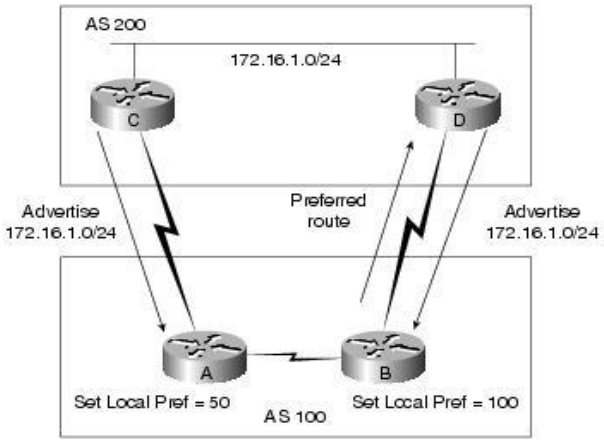
Na temelju informacija koje BGP usmjernik dobije od susjednih usmjernika (popis `Adj-RIBs-In`) i vlastitih saznanja (popis `Loc-RIBs`) te koristeći neki od algoritama usmjeravanja i podatke iz PIB-a, BGP usmjernik odredi jedan najbolji put usmjeravanja za svaki adresni prefiks (odredište) i koristi ga za usmjeravanje podataka koji su namijenjeni tom adresnom prefiksu.

Prije spomenuta težina BGP usmjernika (*engl. weight*) nije vezana za BGP protokol nego je lokalno vezana za usmjernike i ne prosljeđuje se. Prilikom odabiranja puta (slika 4), ako BGP usmjernik AS-a 100 može birati između dva susjedna BGP usmjernika u AS-u 200 za istu rutu koristeći samo njihovu predefiniranu težinu, BGP usmjernik izabire onog susjeda koji ima veću težinu.



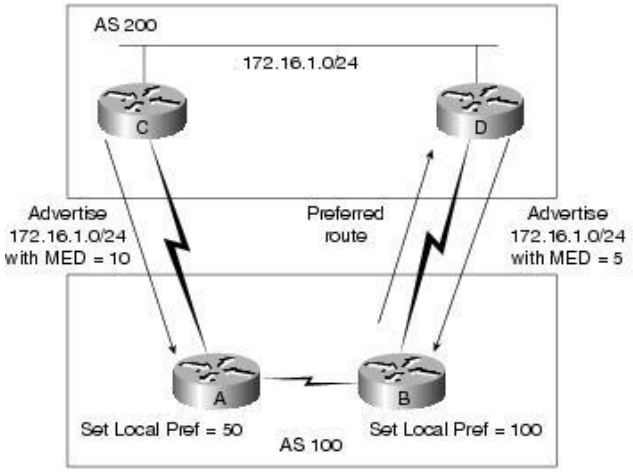
Slika 4, Težina (engl. weight) usmjernika [5]

LOCAL_PREF atribut je vezan za adresni prefiks i prosljeđuje se iBGP UPDATE porukama. U praksi se koristi kako bi se između više mogućih izlaza iz lokalnog AS-a za isti adresni prefiks odredio najbolji. Primjerice na slici 5 možemo vidjeti dva usmjernika u lokalnom AS-u 100 kako dobivaju UPDATE poruke od susjednog AS-a 200 za isti adresni prefiks 172.16.1.0/24. Usmjernik A postavlja LOCAL_PREF za sve prefikse koje dalje prosljeđuje na 50 dok usmjernik B postavlja na 100. Usmjernici A i B čine dva moguća izlaza prema AS-u 200 te se na razini pogleda usmjeravanja izabire usmjernik B jer ima veći LOCAL_PREF.



Slika 5, Izbor prema LOCAL_PREF atributu [5]

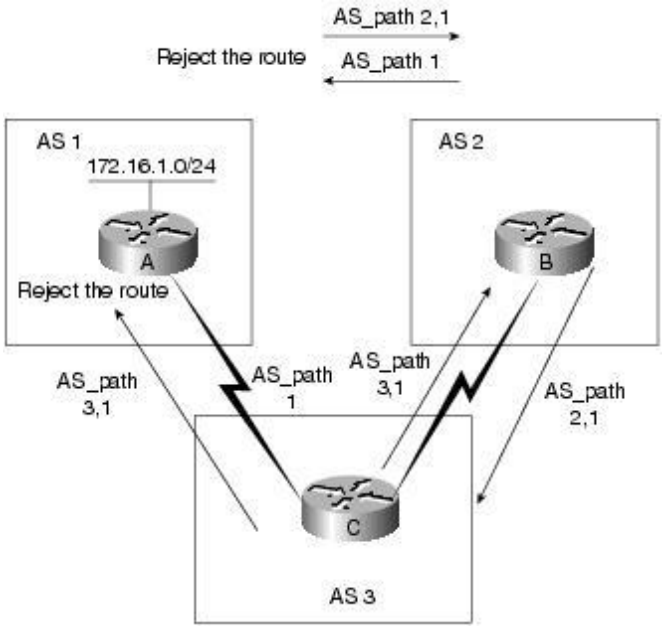
MED atribut se koristi u UPDATE porukama poslanim susjednim AS-ovima kao prijedlog izbora, ako postoji više od jednog mogućeg izbora usmjernika za taj prefiks između dva AS-a. Na slici 6, usmjernik C u UPDATE poruci prosljeđuje adresni prefiks 172.16.1.0/24 sa MED vrijednosti 10 dok usmjernik D čini to isto sa MED vrijednosti 5. Pošto je manja vrijednost MED-a povoljnija, AS 100 će izabrati put prema usmjerniku D kao prioritetni kada bude usmjeravao pakete za taj adresni prefiks.



Slika 6, Izbor prema MED atributu [5]

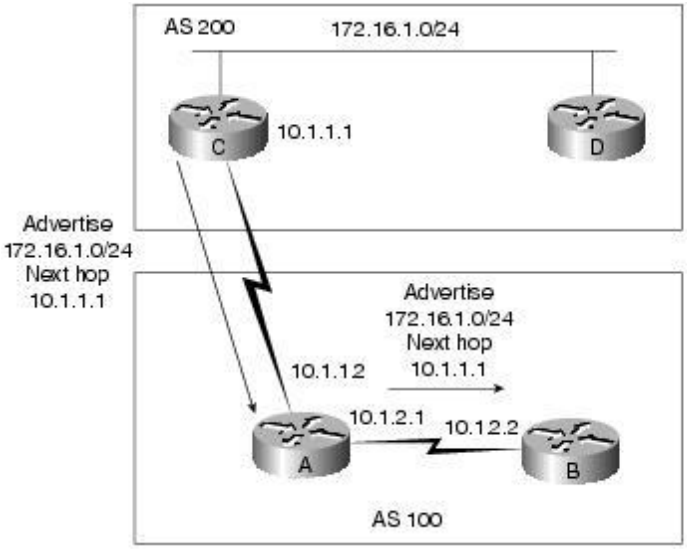
AS_PATH atribut puta UPDATE poruke sadrži popis svih AS-ova kojima je informacija o prefiksu prošla. AS_PATH atribut se koristi prvenstveno u izbjegavanju petlji koje se mogu pojaviti u usmjeravanju UPDATE poruka kao što se može vidjeti na slici 7. AS 1 objavljuje UPDATE porukom svoj adresni prefiks 172.16.1.0/24 AS-u 3 i AS-u 2 te je vrijednost AS_PATH atributa UPDATE poruke {1}.

AS 3 će prefiks pokušati objaviti UPDATE porukama prema AS 1 i AS 2 sa trenutnom vrijednosti AS_PATH {3,1} dok će AS 2 pokušati objaviti UPDATE porukama prema AS 1 i AS 3 sa trenutnom vrijednosti AS_PATH {2,1}. AS 1 će zanemariti objave jer vidi da njegov ASN već postoji u uređenoj listi, a AS 2 i AS 3 neće zanemariti objavu, ali će se odlučiti za već postojeći zapis za taj prefiks sa vrijednosti atributa AS_PATH jednakom {1} jer je kraćeg puta.



Slika 7, izbor prema AS_PATH atributu [5]

NEXT_HOP atribut puta za eBGP-a ima vrijednost IP adrese koja se koristi kako bi se došlo do rubnog BGP usmjernika C AS-a 200 koji propagira UPDATE poruku u AS 100. Za eBGP vrijedi da IP adresa predstavlja konekciju između dva AS-a. IP adresa u NEXT_HOP atributu UPDATE poruke se prenosi unutar AS-a kao na slici 8. Usmjernik C objavljuje adresni prefiks 172.16.1.0/24 UPDATE porukom sa atributom NEXT_HOP vrijednosti 10.1.1.1. Kada usmjernik A propagira dalje UPDATE porukom adresni prefiks, adresni prefiks se unutar AS-a propagira u skladu sa NEXT_HOP atributom i IGP-om koji se koristi. Dođe li do slučaja da usmjernik B ne posjeduje zapis u svojoj tablici usmjeravanja kako doći do adrese zapisane u NEXT_HOP atributu, cijela se UPDATE poruka za taj adresni prefiks odbacuje. Kako do toga ne bi došlo, bitno je da AS-ovi imaju dobro podešeno IGP-om.



Slika 8, Korištenje NEXT_HOP atributa [5]

Ovdje se treba napomenuti da se pravila korištenja NEXT_HOP atributa i zapis koji se nalazi u polju vrijednosti tog atributa mijenjaju ovisno govori li se o UPDATE porukama između BGP usmjernika C i A te A i B. Kao što je spomenuto u poglavlju 2.2.2. postoji eBGP varijanta i iBGP varijanta sa dva scenarija. U eBGP UPDATE poruci, atribut puta NEXT_HOP ima za vrijednost polja izvornu IP adresu BGP usmjernika koji je pošiljatelj 10.1.1.1. U iBGP UPDATE poruci, atribut puta NEXT_HOP ima za vrijednost polja IP adresu vanjskog BGP usmjernika koji je pošiljatelj informacije o prefiksu 10.1.1.1.

U sklopu ovog rada neće se ulaziti u greške i sigurnost BGP-a te će se u nastavku pretpostaviti da je sve što odudara od dobre prakse moguće otkriti i kazniti smanjenjem reputacije odgovornog AS-a ili više njih. U praksi gotovo nikad nije moguće točno odrediti AS koji je izvor problema te se odgovornost dijeli među AS-ovima određenog područja iz kojeg je neželjen promet potekao.

3. Trenutno stanje implementacije programskog sustava

Reputacijski sustav koji mjeri reputaciju na temelju BGP prometa programski je ostvaren u jeziku Python 2.7. U ovom poglavlju će se predstaviti temeljni moduli kao i trenutno stanje implementacije programskog sustava. Reputacijski sustav bi u primjenjivoj izvedbi trebao biti na poslužitelju koji ostvaruje susjedske odnose sa velikim brojem BGP usmjernika te bi od njih trebao skupljati UPDATE poruke. Takav bi se BGP usmjernik ponašao kao i svaki drugi uz tri iznimke:

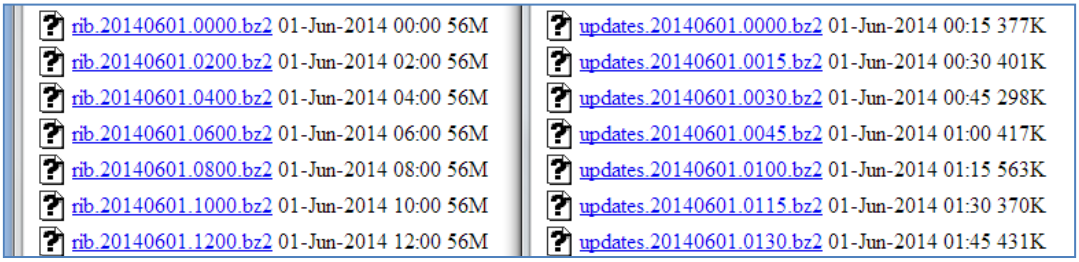
- Samo bi primao UPDATE poruke, ne bi ih prosljeđivao
- Ne bi usmjeravao promet
- Održavao bi susjedske odnose ne samo sa BGP usmjernicima susjednih AS-ova, već i sa udaljenijim AS-ovima

Takvi se poslužitelji odnosno BGP usmjernici nazivaju kolektori te ih održavaju veliki ISP-ovi, registrari i obrazovne ustanove između ostalih. Oregonsko sveučilište jedno je od takvih obrazovnih ustanova koja održava svoj vlastiti kolektor i njihovi podaci korišteni su u mjerenjima u sklopu ovog rada [3].

Dva su osnovna tipa datoteka u kojima kolektori pohranjuju informacije:

- RIB datoteke
- UPDATE datoteke

U RIB datotekama se pohranjuje cjelokupan sadržaj RIB tablica kolektora dok se u UPDATE datotekama pohranjuju sve UPDATE poruke dobivene od kolektorovih susjeda. Pohrane se vrše u pravilnim vremenskim razmacima, obično između 15 min i 20 min, gdje se RIB tablice pohranjuju u datoteku nazvanu prema točnom trenutku pohrane, a UPDATE poruke se pohranjuju u datoteku nazvanu prema trenutku početka intervala pohrane od između 15 min i 20 min.



Slika 9, RIB i UPDATE datoteke [3]

Programsko ostvarenje reputacijskog sustava sastoji se od tri modula. Modul `start.py` služi pokretanju programa te neposredno ovisi o modulu `analyzer.py`. Moduli `analyzer.py` i `core.py` temelj su funkcionalnosti i čine programsko ostvarenje reputacijskog sustava. Oba modula su neposredno ovisna o biblioteci `pybgpdump.py` i `ipaddr.py`; `analyzer.py` je neposredno ovisan o `pychart.py` i `core.py` dok je `core.py` neposredno ovisan o `dpkt.py`.

3.1. Pokretanje programa

Modul `start.py` je glavna i jedina točka ulaza u program, u njemu se vrši učitavanje parametara iz konfiguracijske datoteke `config.ini` prikazano ispisom 1 koji su potrebni za inicijalizaciju glavne klase `Analyzer` u modulu `analyzer.py`. U `config.ini` datoteci kao prikazanoj ispisom 2 mora se podesiti put do pretparsirane RIB tablice koja se želi učitati te svi konfiguracijski parametri o kojima će način učitavanja ovisiti, a to su:

- *time_window* - veličina vremenskog prozora u kojem će se računati reputacija (u minutama)
- *time_limit* – maksimalna duljina intervala u kojem će se računati reputacija (u minutama); ako se stavi 0, onda će se računati za sve update datoteke koje postoje u direktoriju
- *time_start* – vrijeme početka računanja
- *text_output* – hoće li program ispisivati izlazne tekstualne datoteke (0 ili 1)
- *verbose* – koliko detaljno će program ispisivati u konzolu (0 ili 1)
- *alpha, beta, delta* – parametri u formuli reputacije
- *selected_as* – autonomni sustavi za koje se želi da budu iscrtani u grafu
- *graph_x, graph_y* – veličina slike u kojoj se nalazi graf, u pikselima
- *analysis* – metoda analize, može biti jedna ili obje (*pref, link*)
- *profiling* – hoće li program obaviti profiliranje brzine (0 ili 1)
- *rib* – put i ime pretparsirane RIB tablice

```
#---------------------config file constants---------------------------

CONFIG_FILENAME = 'config.ini'
SECTION_NAME = 'Variables'

TIME_WINDOW = 'time_window'
TIME_LIMIT = 'time_limit'
RIB = 'rib'
TEXT_OUTPUT = 'text_output'
ALPHA = 'alpha'
GAMA = 'gama'
DELTA = 'delta'
TIME_START = 'time_start'
SELECTED_AS = 'selected_as'
GRAPH_X = 'graph_x'
GRAPH_Y = 'graph_y'
ANALYSIS = 'analysis'
ANALYSIS_PREF = 'pref'
ANALYSIS_LINK = 'link'
VERBOSE = 'verbose'
PROFILING = 'profiling'

try:
    config = ConfigParser.ConfigParser()
    config.read( CONFIG_FILENAME )
```

Ispis 1, Početak učitavanja `config.ini` u modulu `start.py`

Kako bi se program mogao pokrenuti, nužno je pravilno podesiti parametre u `config.ini` datoteci smještenoj u projektnom direktoriju. Ispis 2 predstavlja `config.ini` datoteku korištena prilikom prvog uspješnog pokretanja programa.

```
#in minutes
time_window: 15
#minutes, set to 0 if no limit needed
time_limit: 60
#"%d %m %y %H %M" - day, month, year, Hour, Minute
time_start: 27 05 14 12 00
#preparsed RIB table
rib: RIB/rib.20140527.1200.parsed
#text output needed? '1'-yes, '0'-no
text_output: 1
#verbose level, for now 0 or 1
verbose: 1
#how much will old reputation influence the new one
alpha: 0.8
#reputation is multiplied by (1-gama), than new reputation multiplied by
#gama is added to the old
gama: 0.1
#scaling reputation to an interval 0<x<1 with an inverse exponential
#function with factor delta
delta: 0.25
#AS which will be in graph, separated by comma (example: 4761, 28666,
#2685)
#selected_as: 2108
selected_as: 390
#width of graphs, in pixels
graph_x: 1024
#height of graphs, in pixels
graph_y: 768
#analysis type one or both separated by comma( pref, link )
analysis: pref
#turn on profiling (1 or 0)
profiling: 0
```

Ispis 2, `config.ini` datoteka

Datoteke s pohranjenim UPDATE porukama koje se koriste su binarne datoteke u koje kolektori pohranjuju sve UPDATE poruke dobivene od svojih susjeda u određenom vremenskom intervalu. Za pretparsiranje izvornih RIB tablica preuzetih sa web stranice koje održava Oregonskog sveučilišta koristi se *zebra-dump-parser* [19] napisan u Perl-u. Pretparsiranje je potrebno jer RIB tablice mogu doći u različitim formatima što bi onemogućilo funkcionalnost reputacijskog sustava te se na ovaj način RIB tablice svode na pogodan oblik za daljnju obradu programskim jezikom Python. Pretparsiranje raspakirane RIB datoteke traje oko 6 h.

Prilikom prvog pokretanja, u projektnoj datoteci nalazila su se sva tri modula `start.py`, `core.py` i `analyzer.py` te direktoriji `out`, `out_links`, `UP` i `RIB`. U direktoriju `RIB` je bila smještena `rib.20140527.1200.parsed` datoteka dok je u direktoriju `UP` bila smještena `updates.20140527.1200` datoteka. Učitavanje pretparsiranog RIB-a traje oko 20 min.

3.2. Analiza podataka

Modul `analyzer.py` sadrži definiciju klase `Analyzer` i konstante potrebne za njezin rad. Ispis 3 sadrži klasu *Analyzer* koja ima metodu `init()` koja služi za inicijalizaciju samog objekta. U klasi `Analyzer` također možemo naći metode `analyzePrefBindings()` i `analyzeLinkBindings()` koje predstavljaju implementaciju dvije različite analize koje možemo vršiti nad autonomnim sustavima, a to su analiza povezanosti autonomnih sustava (*link*) i analiza podataka metodom povezanosti adresnog prefiksa i izvorišnog autonomnog sustava (*pref*) te metodu `drawGraph()` koja generira `.png` datoteku grafa s podacima za svaki izabrani AS.

```
class Analyzer:

    def __init__(self, time_start, time_window, time_limit, preparsed_RIB,
                 selectedAS, text_output, graph_x, graph_y):
        #duration of window (in seconds)
        self.time_win = time_window
        self.time_start = time_start
        self.time_limit = time_limit

        self.preparsed_RIB = preparsed_RIB

        self.selectedAS = selectedAS

        self.text_output = text_output

        self.size_x = graph_x
        self.size_y = graph_y
```

Ispis 3, Metoda `init()`

Kako bi program uspješno radio, nužno je izabrati jedan ili više AS-ova za analizu podataka metodom povezanosti (*pref*) ili dva ili više AS-ova za analizu povezanosti autonomnih sustava. Problem ovakvog pristupa jest što se ne može znati koji se AS-ovi nalaze u RIB-u bez da ga se prvo učita, odnosno program će se morati pokrenuti barem dva puta – prvi puta kako bi se proanalizirao izlaz i našli ASN-ovi, a drugi puta sa izabranim AS-ovim u `config.ini` kako bi se dobilo željene grafove.

Modul `core.py` je temeljni modul koji obavlja računanje, analizu i organizaciju podataka te sadrži definicije dviju glavnih klasa:

- `PrefixPath` koja se koristi u metodi `analyzeLinkBindings()`
- `PrefixASOBinding` koja se koristi u metodi `analyzePrefBindings()`

Pomoćne klase u `core.py` modulu su:

- `Prefix` – služi za pamćenje prefiksa i njegove duljine
- `AS` – služi za pamćenje broja autonomnog sustava
- `AsPath` – služi za baratanje sa AS_PATH atributom UPDATE poruke
- `AsPrefix` – služi za pohranu i manipulaciju podataka o vezi prefiks-AS
- `AsPrefixRep` – služi za računanje reputacije
- `Print` – statična klasa koja služi za kontrolu ispisa u konzolu u ovisnosti o parametru *verbose*

Prilikom razvijanja programa, u modulu `core.py` postoji konstanta `DEBUG` čijom se postavljanjem na 1 znatno ubrzava izvođenje jer se ograničava parsiranje i učitavanje RIB tablice na samo 100 000 zapisa te tako omogućava fokus na druge dijelove programa.

3.3. Izlaz podataka

Izlaz u konzoli se jednostavno rješava i kontrolira pomoćnom klasom `Print` koja prikazana ispisom 4, a nalazi se u modulu `core.py`. Ovisno o inicijalnom parametru *verbose* (0 ili 1), detaljni ispis će se generirati ili u potpunosti zanemariti.

```
class Print():
	"""
	Static class for controlling print outs to the
	console. It depends on verbose parameter.
	"""

	level = 1

	@staticmethod
	def setVerboseLevel( num ):
		Print.level = num

	@staticmethod
	def out( string ):
		#for now its only 1 or 0, so we can put if
		if Print.level:
			print string
```

Ispis 4, Pomoćna klasa `Print`

U modulu `core.py` nalaze se klase `PrefixPath` i `PrefixASOBinding` koje se koriste u glavnim algoritmima modula `analyzer.py` - `analyzeLinkBindings()` i `analyzePrefBindings()`. U `PrefixPath` klasi ispisa 5 nalazi se metoda `FileWriteRep()` zadužena za stvaranje izlaznih `Reputations_N` datoteka u `out_links` direktoriju, odnosno za ispis i rangiranje reputacija AS-a u prozoru N.

```
def FileWriteRep(self, filename):

	f1 = open(filename, 'w')

	cnt = 1
	reporder = 100000

	for key, value in sorted(self.rep.iteritems(), key=lambda (k,v): (v,k)):
		f1.write("Order num.: %s	\tAS %s:	\t Reputation: %s\n"
				% (cnt, key, value) )

		if key==4761:
			reporder = cnt

		cnt = cnt+1

	f1.close()

	pos = 100.0

	try:
		pos=(100.0*reporder)/(cnt-1)

	except ZeroDivisionError:
		pass
```

Ispis 5, `FileWriteRep()` metoda klase `PrefixPath`

U `PrefixASOBinding` klasi se nalazi nekoliko metoda koje su zadužene za izlaz podataka, a to su `FileWritePrefInf()`, `FileWriteRepInf()` i `FileWriteRep()`. Sve tri metode stvaraju izlaz u `out` direktoriju. `FileWriteRep()` metoda prikazana ispisom 6 je zadužena za stvaranje `ASes_reputation_N` datoteka, odnosno za ispis i rangiranje AS-a u prozoru N.

```
def FileWriteRep(self, filename):

       f1 = open(filename, 'w')

       cnt = 1

       for key,value in sorted(self.asRep.iteritems(), key=lambda (k,v):(v,k)):

              f1.write( "Order num.: %s     \tAS %s:           \tReputation: %s\n"
                           % (cnt, key, value) )
              cnt = cnt+1

              if key == self.selectedAS:
                     self.selectedAS_rep_history[self.window_time] = value
                     Print.out( str("selected as:", self.selectedAS,"   time:",
                            self.window_time, "  rep:", value) )


       f1.close()
```

Ispis 6, `FileWriteRep()` metoda klase `PrefixASOBinding`

`ASes_reputation_N` i `Reputations_N` datoteke sadrže popis ukupnih reputacija po AS-ovima nakon N-tog prozora prema ispisu 7, a svaki redak se sastoji od rednog broja, ASN-a te iznosa ukupne reputacije, počevši od najgoreg prema najboljem AS-u.

```
Order num.: 2       AS 37447:      Reputation: 0.636088888889
Order num.: 3       AS 23752:      Reputation: 0.972643555556
Order num.: 4       AS 23456:      Reputation: 0.993248711111
Order num.: 5       AS 36947:      Reputation: 0.999627548444
```

Ispis 7, Isječak ispisa `ASes_reputation_8`

Uloga `FileWriteRepInf()` metode jest stvaranje `ASes_prefix_percentage_N` datoteka, odnosno generiranje ispisa nakon svakog prozora N za svaki AS koji se sastoji od ASN-a, broja prefiksa kojima je trenutni AS izvorišni, prevalencije i perzistencije. Isječak ispisa može se vidjeti po ispisu 8, a kod metode prema ispisu 9.

```
    AS-Num:    57350

    Number of Prefixes                1
    Total Active Percentage Time      0.701890989989
    Average Active Percentage Time    0.701890989989

-----------------------------------
    AS-Num:    55303

    Number of Prefixes                7
    Total Active Percentage Time      0.949944382647
    Average Active Percentage Time    0.949944382647

-----------------------------------
    AS-Num:    36874

    Number of Prefixes                4
    Total Active Percentage Time      0.696329254727
    Average Active Percentage Time    0.696329254727
```

Ispis 8, Isječak ispisa `ASes_prefix_percentage_9`

```
def FileWriteRepInf(self, filename):

       f1 = open(filename, 'w')

       for elem in self.asPrefRep.keys():

              f1.write("\n-----------------------------------\n")
              f1.write("    AS-Num:    " + str(elem) + "\n\n")

              f1.write("    Number of Prefixes                " +
                            str(self.asPrefRep[elem].numberPref) + "\n")

              try:
                     f1.write("    Total Active Percentage Time      " +
                                   str(self.asPrefRep[elem].totalActiveSum/
                                          self.asPrefRep[elem].numberPref) + "\n")

              except ZeroDivisionError:
                     f1.write("    Total Active Percentage Time     " + "0.0" + "\n")

              try:
                     f1.write("    Average Active Percentage Time    " +
                                   str( (self.asPrefRep[elem].totalActiveSumRep/
                                          self.asPrefRep[elem].numberPref) ) + "\n")

              except ZeroDivisionError:
                     f1.write("    Average Active Percentage Time   " + "0.0" + "\n")

       f1.close()
```

Ispis 9, `FileWriteRepInf()` metoda

Datoteka `RIB_Prefixes_Source_AS_information` i datoteke `Prefixes_Source_AS_information_N` sadrže ispis istog formata o objavljenim prefiksima. Dok `RIB_Prefixes_Source_AS_information` datoteka sadrži informacije direktno učitane iz RIB tablice, datoteke `Prefixes_Source_AS_information_N` sadrže kumulativne informacije o prefiksima na kraju N-tog prozora. Informacije se sastoje od trenutno promatranog adresnog prefiksa, ASN-a od izvornog AS-a, vremena kad je objavljen promatrani prefiks u N-tom prozoru, ukupnog vremena koje je prefiks bio objavljen u N-tom prozoru, broja ponovnih objava prefiksa (u slučaju povlačenja) te popisa susjednih usmjernika od kojih je ova objava stigla kao što se može vidjeti po ispisu 10.

```
                            147.28.7.1

                            208.51.134.246

                            80.91.255.62


-----------------------------------
Prefix:  1.53.176.0/20

Source AS:                  18403
Time of Activation:         1401184800.0
Total Active Time:          0
Repetition:                 1
Update from Router(s):

                             157.130.10.233

                            68.67.63.245

                            216.18.31.102
```

Ispis 10, Isječak ispisa iz `RIB_Prefixes_Source_AS_information`

Metoda `FileWritePrefInf()` koja se može vidjeti na ispisu 11 služi za generiranje informacija o prefiksima i koristi se za generiranje datoteka `Prefixes_Source_AS_information_N` i `RIB_Prefixes_Source_AS_information`.

```
def FileWritePrefInf(self, filename):

	f1 = open(filename, 'w')

	for elem in self.prefas0.keys():

		f1.write("\n-----------------------------------\n")
		f1.write("Prefix:  " + socket.inet_ntoa(elem[0])
				+ "/" + str(elem[1]) + "\n")

		for el in self.prefas0[elem]:
			f1.write("\nSource AS:           " + str(el))
			f1.write("\nTime of Activation:  "+str(el.timeOfActivation))
			f1.write("\nTotal Active Time:   " + str(el.totalTime))
			f1.write("\nRepetition:          " + str(el.repetition))
			f1.write("\nUpdate from Router(s): ")

			for e in el.listOfRouters:
				f1.write("\n              " + str(e))
				f1.write("\n")

			f1.write("\n")
	f1.close()
```

Ispis 11, `FileWritePrefInf()` metoda

Prema ispisu 12 možemo vidjeti ispis u konzoli prilikom prvog uspješnog pokretanja programa uz prethodno navedenu konfiguracijsku datoteku i `DEBUG` u modulu `core.py` postavljen na 1.

```
config loaded ok
Parsing RIB...
1.*.*.*
Finished parsing RIB
Parsing file ./UP\updates.20140527.1200
Finished computing of window 1.
Finished computing of window 2.
Finished computing of window 3.
Finished computing of window 4.
Finished computing of window 5.
Finished computing of window 6.
Finished computing of window 7.
Finished computing of window 8.
Finished computing of last window
Finished generating graph:  390_pref_rep.png
```

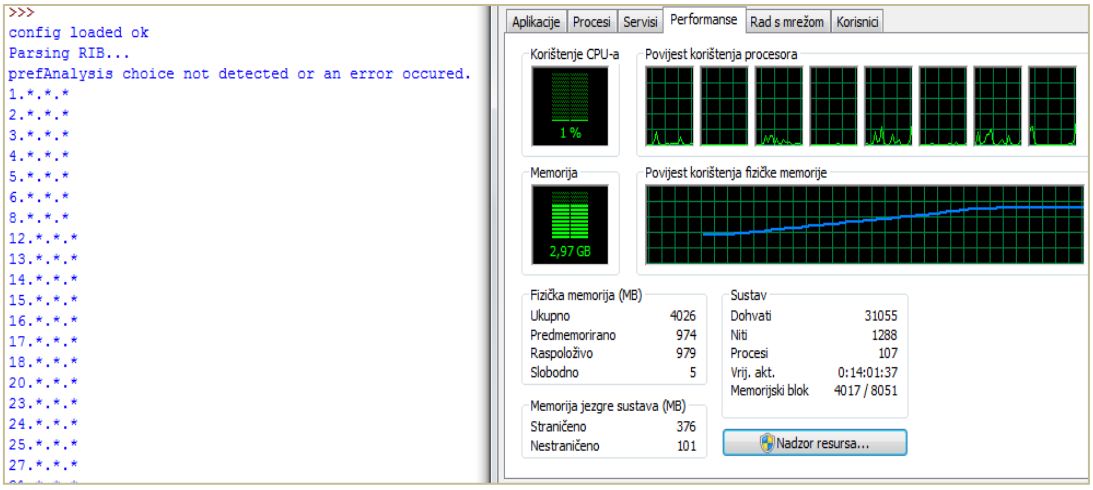
Ispis 12, Ispis u konzoli

4. Poboljšanje implementacije programskog sustava

Iako je reputacijski sustav funkcionalan, postoji niz mogućih poboljšanja koja se mogu napraviti. Osim memorije i vremena potrebnog za izvođenje, pažnja se mora posvetiti preglednosti i razumljivosti samog koda, imenovanju i dosegu varijabli u kodu, korištenju petlji, korištenju ugrađenih tipova i funkcija i tako dalje.

Prilikom prvih pokušaja da se program dovede u pokretljivo stanje primijećen je znatan nedostatak u pogledu vremena izvođenja. Pretparsiranje RIB datoteka *zebra-dumb-parser* programom može trajati i do 6 h. Ipak, ovakva situacija nije samo posljedica izvedbe parsera, već i toga da je RIB datoteka preuzeta sa kolektora za ovaj rad veličine od čak 818 MB. Olakšavajuća okolnost jest što je svaku preuzetu RIB datoteku dovoljno pretparsirati jednom kako bi se omogućila daljnja analiza podataka. Značajniji nedostatak na koji se može utjecati, a isto se tiče vremena, jest duplo učitavanje pretparsirane RIB datoteke prilikom svakog pokretanja. Pretparsirana RIB datoteka korištena u ovom radu jest veličine 1 GB te svako njeno učitavanje traje oko 30 min i na to se troši uvjerljivo najviše vremena. U `core.py` modulu postoje dvije inačice funkcije za učitavanje RIB datoteke smještene u dvije glavne klase koje se bave analizom podataka: `PrefixPath` i `PrefixASOBinding`.

Navedeni problem učitavanja RIB-a je također vidljiv i sa gledišta potrošnje memorijskih resursa. `PrefixPath` klasa osim konzumacije vremena također iscrpljuje memoriju kao što se može vidjeti na slici 10 te onemogućava upotrebu programa jer se testno računalo ruši pri iscrpljenju RAM-a veličine 4 GB za navedeni primjer RIB datoteke.



Slika 10, Memorijski problem link analize

Prve strukturne promjene napravljene su u `start.py` i `core.py` modulu te u `config.ini` datoteci. Sve apsolutne putanje ulaznih i izlaznih datoteka i `DEBUG` konstanta sada se mogu eksplicitno postaviti u `config.ini` datoteci što se vidi u ispisu 13. Kod je dodatno uređen kako bi bio pregledniji.

```
#updates input folder path
updates_folder_input: ./UP

#links method reputation output folder path and filename
links_reputation_folder_output: ./out_links/Reputations

#prefix source information output folder path and filename
prefix_source_information_folder_output: ./out/Prefixes_Source_AS_information

#prefix percentage output folder path and filename
prefix_percentage_folder_output: ./out/ASes_prefix_percentage

#pref method reputation output folder path and filename
prefix_reputation_folder_output: ./out/ASes_reputation

#Data from RIB output folder path and filename
RIBdata_folder_output: ./out/RIB_Prefixes_Source_AS_information

#Turn on debuging mode (shortens initial parsing of preparsed RIB to 100,000
#"writings")
DEBUG: 1
```

Ispis 13, Dodatak u `config.ini` datoteci

`Klasa As` prikazana ispisom 14 uočena je u modulu `core.py` koja je enkapsulirala cijeli broj i nije nudila nikakve dodatne funkcionalnosti te je u potpunosti maknuta iz programa što je rezultiralo dodatnim ubrzanjem izvođenja.

```
#base AS class
class As:

	#constructor takes integer AS number
	def __init__(self, AS):
		self.AS = AS

	#returns integer AS number
	def GetIntegerAs(self):
		return self.AS

	#returns string AS number
	def GetStringAs(self):
		return str(self.AS)

	#sets AS number, takes integer AS number as argument

	def SetAs(self, AS):
		self.AS = AS
```

Ispis 14, Klasa `As` izbačena iz programa

Prilikom detaljnije analize dvije `ReadRIB()` metode klasa `PrefixPath` i `PrefixASOBinding` u modulu `core.py` te njihovom usporedbom, ustanovljeno je da se izrazito krši „ne ponavljaj se“ (*engl. Don't repeat yourself, DRY*) načelo programiranja jer se metode razlikuju u samo dvadesetak linija koda. Ovo je ujedno i primjer kako se naizgled jednostavnim rješenjem postignutim prepisivanjem koda može vidljivo i osjetljivo udvostručiti vrijeme izvođenja programa. Optimizacija će se provesti tako da se izdvoji dio koji se ponavlja. Klase `PrefixPath` i `PrefixASOBinding` će zadržati dijelove svojih `ReadRIB()` metoda, ali će metode biti preimenovane u `AnalyzeRIB()` dok će zajednički dio postati glavna statička metoda `ReadRIB()` i biti korištena prilikom inicijalizacije `Analyzer` klase. Ove su promjene zbog iznimne međuovisnosti modula nužno značile i preinake u `analyzer.py` i `core.py` modulima. U `start.py` modulu se izmijenila `run()` metoda (Ispis 15) kako bi se omogućilo instanciranje `PrefixPath` i *PrefixAS0Binding* klasa. `Analyzer` klasi su se dodale tri nove varijable: `debug`, `links` i `pref`.

```
def run():

    # Starting point & total time calculation
    t = time.time()

    ######################################################################
    # PrefixPath and PrefixAS0Binding instances are made according to choice made in
confing.ini
    # It is assumed that new analysis methods won't be added on a regular basis...
    links = 0
    pref = 0
    for opt in analysis_list:
        if opt == ANALYSIS_LINK:
            links = PrefixPath(selectedAS, gama, delta)
        if opt == ANALYSIS_PREF:
            pref = PrefixAS0Binding(selectedAS, alpha)

    ######################################################################
    analyzer = Analyzer(time_start, time_window, time_limit, preparsed_RIB,
                        selectedAS, text_output, size_x, size_y,
                    updates_input, linksRep_output, prefInf_output, prefPerc_output,
                    prefRep_output, rib_output, debug, links, pref)

    ######################################################################
    # Depending on config.ini, one or both analysis are ran
    for opt in analysis_list:
        if opt == ANALYSIS_LINK:
            analyzer.analyzeLinkBindings(gama, delta)
        if opt == ANALYSIS_PREF:
            analyzer.analyzePrefBindings(alpha)

    # Finishing point & end of total time calculation
    print "Total time spent: %.2f  minutes" % ((time.time() - t)/60)
```

Ispis 15, Nova `run()` metoda

Ispis 16 sadrži isječak koda iz `analyzer.py` modula. U gornjem dijelu nalazi se dodatak `init()` metodi – poziv `ReadRIB()` metode koja učitava pretparsiranu RIB datoteku prilikom instanciranja klase `Analyzer`. U donjem dijelu se vidi promjena `analyzeLinkBindings()` metode (promjena `analyzePrefBindings()` metode je analogna) – na početku se umjesto instanciranja klase samo prosljeđuje instanca koju klasa `Analyzer` već sadrži.

```
       #Analysis methods
       self.pref = pref
       self.links = links

       #RIB is loaded only once and can be reused in each analysis method
       print "Parsing RIB..."
       core.ReadRIB(self.time_start, self.preparsed_RIB, self.debug, self.links,
       self.pref)
       print "Finished parsing RIB"

#---------------------------------------------------------------------------
#---------------------------------------------------------------------------
def analyzeLinkBindings(self, gama, delta):
        links = self.links
```

Ispis 16, Korištenje `ReadRIB()` metode

Dvije stare `ReadRIB()` metode razlikovale su se prema tome što rade kada se u for-petlji koja učitava liniju po liniju iz RIB datoteke detektira na početku slovo A. Na ispisu 17 se može vidjeti dio koda nove statičke `ReadRIB()` metode koji ovisno o izboru u config.ini datoteci za svaku takvu liniju pokreće prvu, drugu ili obje metode `analyzeRIB()`. Problem je bio u `analyzeRIB()` metodi klase `PrefixAS0Binding` jer se prvobitno nije uzeo u obzir doseg lokalnih varijabli koje izlazom iz svakog poziva `analyzeRIB()` metode nestaju. Tome se problemu doskočilo uvođenjem novih varijabli u klasu `PrefixAS0Binding`.

```
# ---------------------------
# Analysis control (speed up)
try:
       if prefChoice:
              # here we call prefix analyze method
              prefAnalysis.AnalyzeRIB(time_start, pref, leng, source, aspath)
       except:
              print "prefAnalysis choice not detected or an error occured."
              prefChoice = False
      ############################
       try:
              if linksChoice:
              # here we call link analyze method
              linksAnalysis.AnalyzeRIB(time_start, pref, leng, source, aspath)
       except:
              print "linksAnalysis choice not detected or an error occured."
              linksChoice = False
```

Ispis 17, Pokretanje `AnalyzeRIB()` metoda u `ReadRIB()` metodi

5. Analiza mjerenja reputacije

Prototip reputacijskog sustava temeljen na upravljačkom BGP prometu izgrađen je na temelju ideje decentraliziranog Interneta u koji se ipak treba uvesti nekakav red kako bi se poboljšala sigurnost mreža koje ga čine te cjelokupna stabilnost Interneta. Pošto je sama priroda Interneta takva da je podijeljen na autonomne sustave pod nadležnostima ISP-ova koji vode svoje privatne politike i gledaju svoj profit, ovakav model rješenja upravo pogoduje takvoj strukturi jer pruža dodatnu zaštitu ISP-ovima omogućivši im analizu upravljačkog prometa prema kojoj mogu bolje formirati svoju politiku usmjeravanja u odnosu na druge AS-ove, ali istovremeno rješava problem sve većeg broja nekvalitetnih ili malicioznih mreža na Internetu. Općim prihvaćanjem ovakvog sustava bi naposljetku takve mreže znatno manje ugrožavale stabilnost dijelova Interneta. Glavni način na koji bi ISP-ovi (iz perspektive svojih autonomnih sustava) promatrali i rangirali ostale AS-ove od interesa bila bi reputacija.

Reputacija je vrijednost koja se može dobiti raznim metodama analize upravljačkih BGP podataka. U trenutnom modelu reputacijskog sustava postoje dvije implementirane analize, a to su analiza povezanosti autonomnih sustava (*link*) i analiza podataka metodom povezanosti adresnog prefiksa i izvorišnog autonomnog sustava (*pref*) koje su kratko spomenute u poglavlju 3.2. Analiza podataka.

Analize podataka odvijaju se kroz vremenske prozore tijekom čijih trajanja se bilježe sve nepostojanosti veza. Kod analize povezanosti autonomnih sustava, svaka nepostojanost veza između autonomnih sustava se kažnjava. Kod analize povezanosti adresnog prefiksa i izvorišnog autonomnog sustava, svaka nepostojanost veze dovodi do kažnjavanja izvorišnog autonomnog sustava.

Ukupna se reputacija na kraju prozora za metodu analize povezanosti AS-a dobiva kao kombinacija prijašnje reputacije i reputacije za promatrani prozor koristeći faktor zaboravljanja *gama* koji se podešava u `config.ini` datoteci. Reputacija u prozoru N računa se na temelju formule na slici 11.

$$r_N = (1 - \gamma) \cdot r_{N-1} + \gamma \cdot R_N$$

Slika 11, Model izračuna reputacije za N-ti prozor [1]

U detalje izračuna same vrijednosti reputacije R za N-ti prozor na temelju grešaka BGP-a se neće ulaziti u sklopu ovoga rada.

U sklopu ovog rada moguće je analizirati izmjerene reputacije autonomnih sustava s obzirom na `updates.20140527.1200` i `rib.20140527.1200` datoteke preuzete sa kolektora Oregonskog sveučilišta. Svi AS-ovi koje je moguće izabrati u `config.ini` datoteci radi analize, predstavljeni su prema svojim ASN-ovima. Za potrebe ovog poglavlja, analizirat ćemo mjerenja prikupljena nad AS-om 56203. Radi se o australskom autonomnom sustavu „BIGRED—NET-AU Big Red Group“ koji posjeduje prefikse 1.0.4.0/22 te 103.2.176.0/22 [14].

```
Order num.: 497     AS 45528:     Reputation: 0.999999488
Order num.: 498     AS 45996:     Reputation: 0.999999488
Order num.: 499     AS 56120:     Reputation: 0.999999488
Order num.: 500     AS 56203:     Reputation: 0.999999488
```

Ispis 10, Dio ispisa iz `ASes_reputation_9`

Prema provedenim mjerenjima i zapisu u `ASes_reputation_N`, reputacija tog autonomnog sustavu na kraju zadnjeg prozora jest 0.999999488 te je jedna od najboljih zabilježenih reputacija u tom uzorku. Autonomni sustav identificiran prema najgoroj reputaciji je bio AS 27064 reputacije 0.000889877641824. U sklopu ovog uzorka, detektirano je četiri prefiksa AS-a 56203 koji su neprestano aktivni što se može vidjeti po ispisu 18.

```
AS-Num:    56203

    Number of Prefixes                4
    Total Active Percentage Time      1.0
    Average Active Percentage Time    1.0
```

Ispis 19, Dio ispisa `ASes_prefix_percentage_9`

U ispisu `RIB_Prefixes_Source_AS_information` možemo naći prefikse 1.0.6.0/24, 1.0.7.0/24, 1.0.4.0/24 i 1.0.5.0/24 koji imaju AS 56203 postavljen kao svoj izvor te vidimo da ti prefiksi odgovaraju podacima preuzetim sa Interneta [14] i po ispisu 19. Iako je sustav bio u mogućnosti generirati grafičke prikaze, u sklopu ovog rada testiranje se vršilo na Windows 7 OS-u te je ustanovljeno da generiranje grafova korištenjem *PyChart-1.39* biblioteke nije moguće.

6. Zaključak

Problem današnjeg Interneta nije u njegovoj decentraliziranosti, već u negativnim posljedicama takve strukture. Temeljna ideja koju ovaj rad slijedi jest pronaći način kako poboljšati kvalitetu i sigurnost Interneta čuvajući pritom slobodu i pravo na privatnu politiku ISP-ova. Prototip reputacijskog sustava na temelju BGP upravljačkog prometa može se koristiti od strane svakog ISP-a kako bi se što bolje formirala politika usmjeravanja prema autonomnim sustavima čiji ISP-ovi ne drže do prometa koji stvaraju. Na taj bi način svaki ISP bio svojevrstan promatrač koji bi kažnjavao loše autonomne sustave filtrirajući njihov dolazni i odlazni promet u sklopu svoje privatne politike te bi time poboljšao kvalitetu usluge koju pruža svojim korisnicima, ali i kvalitetu i sigurnost samog Interneta.

U sklopu ovog rada predstavilo se usmjeravanje između autonomnih sustava odnosno usmjeravanje korištenjem BGP-a i proanalizirala se implementacija prototipnog sustava za procjenu reputacije autonomnih sustava temeljeno na praćenju BGP prometa. Također se obavila analiza mjerenja reputacije i postiglo se poboljšanje u strukturi i vremenskoj složenosti programa.

7. Literatura

[1] Dejdar T, *Određivanje reputacije autonomnih sustava temeljeno na praćenju upravljačkog prometa rubnih usmjernika*, diplomski rad, Fakultet elektrotehnike i računarstva u Zagrebu, Zagreb, 2011

[2] Lovrek I, Matijašević M, Ježić G, Jevtić D, *Komunikacijske mreže*, Fakultet elektrotehnike i računarstva u Zagrebu, Zagreb, 2013

[3] Route Views Project Page
URL: http://www.routeviews.org (25/5/2014)

[4] Centar informacijske sigurnosti, *BGP protokol*, 2011
URL: http://www.cis.hr/files/dokumenti/CIS-DOC-2011-03-006.pdf

[5] Cisco, *Border Gateway Protocol*
URL:http://docwiki.cisco.com/wiki/Border_Gateway_Protocol (13/7/2013)

[6] Wikipedia, *Routing table*
URL: http://en.wikipedia.org/wiki/Routing_table (15/4/2014)

[7] Technopedia, Janssen C, *External Border Gateway Protocol*
URL: http://www.techopedia.com/definition/26865/external-border-gateway-protocol-ebgp (20/4/2014)

[8] Wikipedia, *Border Gateway Protocol*
URL: http://en.wikipedia.org/wiki/Border_Gateway_Protocol (30/3/2014)

[9] Rekhter Y, Li T, Hares S, *A Border Gateway Protocol 4 (BGP-4)*, The Internet Engineering Task Force (IETF), RFC 4271
URL: http://tools.ietf.org/html/rfc4271 (1/1/2006)

[10] Python v2.7.7 documentation
URL: https://docs.python.org/2/index.html (2/6/2014)

[11] Wikipedia, *Tier 1 network*
URL: http://en.wikipedia.org/wiki/Tier_1_network (12/3/2014)

[12] BGP UPDATE message format
URL: http://www.tcpipguide.com/free/diagrams/bgpupdateformat.png (25/5/2014)

[13] Lutz M, *Learning Python 5th edition*, O'Reilly Media, Sebastopol, 2013

[14] Internet Storm Center
URL: https://isc.sans.edu/asreport.html?as=56203 (9/6/2014)

[15] Andersson L, Davies E, *Report from IAB workshop on Unwanted Traffic*, The Internet Engineering Task Force (IETF), RFC 4948
URL: http://tools.ietf.org/html/rfc4948 (9/10/2006)

[16] NETCERTS.NET, *BGP Path attributes*
URL: http://netcerts.net/bgp-path-attributes-and-the-decision-process/ (21.5.2010)

[17] Fabris M, *AS Reputation code*, seminarski rad, Fakultet elektrotehnike i računarstva u Zagrebu, Zagreb, 2013
URL: https://github.com/sgros/asreputation-mirko-fabris

[18] Birhanu D, *BGP Techniques for ISPs*
URL: https://www.nanog.org/sites/default/files/tuesday.tutorial.birhanu.bgp101.48.pdf
(7/2/2014)
[19] d'Itri M, *Zebra-dumb-parser*
URL: https://github.com/rfc1036/zebra-dump-parser
(12/3/2013)

Sažetak

U ovom radu je opisan BGP protokol koji se koristi za usmjeravanje među autonomnim sustavima na Internetu. Predstavljena je ideja reputacije kao način mjerenja kvalitete pojedinih autonomnih sustava te je analiziran prototipni reputacijski sustav koji radi procjenu reputacije autonomnih sustava na temelju praćenja upravljačkog BGP prometa. Većina analize praćena je primjercima koda programa, ispisima izlaza i primjerima potrebnih ulaznih datoteka. U praktičnom dijelu se vršio postupak optimizacije reputacijskog sustava u kojem su se proučavala i radila poboljšanja s obzirom na vremensku i prostornu složenost. Na kraju su provedena mjerenja radi provjere funkcionalnosti samog reputacijskog sustava te analiza mjerenja u skladu sa podacima koji su dostupni na Internetu.