eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Informatika Ingeniaritzako Gradua

Konputagailuen Ingeniaritza

Gradu Amaierako Lana

Monero kriptotxanpon-sarearen azterketa Levin protokoloa erabiliz

Egilea

Iñigo Pérez

2021

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Informatika Ingeniaritzako Gradua
Konputagailuen Ingeniaritza

Gradu Amaierako Lana

Monero kriptotxanpon-sarearen azterketa Levin protokoloa erabiliz

Egilea

Iñigo Pérez

Zuzendariak
Olatz Pérez de Viñaspre
Jose A. Pascual

Laburpena

Proiektu honetan, Monero kriptotxanpon-sareak nola funtzionatzen duen ikertu da, sareko nodoak sinkronizatuta mantendu daitezen, automatikoki sortzen den sare birtuala eta parekoen arteko komunikazioa ahalbidetzen duen aplikazio-mailako Levin protokoloa aztertuz. Hau da, Levin protokoloak zehazten dituen hainbat mezu erreplikatu eta bidali dira, erantzun bakoitzaren informazioa gordez eta datu horiek modu txukunean adieraziz. Izan ere, prozesu bakoitzak lortutako informazioa fitxategietan idazteaz gain, mapamundi baten bidez, nodo bakoitza bere koordenatuetan kokatu da. Gainera, sagua mapako puntu baten gainean jartzerakoan, puntu horretan dauden nodoen informazio laburra (bakoitzaren IP helbidea eta horrek bidalitako transakzioak) ikus daiteke.

Garatutako tresnekin, guztira 36553 nodo (IP helbide desberdin) aurkitu dira Monero sarean, eta 48277 transakzio jaso dira horietako batzuetatik, baina geroz eta denbora gehiago egon exekuzioan, orduan eta transakzio gehiago jasoko dira. Halaber, nodoak mapan kokatu heinean, motelagoa izango da programa horren exekuzioa. Lortutako informazioarekin, sarearen propietateak eta lan-kargaren banaketa aztertu da.

Erabiltzailearen pribatutasuna bermatzea helburu izanik, dibisa kriptografikoa erabiltzen den esparruak kontuan hartuta emaitzen analisi zehatza egin da. Izan ere, mapan ikus daitekeen kokapenak ez du zertan Monero erabiltzaile batena izan; hau da, pribatutasun zerbitzu bat eskaintzen duen nodo bat izan daiteke mapan kokatutakoa: *The Onion Router* (Tor) sareko irekiera nodo bat, adibidez. Nabigatzaile horrek hainbat erabiltzaile modu anonimoan internet sarean komunikatzea ahalbidetzen du, eta ezkutuko zerbitzuak atzi daitezkeenez, Monero esparru horietan erabilia dela pentsa daiteke.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1. KAPITULUA

Sarrera

Azken urteetan, diru digitalaren bilakaera hainbat esparrutan antzeman ahal izan da. Zehazki, Bitcoin-en sorrera 2009an izan ondoren, kriptotxanpon asko sortu dira (askotan hura oinarritzat edukiz), bakoitza berezitasun eta helburu desberdinekin. Gaur egun, 8400 dibisa kriptografiko existitzen direla estimatzen da. Diru digitalaren trukea posible egiteko, zifratze-teknikak erabiltzen dira, eta kostu konputazional altua duten ebazpenak bilatzen dira (PoW, ingeleseko *Proof of Work* akronimotik) edo beste kasu batzuetan partehartzearen froga (PoS ingeleseko *Proof of Stake* akronimotik), kriptotxanponaren ahalmena eta edukiera handitzeko. Hala ere, azken mekanismo honekin, sareko segurtasunaren ardura zentralizatu egiten da inbertsio handiena duten erabiltzaileen artean, eta frogaren kostu konputazionala txikiagoa denez, sarearen eskalagarritasuna handiagoa izango da.

Monero sarean, Levin (parekoen arteko komunikazioa ahalbidetzen duen) aplikazio-mailako protokoloa erabiltzen da. Hortaz, transakzioen egiaztapena posible egiteko, transakzioak gordetzeko balio duten blokeen sorrera eta bestelako informazioa modu eraginkor eta anonimoan partekatu ahal izateko, *peer-to-peer* edo parekoen arteko sarea (P2P) sortzen da. Garrantzi handikoa da sarearen kudeaketa eta sinkronizazioa nodo guztien artean egiten dela, eta bloke berrien sorrera RandomX PoW algoritmoaren bidez, sarea deszentralizatuta mantenduz. Horrez gain, *ring signatures* teknologia erabiltzen da transakzioak hainbat erabiltzaileen artean egiaztatzeko eta transakzioen emaile eta hartzailea ezkutuan mantendu ahal izateko.

Honako proiektuan, erabiltzailearen pribatutasuna eta anonimo izatea helburu nagusitzat duen dibisa kriptografiko ezagunenak sortzen duen sare birtuala aztertuko da: Monero kriptotxanpon-sarea, hain zuzen ere. Gainera, azken hau ez da Bitcoin-en gainean egindako aldakuntza bat, eta hortaz, desberdintasun handiak dituzte. Horrez gain, Monero proiektua kode irekikoa denez, inplementazioan nola funtzionatzen duen xehetasun osoarekin azter daiteke.

Beraz, Monero dibisa kriptografikoaren sarean bidaltzen diren mezuak atzeranzko ingeniaritza erabiliz aztertu ondoren, mezu horiek sareko nodoei bidaliko zaizkie informazioa lortzeko asmoarekin. Mezu hauek bidaliz, nodo bakoitzaren parekoen zerrenda (gutxi gorabehera 250 IP helbidez osatuta dagoena) eskatuko da, sarera konektatuta jarraitzen duten aztertuko da eta azkenik, atzigarri dauden horietatik transakzio berriak jasotzen ahaleginduko da. Gainera, lortutako nodoak mapan kokatuko dira munduan zehar duten banaketa, jasotako transakzioekin batera analizatu ahal izateko.

2. KAPITULUA

Proiektuaren helburuak

Kapitulu honetan, proiektu honek dituen helburuak eta lortu beharreko emaitzak zeintzuk diren adieraziko dira.

Helburu nagusia Monero dibisa kriptografikoaren azterketa eta zehatzago, era automatikoan sortzen eta kudeatzen den sarearen azterketa aurrera eramatea da. Moneroren funtzionamendua dela eta, transakzioen egiaztapena eta erabiltzaile bakoitzaren pribatutasuna eta anonimotasuna mantentzearen lana erabiltzaileek soilik egiten dute, inongo hierarkiarik gabe nodo horien artean; sareko nodo guztiak maila berean daudenez, ekintza berdinak izango dituzte. Sortzen den sare mota hori (P2P) aztertuko da, Levin Moneroren barneko protokoloak deskribatzen dituen mezuak jasoz eta bidaliz.

Moneroren sareari buruz, proiektuaren garapenean lortu nahi den informazioa honakoa da: mundu mailan zer nolako banaketa jarraitzen duen eta zenbateraino dagoen zentralizatuta. Izan ere, sarearen ezaugarriak direla eta, "% 51eko erasoa"ren biktima izan daiteke. Erakunde batek sarearen konexio gehienak mantentzen dituenean gerta daiteke, sarearen segurtasuna eta transakzioen baliozkotasuna arriskuan jarriz, erabaki hori gehiengoaren menpe dagoelako, eta erakunde horrek izango luke gehiengo hori. Hortaz, nodoek lan-karga antzekoa duten kontuan hartu beharko da, baita nodo gutxi batzuk lan-karga asko jasaten duten ere.

Gainera, proiektu bat, graduarekin lortutako ezagutzekin aurrera eramateko gaitasuna lortzea, zeharkako helburua izango da. Proiektuaren kudeaketatik hasita, garapena eta azkenik dokumentazioarekin lortutakoa kontuan hartuz.

Zehazki, proiektu honen helburuak ondorengoak izango dira:

- Moneroren sarearen ezaugarriak aztertzea
- Levin protokoloa atzeranzko ingeniaritzarekin ulertzea eta erabiltzen ikastea
- Sareko nodoen mundu mailako banaketa eta zentralizazio-maila analizatzea
- Graduarekin erlazionatutako ezagutzak proiektu praktikoan aplikatzea

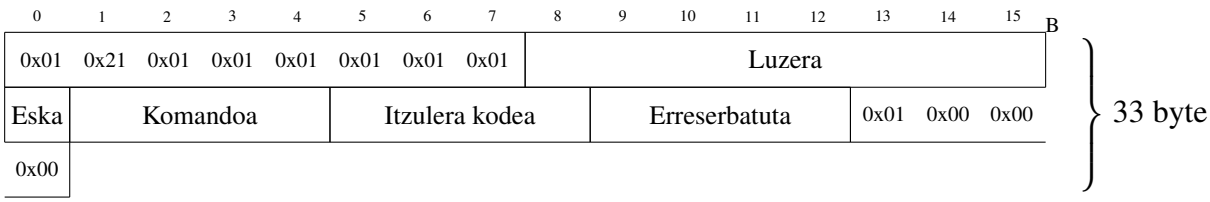
3. KAPITULUA

Levin protokoloa

Moneroren sarean zehar informazioa parekoen artean bidaltzeko, aplikazio-mailako Levin protokoloa erabiltzen da, sinkronizazioa modu arrakastatsuan betetzea ahalbidetzen duena. Bidali daitekeen informazioaren arabera, mezu desberdinak sortzen dira, eta horiek eskaerak, erantzunak eta jakinarazpenak izan daitezke. Komunikazioa segurua eta akatsik gabekoa izan behar da, eta horregatik, garraio-mailan TCP-ren gainean egiten da transmisioa, UDP ez da egokia kasu honetarako.

Levin protokoloa betetzen duten mezuak 33 byte-ko goiburukoaz eta tamaina aldakorreko datuez osatuta dauden egiturak dira. *Payload* edo gorputzaren tamaina mezuaren goiburukoan zehaztuta egongo da, maximo lehenetsia 100 MB-ekoa izanik. Modu berean, zer informazio mota bidaltzen edo eskatzen den adierazten da goiburuko honekin.

Gainera, mezuen goiburukoan hasierako eta bukaerako karaktereak bidaltzen dira, beti berdinak dira eta muga horiek argi bereizteko balio dute. Hurrengo irudian ikus ditzakegun atalek osatzen dute goiburukoa, Monero proiektuan dokumentatuta dagoen bezalaxe:



3.1 Irudia: Levin goiburukoa: ikuspegi orokorra

Hala ere, github-eko Monero proiektuaren dokumentazioaren hasieran, sarean bidaltzen den informazio guztia zehaztu ez dela argitzen da, denbora izanez gero ikerketa hori egitera gonbidatuz.

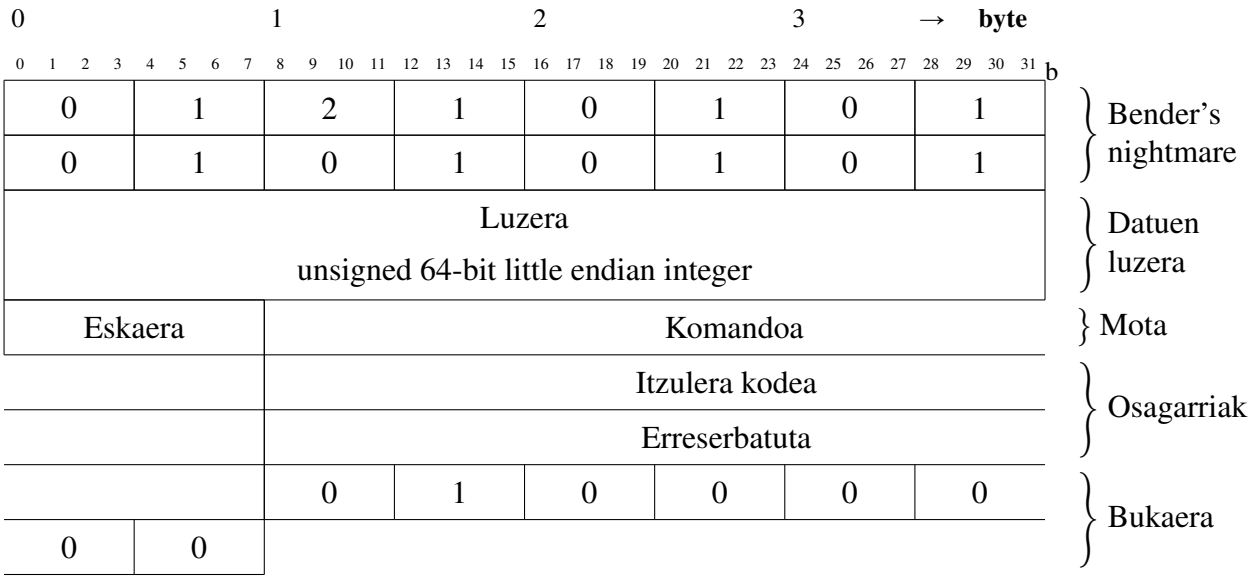
Horregatik, Monero sarera konektatuta egonda, trafikoa harrapatuz eta datu bitar horiek notazio hamaseitarrera eta ASCII kodeketara bihurtuz, mezu bakoitzean bidalitako informazioa lortu, eta atzeranzko ingeniaritza aplikatuz, Monero sarearen protokoloa ulertu eta barneratu ahal izan da. Hainbat script-en exekuzioa aztertuz eta atzeranzko ingeniaritza aplikatuz, Levin protokoloa gehiago zehaztea lortu da.

Proiektu hau garatzeko erabili den sistema eragilea UNIX familiakoa izan denez, ondorengo komandoak erabili dira informazio baliagarria lortzeko script-ak programatuz:

- `tcpflow` : TCP protokoloaren gainean bidaltzen den trafikoa harrapatzeko tresna.
- `hexdump` : Jasotako byteak hamaseitarrera eta ASCII kodeketara bihurtzeko tresna.
- `grep, sed, cut, awk...` : Testu-tratamendurako komandoak.

3.1 Goiburukoa

Atal honetan, xehetasun handiagoarekin ikusiko dugu Levin goiburukoak adierazten duen informazioa eta egiturari dagokionez nolakoa den:



3.2 Irudia: Levin goiburukoa: xeheago

Mezu guztien goiburukoak **sinadura** edo karaktere-segida berarekin hasten dira (*Bender's Nightmare* deritzaion segidarekin) 8 byte erabiliz, 3.2 irudian ikus daitekeen bezalaxe.

Ondoren, mezuaren datuen **luzera** (byte kopurua) ematen da, zeinurik gabeko 8 byte-ko egitura batean eta *little endian* ordenean. Hortaz, barneko balioa pisu txikieneko byte-arekin hasi eta pisu handieneko byte-arekin bukatzen da. Aipatzekoa da luzerak ez duela goiburukoa barne hartzen, datuen luzera soilik. Izan ere, goiburukoak beti izango ditu 33 byte baina datuen luzera aldakorra izango da mezuaren arabera eta baita mezu bakoitzean bidaltzen den informazioaren arabera.

Egitura honetan adierazi daitekeen gehienezko luzera 2^{64} bytekoa (18 EB) da, baina protokoloak zehazten duen moduan, luzera ez da 100 MB baino handiagoa izango. Hala ere, protokoloak sarean zatitutako mezuak bidali daitezkeela aipatzen du, baina gure helburuak lortzeko ez dugu mota horretako mezurik bidali beharko. Gainera, aztertutako mezuetan ez da halako mezurik jaso.

Adibidez, luzera bezela 17 04 00 00 00 00 00 $00_{(16)}$ kode hamaseitarra jasotzen bada, hurrengo bihurketa egin beharko da zenbaki hamartar bezala interpretatzeko: *big endian* ordenera pasa (04 17), zifra hamaseitar bakoitzari bere pisua biderkatu eta emaitza horiek batu.

$$\mathbf{0} * 16^3 + \mathbf{4} * 16^2 + \mathbf{1} * 16^1 + \mathbf{7} * 16^0 = 1024 + 16 + 7 = \mathbf{1047}_{(10)} \text{ byte}$$

Eskaera eremuak 0 balioa izango du erantzuna edo jakinarazpena den kasuetan, eta 1 ordea, mezua eskaera bat bada. Informazio hau byte batekin adierazten da, nahiz eta aldatzen dena bit bakarra izan.

Komando zenbakiarekin, mezuaren egitekoa zein den eta partekatu beharreko informazioa zein den zehazten da. Une hauetan, hainbat mezu mota inplementatu gabe geratu dira Moneroren kodean, baina hala ere ☒ zeinuarekin aipatuko dira. Hauek dira Moneroren sarean bidaltzen diren mezu mota desberdinak:

Komandoak			
Eskaera eta erantzunak		Jakinarazpenak	
Zenbakia	Deskribapena	Zenbakia	Deskribapena
1001	Konexioa ezartzeko mezua (*Handshake*)	2001	Egitura nagusian bloke berri baten sorreraren jakinarazpena (*New Block*)
1002	Maiztasun jakin batekin, sinkronizatuta jarraitzeko (*Timed Sync*)	2002	Transakzio berrien jakinarazpena (*New Transactions*)
1003	Parea sarean atzigarri dagoen jakiteko mezua (*Ping*)	2003	(*Request Get Objects*) ☒
1004	(*Stat Info*) ☒	2004	(*Response Get Objects*) ☒
1005	(*Network State*) ☒	2006	Egitura nagusiaren eskaera (*Request Chain*)
1006	Nodoaren identifikadorea partekatzeko mezua (*Peer ID*)	2007	Egitura nagusiaren gainean egindako eskaeraren erantzuna (*Response Chain Entry*)
1007	Konexioa ezartzerakoan bidaltzen den mezua (*Support Flags*)	2008	Egitura nagusia modu eraginkorrean partekatzeko mezua (*New Fluffy Block*)
		2009	(*Request Fluffy Missing TX*) ☒

3.1 Taula: Levin mezuen sailkapena

Levin mezuen portaera ez dago zehazki deskribatuta Moneroren dokumentazioan, eta halaxe aitortzen da dokumentuaren hasieran: dokumentuak ez du sarean zehar bidaltzen den informazio guztia zehazten. Probak eginez komunikazioa aztertzeaz gain, iturburu-kodean programatuta dagoena aztertu da mezu horiek zertarako erabiltzen diren jakin ahal izateko. Horrez gain, 5 mezu mota inplementatu gabe daudela aztertu da, 3.1 taulan gurutze gorriarekin (☒) markatuta daudenak, hain zuzen ere.

Itzulera kodea 4 byte-ekin adierazten da, *little endian* ordenean irakurrita eta 0 izango da mezua eskaera bat den kasuetan, Levin protokoloaren dokumentazioaren arabera.

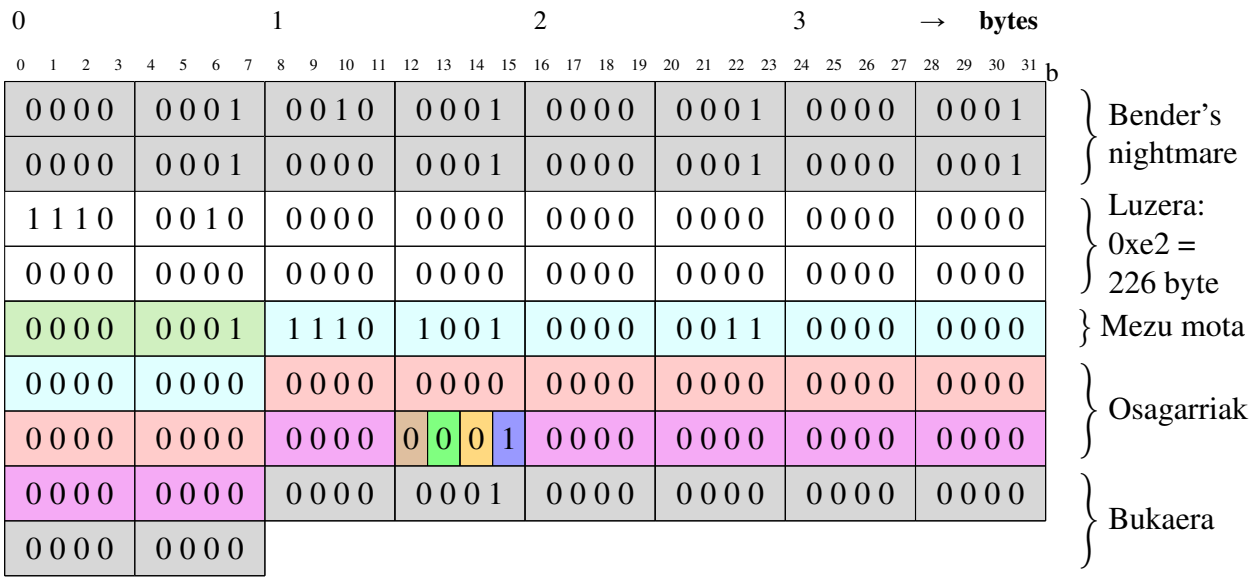
Hala ere, komunikazioa aztertu denean, 1003 mezuaren eskaerak erantzuna lortu ahal izateko itzulera kodean 1 balioa hartu behar duela ikusi da. Beraz, esan daiteke github-eko Monero proiektuan Levin protokoloa ez dagoela oso zehatz dokumentatuta.

Erreserbatutako balioa *little endian* ordeneko 4 byte-etan adierazten da, eta pisu txikieneko byte-an ondorengo 4 bitak (pisu txikienekoak) aktibatu daitezke mezuaren funtsaren arabera:

- Q: aktibatuta mezua eskaera bada.
- S: aktibatuta mezua erantzuna bada.
- B: aktibatuta zatitutako mezu baten hasiera bada.
- E: aktibatuta zatitutako mezu baten bukaera bada.

Amaitzeko, mezu guztien goiburukoek **bukaera** eremuan ikus daitekeen bit-segida 1 balioarekin erabiltzen dute (*little endian* ordenean), eta jarraian datuak bidaltzen hasten da, goiburukoak zehazten duen moduan.

Hurrengo irudian Levin pakete baten goiburukoa atal horietan sailkatuta ikus dezakegu, konexioa ezartzeko eskaera egiteko bidali beharreko informazioa, hain zuzen ere:



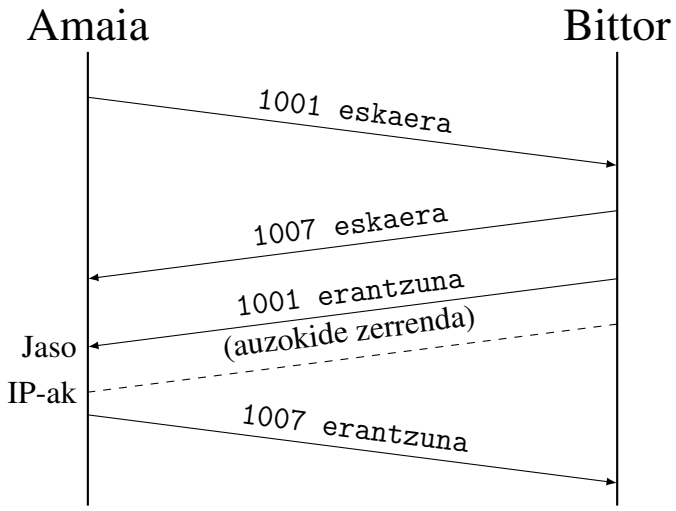
3.3 Irudia: Levin goiburukoaren adibidea (*handshake request*)

Irudian (3.3) adierazitako informazioa 1001 (e9 03_{16} *little endian*) mezuaren goiburukoak duena da eta eskaera bat dela ikus daiteke, eskaera eremuan 1eko balioa dagoelako eta erreserbatutako lehen byte-an Q bit-a gaituta dagoelako. Byte bakoitzaren barnean pisu handieneko bita ezkerreko mugan dagoena da, byte-ei dagokionez berriz, pisu handieneko byte-a eskuinekoa da *little endian* ordena jarraitzen duen kasuetan.

3.2 Bidalketa prozesua

Levin mezuaren goiburukoak zehaztu duen informazioarekin, jarraian mezuaren datuak jasotzeko prestatu egiten da; hau da, datuen tamaina jakin daitekeenez, jaso beharreko byte kopurua mugatuko da.

Gainera, pareko bakoitzak ez du zertan berehala erantzun jasotako eskaera. Izan ere, 1001 eskaera jasotzerakoan, orokorrean (ez beti) 1007 eskaera bidaltzen da, eta ondoren bai, 1001 mezuaren erantzuna bidaltzen da, auzokideen zerrenda zabalduz. Hurrengo irudian bi nodoen komunikazioa nolakoa den adieraziko da:



3.4 Irudia: Levin komunikazioaren diagrama (*handshake*)

Sareko nodo batek 1001 mezuaren erantzunarekin, auzokideen zerrenda (gutxi gorabehera 250 IP helbide eta portu bikotez osatuta) helaraziko dio beste nodoari: IP helbide bakoitza adierazteko 4 byte behar dira, 8 bit eremu bakoitzeko, [0-255] tarteko 4 eremu baititu, puntu batez bereizita. Informazio horren aurretik, zenbait hitz gako identifika daitezke, adibidez addr hitza (ASCII kode hamaseitarra '61 64 64 72'). Horrez gain, IP helbidearen 4 byte-ak '06' (ASCII taulan `ACK`) byte-aren tartean agertuko dira. IPv4 maskaratuak ere jasotzen dira, formatu oso antzekoarekin.

Mezu hauek bidali ondoren, konexioa ezarritako nodoek Levin protokoloak definitzen dituen beste mezuak jaso daitezke. Adibidez, bloke berrien mezuak (2001), transakzio berrien mezuak (2002), sinkronizazio mezuak (1002), Ping mezuak (1003)...

Proiektu honetan bidaliko diren mezuak 1001 eskaera (nodo bakoitzaren auzokide zerrenda jasotzeko) eta 1003 eskaera (Monero sarera konektatuta jarraitzen duten jakiteko) izango dira. Ondoren, sarean atzigarri aurkitu diren nodoetatik transakzioak (2002 mezuak) jasotzen saiatuko da, sarearen lan-karga aztertzeko.

Horrez gain, proiektuko helburuak lortzeko beharrezkoa ez izan arren, Levin protokoloak hainbat mezu mota definitzen ditu I2P eta Tor sareen gainean bidali ahal izateko, eta goiburuko ondorengo balioek alda dezakete mezuaren funtsa:

- Jakinarazpenen kasuan, eskaera eremuak 0 balioa izango du, baina Q bit-a gaituta egongo da, gainerakoak ez.
- Eskaeren kasuan, eskaera eremuak 1 balioa izango du eta Q bit-a izango da gaituta egongo den bit bakarra.
- Erantzun mezuetan, eskaera eremuak 0 balioa hartuko du, eta S bit-a aktibatuko da soilik.
- Zatitutako mezuetan aldiz, B eta E bit-ak ez dira inoiz aldi berean aktibatuko. Izan ere, B bit-ak mezu mota honen hasiera markatzeko balio du eta E bit-tak azken mezu zatia dela adierazteko. Q eta S bitak ez dira inoiz gaituko eta eskaera eremuan 0 balioa egongo da. Zatitutako mezu hauek I2P eta Tor-ren gainean erabiltzeko sortu ziren, eta datu “errealak” bidaltzen dituzten mezuak hurrengo mezu motatik bereizteko balio du.
- *Dummy* mezuen kasuan, B eta E bit-ak gaituta egongo dira, Q eta S aldiz, desgaituta. Gainera, eskaera eremuan 0 balioa izango da eta mezuaren informazioa alde batera utz daiteke. Hau da, mezuaren datuek ez dute zerikusirik mezuaren funtsarekin.

Monero tresna pribatutasuna eta anonimo izatea bermatzen duen kriptotxanpon bakarrenetakoa denez (lehiakideak Zcash eta Dash izanik), transakzio asko sare sakonean gauzatuko direla espero da.

4. KAPITULUA

Diseinua eta inplementazioa

Kapitulu honetan, Monero sareko nodoekin komunikatu eta informazio baliagarria lortu ahal izateko garatutako tresnaren diseinua eta funtzionamendua azalduko da. Garrantzitsua da kontuan edukitzea sarean nodo guztiek berdin jokatzen dutela, parekoen arteko (P2P) sarea da eta. Horregatik, konektatuta dauden nodoei eskaerak eginez, sareari buruzko informazioa lor daiteke. Garatutako proiektua[1] github-en atzigarri dago.

Aurreko kapituluan (3. kapituluan) aztertu den Levin protokoloaren dokumentazioa githubeko Monero proiektuan urri samarra denez, hainbat proba egin dira (`tcpflow` sareko trafikoa harrapatzeko tresna erabiliz, hainbat script antolatuz `hexdump` iraulketa hamaseitarra baliatuz eta testu tratamendurako komandoak konbinatuz), nola komunikatzen diren hobeto zehazteko. Jarraian, atal honetan azalduko diren tresnak garatu dira, C eta Python programazio-lengoaiak erabiliz, Monero sarean eskaerak eginez eta lortutako informazioa mapan adierazten, hurrenez hurren.

Programaren helburua Monero sarera konektatuta dauden ahalik eta nodo gehien lortzea da, baita ere horiek bidaltzen dituzten transakzioak jasotzea, sarearen lan-karga lekuaren arabera edo nodoz nodo aztertzeko. Izan ere, nodo guztiek Monero sarean berdin jokatu arren, Internet-en pribatutasuna eta anonimo izatea bermatzeko mekanismoak erabili ohi direnez, azken horiek izan ditzaketen eraginak aztertuko dira lortutako emaitzekin.

Exekuzioa bukatzean idatzitako fitxategietan lortutako informazio osoa izango da, analisia zuzena egin ahal izateko; hari bakoitzak bere fitxategian idatziko du exekuzioaren gertaerei buruzko informazioa, haien arteko nahasteak ekidinez.

[1] https://github.com/iperezz97/monero-sarea-aztertzen

Programa nagusiak pantailan idazten du lortutako informazioa mapan era errazean ikusi ahal izateko. Eragiketaren iniziala (a, r, t edo q; *add, remove, transaction* eta *quit* ingeleseko hitzetatik), nodoaren koordenatuak, IP helbidea eta hiria edo transakzio kopurua idatziko da. Python programak aldiz, sarrera estandarra irakurriko du etengabe eta bi programek aldiberean funtzionatu dezaten *pipe*-a erabiltzen da, programa nagusiaren irteera estandarra bigarren programaren sarrera estandarrera berbideratzeko. Modu horretan, bi programak exekuzioan egongo dira etengabe datuak lortzen eta mapan bistaratzen.

Kapitulu honetan agertzen diren irudietan, garatutako programen funtzionamendua modu laburrean adierazten da, hari bakoitzaren portaera deskribatuz fluxu-diagramaren bidez. Horrez gain, proiektuko fitxategien helburua eta deskribapen orokorra azaltzen da.

Hala ere, diseinua aurkeztu aurretik, hurrengo irudian elementu bakoitzak diseinuan izango duen esanahia argituko da:

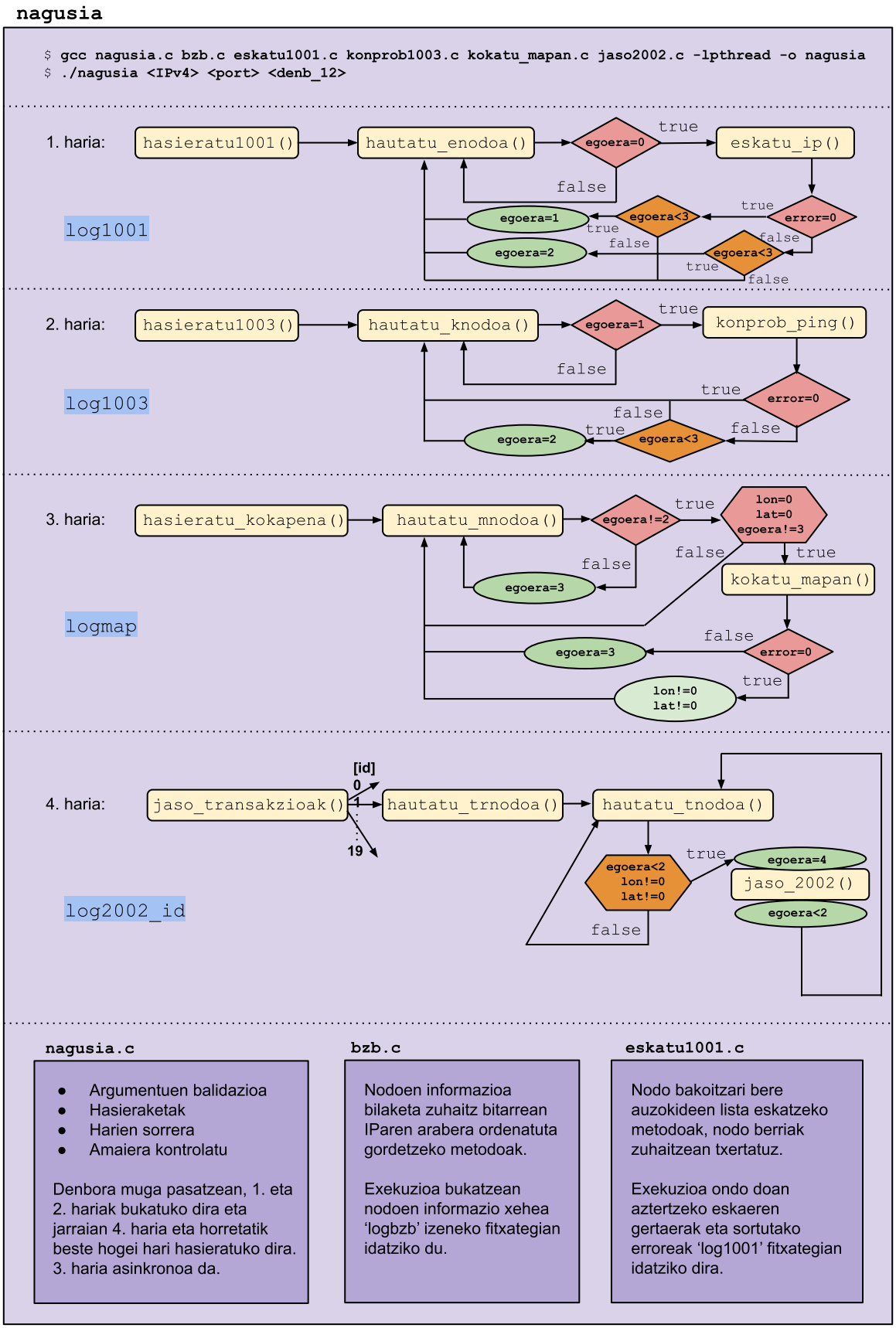
Forma	Esanahia
`metodoa()`	Egoera: prozesua unean exekutatzen ari den metodoa adierazteko erabiliko da
`a = b`	Baldintza: barneko adierazpena betetzen den aztertzeko balio du, berdinak diren adibidez.
`a = b`	Ekintza: diagraman eragina duen ekintza bat exekutatuko da, esleipena adibidez.
→	Fluxu lerroa: prozesuaren uneko trantsizioa (norantza) zehazten du.

4.1 Irudia: Fluxu-diagramako elementuen definizioa

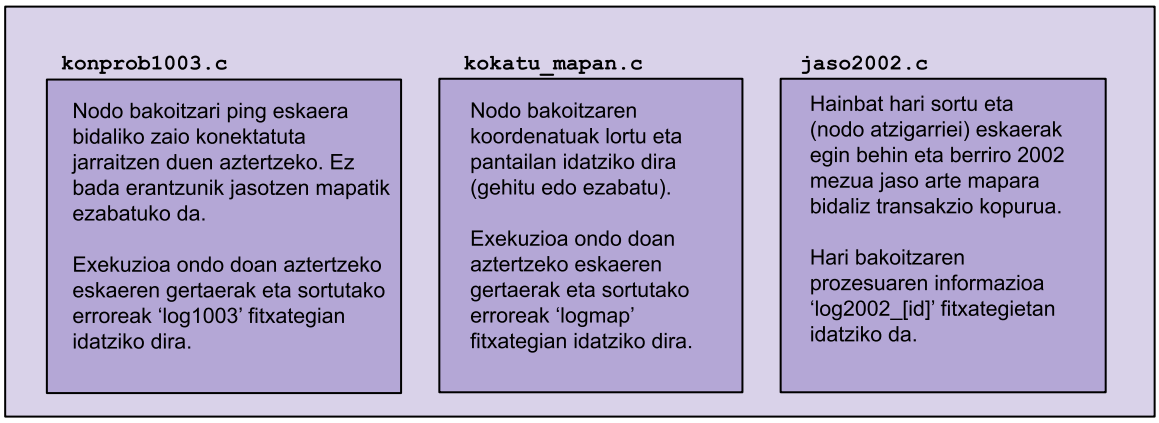
Forma berdina baina kolore biziagoarekin koloreztatuta dagoen kasuetan, uneko nodoa blokeatu egin dela adierazten da. Izan ere, atomikotasuna bermatu behar da, hari batek aldaketa bat egin bitartean beste batek datu okerrak irakurtzea saihestuz. Gainera, koherentzia mantendu behar da uneko prozesu eta nodoaren egoerarekin, beraz, sekzio kritikoetan nodoa blokeatu egiten da beste hariei nodo horren atzipena ukatuz.

4.1 Programa nagusia

Hurrengo irudian programa nagusiaren diagrama-fluxua eta beste hainbat zehaztapen azter daitezke:



4.2 Irudia: Programa nagusiaren diseinua (1/2)



4.3 Irudia: Programa nagusiaren diseinua (2/2)

Hasieran, programa nagusiak argumentuak ondo zehaztu diren aztertuko du eta okerrak diren kasuan, exekuzioarekin hasteko laguntza mezua pantailaratuko da, baita Monero hazi nodoen (*Monero seed nodes*-en) IP helbide eta portua ere. Irudian (4.2) ikus daitekeen bezala, programa nagusiak guztira lau hari sortzen ditu (nahiz eta denbora muga pasa arte 3 hari izan exekuzioan, eta 4.a hasieratu aurretik 1. eta 2. harien exekuzioa amaitzen den), bakoitzak zeregin desberdina izanik:

- Lehenengo hariak *socket*-a sortu, argumentu bezala zehaztutako IP helbide eta portuari konektatu eta *handshake* (1001) eskaera egingo du (erroa izango den nodoari), nodoaren auzokideen zerrenda eskatuz. Erantzunik jasotzen ez bada, programa amaituko da ez baita nodorik geratuko eskaerak bidaltzeko. Aldiz, erantzuna jasotzen bada, mezuaren barnean IP helbideak aurkitu eta bilaketa zuhaitz bitarrean txertatzen dira. Ondorengo iterazioetan, lortutako nodoei eskaera bera egingo zaie *pre-order* ordenean, eta honela, nodoak gehitzen diren heinean datu-egitura hedatu egingo da, argumentu bezala pasatako denbora muga bete arte. Nodo bakoitza behin sartuko da prozesu honetara eta bukatzerakoan, egoera aldatzeko nodoa blokeatuko da, aldi berean beste hari baten atzipenak programaren portaeran eraginik ez izateko. Hari honek bere prozesuari buruzko informazioa '`log1001`' fitxategian idatziko du.

- Bigarren hariak *handshake* erantzuna bidali duen nodo bat aukeratuko du prozesuarekin hasteko, *pre-order* ordenean zuhaitzaren nodo guztiak atzituz behin eta berriro, denbora muga igaro arte. Bidaltzen den eskaera *ping* (1003) mezuarena da, eta erantzunik jasotzen ez den kasuetan, nodoa blokeatu eta egoera aldatuz berriro prozesua errepikatzea ekidingo da. Zehazki, nodo bakoitzerako *socket*-a sortu,

konexioa ezarri, eskaera bidali eta erantzuna jasotzen saiatuko da. Segundu bateko denbora muga izango dute konexioa ezartzeko eta beste bat erantzuna jasotzeko, zain ez geratzeko. Hari honek '`log1003`' fitxategian idatziko du bere prozesua bete ahala, ondo joan den edo zein errore egon den nodo bakoitzeko.

- Hirugarren hariak modu asinkronoan funtzionatzen du, datu-egitura nagusia behin eta berriro *pre-order* ordenean iteratzen, lortutako nodo guztien koordenatuak lortuz eta irteera estandarrean idatziz. Nodo bakoitza behin sartuko da prozesu hau exekutatzera, aurretik jaitsi beharreko fitxategi bat ('maxmind4.dat') eta `geoiplookup` komandoa erabiliz koordenatuak lortu ahal izateko. Fitxategi hori deskargatzeko, `get_maxmind.sh` script-a sortu da, geoip karpetan erauziko duena, bestela github-eko proiektuaren geoip karpetan aurki daiteke. Nodoaren koordenatuak lortzerakoan, irteera estandarrean idatziko dira, ondoren maparen programak *pipe*-aren bidez irakur ditzan eta puntua mapan koka dezan. Beste alde batetik, uneko nodoak aurreko hariren bati erantzun ez badio, pantailan idatziko duen eragiketa nodoa mapatik ezabatzeko izango da. Hari honek exekuzioa amaitzeko, mapan gehitu dituen nodo kopurua kontatuko du eta 1. hariak bukatu duen aztertuko du. Horrek bukatutakoan, eta lortutako nodo kopurua mapan gehitu diren nodo kopuruaren berdina denean, 3. haria itxi egingo da. Bere prozesuari buruzko informazioa '`logmap`' fitxategian idatziko du hari honek.

- Laugarren haria lehenengo bi hariak amaitzerakoan hasiko da, aurretik lortu diren eta atzigarri egon diren nodoekin konexioa ezarriz, haietatik transakzio mezuak harrapatzeko helburuarekin. Horretarako, hogei hari sortuko dira, prozesua konkurrentea eginez, hari bakoitzak nodoa blokeatuko baitu beste batek atzi ez dezan egoera aldatzen den bitartean, eta beste kasuetan bezala, *pre-order* ordenean iteratuko da bilaketa zuhaitz bitarra behin eta berriro, bukatzeko seinalea bidali arte. Transakzioen jakinarazpen mezuak sareko nodoetatik jasotzeko, *handshake* eskaera bidaltzea beharrezkoa da, eta ondoren, bost segunduko denbora mugarekin, mezu gehiago jasotzen geratuko da 2002 mezua jaso arte. Hori jasotzean, mezuaren tamaina kontuan hartuta, bidali diren transakzio kopurua estimatuko da, 500 byteko transakzioak direla suposatuz. Hala ere, ohikoa da transakzioetan segurtasun-mekanismoak erabiltzeagatik tamaina hori asko handitzea. Azkenik, estimatu den transakzio kopurua nodoaren informazioarekin batera irteera estandarrean idatziko da maparen programak irakur dezan. Hariak sortzen dituen harietarako ($id = 0, 1, ...19$) '`log2002_id`' fitxategietan idatziko da prozesu bakoitzaren informazioa.

Irudian agertzen den nodo bakoitzaren egoera aldagaia kritikoa da harien funtzionamendu zuzena bermatzeko eta jarraian egoera bakoitzak duen esanahia deskribatuko da:

- Egoera 0: aurkitua izan da beste nodo bati 1001 bidaliz
- Egoera 1: pareen zerrendarekin erantzun du (1001 jasota)
- Egoera 2: ez du erantzun eta mapatik ezabatuko da (egituratik ez)
- Egoera 3: mapatik ezabatu da
- Egoera 4: erreserbatuta transakzioak jasotzeko prozesuarentzako

Tresna hau garatu den bitartean, kontuan hartu da hari bakoitzak hainbat deskriptore izango dituela irekita, *socket*-a eta fitxategi deskriptorea(k). Izan ere, kernel-ak defektuz mugatu egiten du irekita egon daitezkeen deskriptore kopurua.

Horrez gain, konexioa ezartzerakoan, ezer gehiago adierazi ezean, ausazko portu zenbakia erabiliko litzateke (konexio bakoitzean desberdina izanik) eta konexio asko ezarri ondoren, portu bakoitzari lotuta hondakin-informazioa geratzen denez, ez da porturik geratuko erabilgarri. Horregatik, konexio guztietarako portu berdina (28080) berrerabiltzea adierazi zaio programari. Konexioa ezartzeko segundu bat eskaintzen zaio, eta modu berean, mezu bat jasotzeko itxarongo den gehienezko denbora segundu batekoa izango da.

Programa nagusiak hasi eta bukatu bitartean, `SIGINT` (teklatu bidezko etena edo Ctrl + C) eta `SIGTERM` (`kill` komandoaren lehenetsitakoa) seinaleak harrapatuko ditu exekuzioa modu egokian bukatzeko. Nodoen informazio osotua (IP-aren arabera ordenatuta) `logbzb` fitxategian idatziz, eta maparen programa modu egokian itxiarazteko, `q` (*quit*) eragiketa irteera estandarrean idatziz. Programaren exekuzioa era errazean bukatzeko, programatutako `exekuzioa_bukatu.sh` script arrunta exekuta daiteke.

Hari bakoitzak log fitxategi batean idatziko ditu prozesuaren urrats nagusiak eta errorerik balego, errorearen mezua eta zenbakia idatziko ditu kodean kokatu ahal izateko. Horrez gain, lehenengo hiru hariek exekuzioa ixterakoan, bukatu hitza idatziko dute, berea den log fitxategi horretan.

Adibide bezala, lehenengo hariak prozesua bete bitartean sortu duen fitxategiaren hasiera azter dezakegu (`log1001`):

```
Socket-a sortzen...   Helburuko nodoa: 212.83.175.67 18080
Socket deskriptorea: 5
Portua zehaztuta      Ongi konektatu da
1001 eskaeraren goiburukoa bidalita   Pakete tamaina: 33 (10),    21 (16).
1001 eskaeraren datuak bidalita       Pakete tamaina: 226 (10),   e2 (16).
1007 eskaera jasota (baztertu)        Pakete tamaina: 10 (10),    0a (16).
1001 erantzunaren goiburukoa jasota   Pakete tamaina: 33 (10),    21 (16).
1001 erantzunaren datuak jasotzen...  Pakete tamaina: 15140 (10), 3b24 (16) > 212.83.175.67

Socket-a sortzen...   Helburuko nodoa: 104.195.140.115 18080
Socket deskriptorea: 5
Portua zehaztuta      Ongi konektatu da
1001 eskaeraren goiburukoa bidalita   Pakete tamaina: 33 (10),    21 (16).
1001 eskaeraren datuak bidalita       Pakete tamaina: 226 (10),   e2 (16).
1007 eskaera jasota (baztertu)        Pakete tamaina: 10 (10),    0a (16).
1001 erantzunaren goiburukoa jasota   Pakete tamaina: 33 (10),    21 (16).
1001 erantzunaren datuak jasotzen...  Pakete tamaina: 14861 (10), 3a0d (16) > 104.195.140.115

Socket-a sortzen...     Helburuko nodoa: 135.181.96.133 18080
Socket deskriptorea: 5
Portua zehaztuta      Ongi konektatu da
1001 eskaeraren goiburukoa bidalita   Pakete tamaina: 33 (10),    21 (16).
1001 eskaeraren datuak bidalita       Pakete tamaina: 226 (10),   e2 (16).
Lehenik Bender's nightmare sinadura eta ondoren 1007 (komando) mezuaren goiburukoa jasota.
1007 eskaera jasota (baztertu)        Pakete tamaina: 10 (10),    0a (16).
1001 erantzunaren goiburukoa jasota   Pakete tamaina: 33 (10),    21 (16).
1001 erantzunaren datuak jasotzen...  Pakete tamaina: 15764 (10), 3d94 (16) > 135.181.96.133
```

4.4 Irudia: Lehenengo hariak sortzen duen `log1001` fitxategiaren hasiera

Hariak konexioa ezarri duen nodoaren IP helbidea izentzat izango duen fitxategian gordeko ditu nodo horretatik jasotako mezuak (datu gordinak). Hala ere, azken fitxategi bitar horiek `unlink` metodoarekin ezabatzen dira prozesuaren bukaeran, fitxategi askoren sorrera ekiditeko (fitxategi bitar horietan proben emaitzak egiaztatzen baitziren). Gainera, errorerik balego, pantailaratzean gorriz nabarmenduta agertuko litzateke kodean aurkitzeko errore zenbakia, errore mezuaren ondoren.

Eskaerak bidaltzeko prozesua berdintsua da kasu guztietan, aldatzen dena jasotako datuen tratamendua da. Hortaz, bigarren hariak sortzen duen log fitxategia ez da azalduko. Izan ere, antza handia du aztertu berri dugun fitxategiarekin, baina sinpleagoa da hariak ez duelako informaziorik biltegiratu behar, soilik erantzuna jaso duen aztertuko du.

Gainera, nodo batzuk 1001 eskaera jasotzerakoan, 1007 eskaerarekin erantzun ordez, 1001 erantzuna bidaltzen dute zuzenean, eztabaida honetan [2] aztertzen den bezalaxe. Horrez gain, 4.4 irudian ikus daitekeen bezala, lehenik *Bender's Nightmare* sinadura eta ondoren gainerako goiburukoa bidaltzen duten kasua tratatzen da. Garrantzitsua da 1001

[2] https://github.com/monero-project/monero/issues/6272

erantzun mezuan, auzokideen IPv4 eta IPv6 helbideak bidaltzen direla, azken hauek IPv4 maskaratuak izanik. Beraz, horiek guztiak IPv4 forman gordetzen dira, bihurketa eginez beharrezkoa den kasu horietan.

Hurrengo informazioa 3. hariak prozesua bete bitartean idazten duena izango da, `logmap` fitxategiaren hasiera, hain zuzen ere:

```
IP helbideen datuak 'geoip' karpetan deskargatzeko exekutatu 'sh get_maxmind.sh'
Koordenatuak lortzen... Helburuko nodoa: 185.115.97.150 18080
Pipe-an exekutatuko den komandoa:
geoiplookup -f ./geoip/maxmind4.dat 185.115.97.150 | cut -d ',' -f 5,7-8
Pipe-a ongi ireki da.
Lon: 30.340500    Lat: 59.687302  City:  Tsarskoye_Selo
----
Koordenatuak lortzen... Helburuko nodoa: 135.181.96.133 18080
Pipe-an exekutatuko den komandoa:
geoiplookup -f ./geoip/maxmind4.dat 135.181.96.133 | cut -d ',' -f 5,7-8
Pipe-a ongi ireki da.
Lon: 24.934000    Lat: 60.179001  City:  Helsinki
```

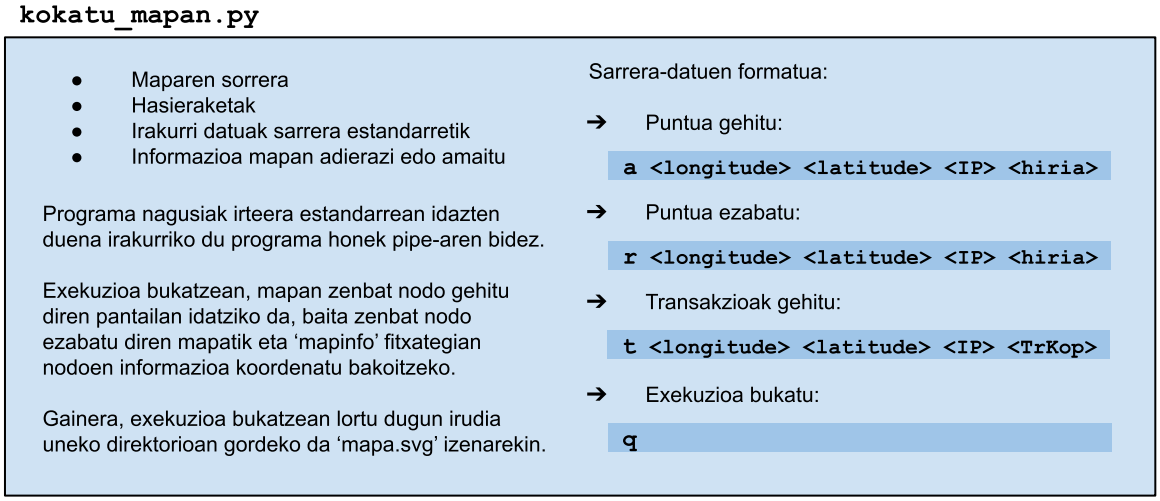
4.5 Irudia: Hirugarrren hariak sortzen duen `logmap` fitxategiaren hasiera

Nodo bakoitzaren koordenatuak lortzeko erabili den fitxategi bitarra aurretik aipatutako `get_maxmind.sh` script-a exekutatuz apirilean deskargatu zen. Script-ak ondorengoa egiten du; uneko direktorioan ‘geoip’ karpeta sortu (sortuta ez badago), deskargatu konprimitutako fitxategia eta erauzi. Gutxi gorabehera hilero eguneratzen da fitxategi hori, beraz erabiltzen den bertsioaren arabera, nodoen koordenatu desberdinak lor daitezke.

Nodo baten IP helbidea fitxategian aurkitzen ez bada, errorea harrapatu eta nodoa 3. egoerara pasako da mapatik ezabatuta bezala. Gainera, gerta daiteke hiria-eremuan ‘N/A’ balioa agertzea, baina kokapen ezagunenen koordenatuak konparatu eta horiei dagokien hiria esleituko zaie mapan kokatzeko programan: Parisen kasuan, adibidez.

4.2 Datuak mapan kokatzeko programa

Sarean eskaerak eginez lortutako informazioa mapan kokatzeko Python programaren helburua eta formatu zehaztapena zein diren ikus daiteke ondorengo irudian:



4.6 Irudia: Nodoak mapan kokatzeko programaren portaera

Programa honek datuak sarrera estandarretik jasotzen ditu, baina kasu honetan, *pipe*-aren bidez jasoko ditu; programa nagusiak irteera estandarrean idazten dituenak, hain zuzen ere. Horrela, bi programak exekuzioan ariko dira datuak lortu bitartean, exekuzioa eten nahi den arte aurretik aipatutako script-arekin edo prozesuak akatzeko seinaleak bidaliz. Aipatzekoa da programa honen lan-karga nahiko handia dela, eta proba txikia ez den kasuetan, programa nagusiak idatzi baino motelago irakurtzen dituela datuak.

Longitude balioak -180 eta 180 artean egon beharko dira, latitude balioak aldiz -90 eta 90 artean. Notazio hamartarra erabiltzen da, eta ez notazio hirurogeitarra. Sarean aurkitutako nodo bakoitza mapan kokatuko da, eta saguarekin puntu baten gainetik pasatzerakoan, puntu horretan dauden nodoak (bakoitzaren IP helbidea) eta haiek bidalitako transakzio kopurua ikusi ahalko da, horiek egitura berean gordetzen direlako. Horrela, sarearen lan-karga deszentralizatua dagoen aztertuko da eta zein lekuetan erabiltzen den gehien.

Hala ere, Monero erabiltzaileek askotan segurtasun-mekanismoak dituztenez, zerbitzu hori eskaintzen duten makinak identifika daitezke zuzenean erabiltzaile horiek ordez; pribatutasuna bermatzen duen azpi-sare baten irekiera nodoa, adibidez.

Exekuzioa hasterakoan, datuak irakurtzeko formatua inprimatuko da (behin soilik). Datuen formatuan akatsik egin bada, errore mezua pantailaratuko da. Ondoren, transakzioak jaso bitartean, transakzio horien iturburu nodoaren IP helbidea adieraziko da, baita jasotako transakzio kopurua eta guztira bidali dituen transakzio kopurua ere, mapan bezala.

Horrelako irteera ikusiko dugu bi programak exekutatzen ari diren bitartean:

```
Koordenatuak irakurtzen...
<Eragiketa> <Longitude> <Latitude> <IP> <City/Tr>:
Transakzioak jasotzen...
Jasotako transakzioak:  14   ( 24.10.144.178 14 )
Jasotako transakzioak:  7    ( 24.54.163.12 7 )
Jasotako transakzioak:  14   ( 24.101.115.189 14 )
Jasotako transakzioak:  2    ( 31.187.54.2 2 )
Jasotako transakzioak:  2    ( 5.9.193.22 2 )
Jasotako transakzioak:  2    ( 14.137.221.49 2 )
Jasotako transakzioak:  5    ( 37.135.120.50 5 )
Jasotako transakzioak:  2    ( 24.84.229.27 2 )
Jasotako transakzioak:  10   ( 79.136.72.137 10 )
Jasotako transakzioak:  10   ( 24.132.75.232 10 )
Jasotako transakzioak:  6    ( 47.185.243.76 6 )
Jasotako transakzioak:  3    ( 24.10.144.178 17 )
```

4.7 Irudia: Maparen programak sortzen duen irteera estandarra

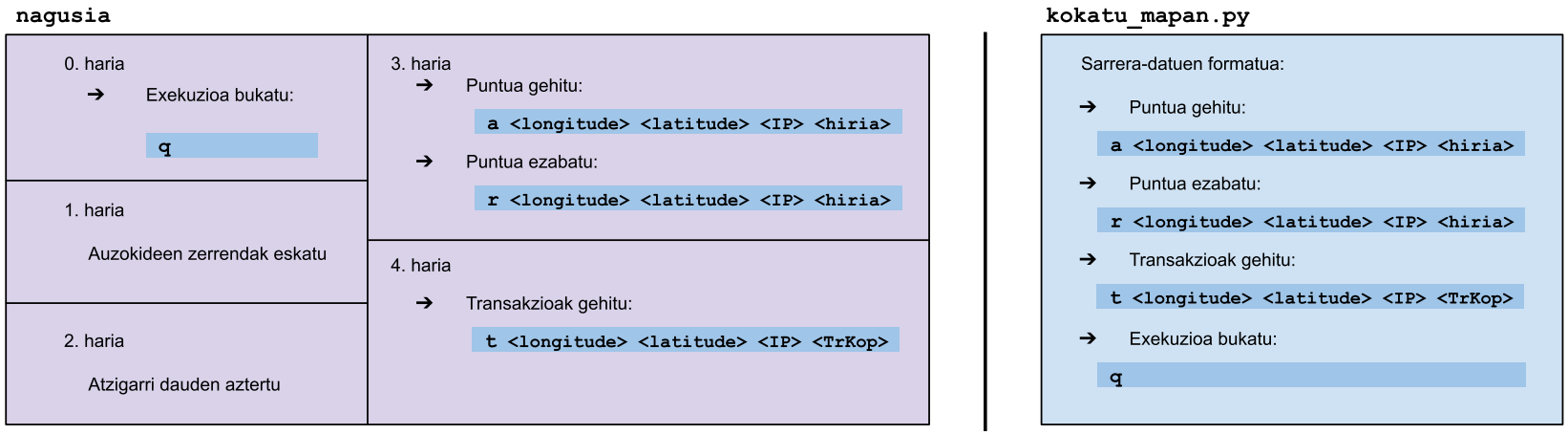
Jaso diren lehen eta azken transakzio mezuak nodo beretik jaso direla ikus daiteke, eta kopuru horien batura gorde da nodo horren informazioan. Transakzio bakoitzak 500 byte dituela estimatu dela kontuan hartuz, lehen mezuak 7000 byte inguru zituen, eta azkenak aldiz, 1500 byte inguru. Hortaz, 24.10.144.178 IP helbidea duen nodoak guztira 17 transakzio bidali dituela gorde da, nahiz eta transakzioen segurtasun-maila handiagotzean horien tamaina ere handiagotuko den: Monero transakzioen batez besteko tamaina 3 KB-ekoa da. Monero transakzioen hainbat adibide ematen dira eztabaida honetan[3].

4.3 Prozesuen arteko komunikazioa

Programa nagusiak irteera estandarrean idatziko duen informazioa maparen programak *pipe*-aren bidez irakurri beharko duenez, protokolo txiki bat definitu da bi prozesuen arteko komunikazioa ahalbidetzeko.

Hurrengo irudian (4.8 irudian) ikus daiteke bi prozesuen komunikazioa nola gauzatzen den:

[3] https://monero.stackexchange.com/questions/211/how-much-larger-are-monero-transactions-compared-to-the-average-bitcoin-transact



4.8 Irudia: Bi prozesuen arteko komunikazioa *pipe*-aren bidez

Lehenengo eta bigarren hariek ez dute pantailan ezer idatziko, eskaerak egiten eta erantzunak jasotzen mantenduko dira argumentu bezala pasatako segundu kopurua igaro arte. Bitartean, hari guztien guraso den hariak (zerogarren hariak) programaren bukaera kontrolatuko du eta bukatzeko seinalea harrapatzerakoan, python programa era zuzenean ixteko eragiketa idatziko du pantailan (q). Horrela, maparen programak lortu duen informazio guztia gordeko da. Hirugarren hariak nodoen koordenatuak lortu eta pantailan idatziko ditu python programak mapan irudikatzeko. Modu berean, nodoak erantzunik bidali ez badu, hari honek nodoa mapatik ezabatzeko eragiketa idatziko du pantailan. Azkenik, laugarren hariak nodo bakoitzetik une bakoitzean jasotako transakzio kopurua idatziko du pantailan.

Gainera, proiektuaren helburuak lortzeko bi programen exekuzioa konbinatu den arren, bi programak beraien kabuz exekuta daitezke, bestelako probak egin nahi badira.

5. KAPITULUA

Datuen analisia

Kapitulu honetan, garatutako tresnekin lortutako informazioa eta emaitzak aztertu, eta beste eredu batzuekin konparatuko dira.

Aurretik aipatu bezala, programa nagusia konpilatzeko hurrengo komandoa exekutatuko da proiektuaren direktorioan:

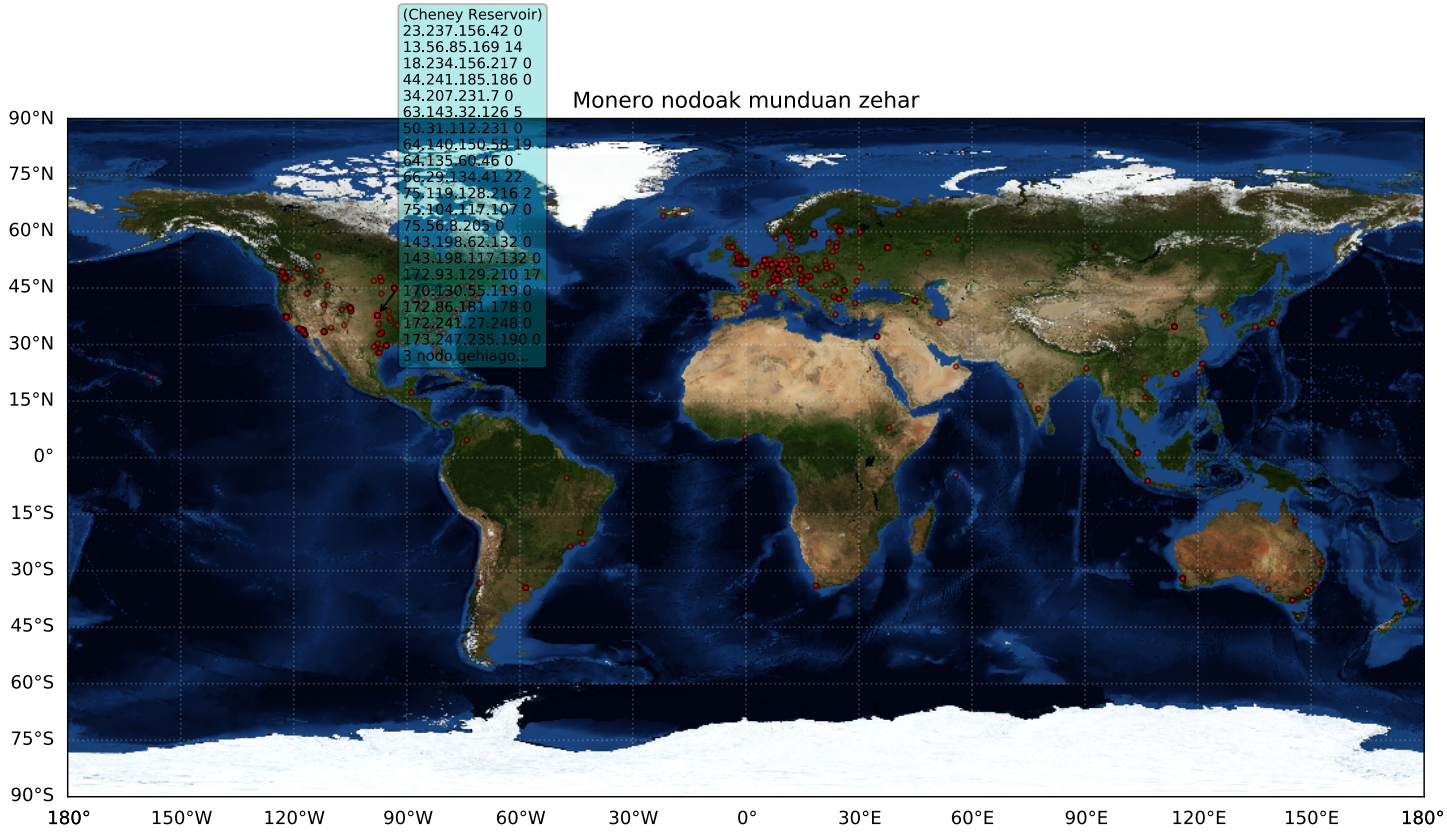
```
$ gcc nagusia.c bzb.c eskatu1001.c konprob1003.c kokatu_mapan.c jaso2002.c -lpthread -o nagusia
```

Ondoren aztertuko den irudia terminalean (proiektuaren direktorioan kokatuta) ondorengo komandoa exekutatuz lortu da:

```
$ ./nagusia 212.83.175.67 18080 1 | python kokatu_mapan.py
```

Programa exekutagarriaren izenaren ondoren, eskaerak zein nodorekin hasiko diren zehaztu behar da, adierazitakoa Monero hazi nodo bat izanik (portu zenbakia ere beharrezkoa da), eta jarraian eskaerak egiteko (1. eta 2. hariek) izango duten denbora muga zehaztu behar da (segunduetan). Denbora horren bitartean, nodo berriak aurkituko dira (bilaketa zuhaitz bitarrean txertatuz eta mapan kokatuz) eta horietatik erantzunik jasotzen ez bada, mapatik ezabatzeari ekingo zaio.

Egindako proban, 30 minutuz transakzioak jasotzen egon da informazio guztia mapan kokatzen zuen bitartean. Nodo asko aurkitzen diren kasuetan, lan-karga handia ematen dio informazio guztia mapan kokatzeak, batez ere bereizmen handiko irudia sortzen bada.



5.1 Irudia: Segundu bateko proban aurkitutako Monero nodoak mapan

Bi programen exekuzioarekin lortutako irudia ikus dezakegu 5.1 irudian, 567 nodo atzigarri kokatu dira mapan, eta guztira 3170 transakzio jaso dira.

Sagua puntu gorri handienaren gainean jarri da, koordenatu horietan kokatu diren nodoen IP helbidea eta bakoitzetik jasotako transakzio kopurua etiketa batean adieraziz. Etiketa guztiak nodoak kokatu diren hiri edo lekuaren izenarekin hasten dira. Probako kasuan, Cheney Reservoir ez da hiri bat zehazki, baina uste da toki horretan seinale asko elkartzen direla eta horregatik, antzematen dira hainbeste nodo kokapen horretan.

Horrez gain, mapan agertzen den informazioa fitxategietan idazten den informazioarekin egiazta daiteke. Izan ere, hari bakoitzak bere prozesuaren informazioa idazten du eta bukaeran maparen informazioa (`mapinfo` fitxategian) eta datu-egitura nagusiaren informazioa (`logbzb` fitxategian) idazten da.

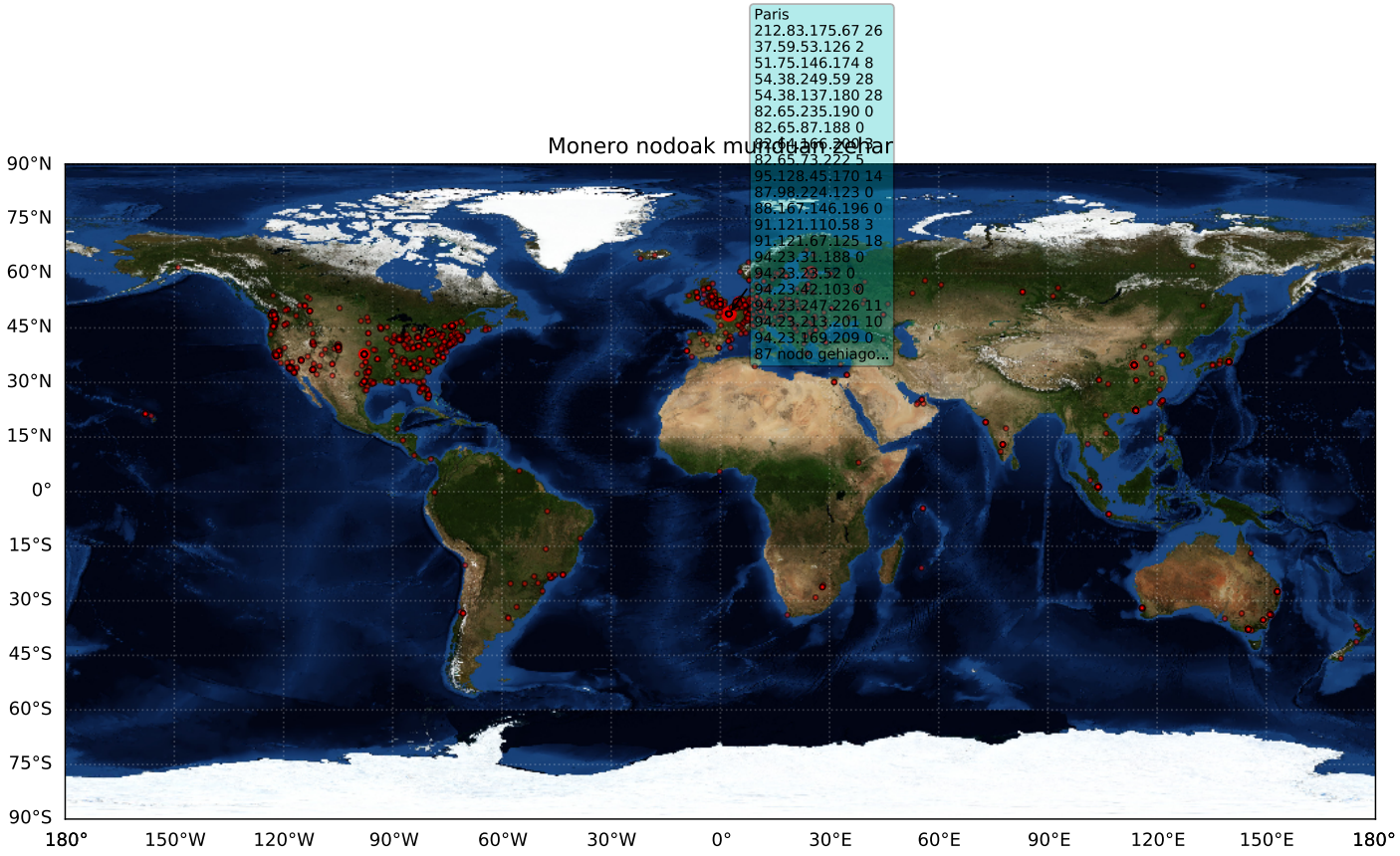
Oraingoan bilaketa zuhaitz bitarrak gordetzen duen informazioa aztertuko da, `logbzb` fitxategiko lehen lerroak, exekuzioaren bukaeran idatzita:

Sak.	IPv4	Port	Eg.	Lon	Lat	TrKop.
9	1.172.211.48	18080	3	0.000000	0.000000	-1
8	5.8.18.31	18080	0	29.471100	46.830601	0
9	5.9.120.18	18080	3	0.000000	0.000000	-1
7	5.9.120.250	18080	0	9.490900	51.299301	8
8	5.12.152.181	18080	0	26.169001	44.420502	0
6	5.39.49.229	18080	0	2.338700	48.858200	0
7	5.54.43.44	18080	4	23.735300	37.984200	0
5	5.65.201.242	18080	0	-2.509000	51.489498	23
9	5.189.152.182	18080	0	11.161700	49.404999	0
10	5.189.171.193	18080	0	11.161700	49.404999	32
11	13.56.85.169	18080	0	-97.821999	37.750999	14
8	18.132.124.81	18080	0	-0.093000	51.516399	0
7	18.156.193.26	18080	0	8.684300	50.118801	26
10	18.162.142.29	18080	0	114.165703	22.257799	18
9	18.234.156.217	18080	0	-97.821999	37.750999	0
10	20.198.250.94	18080	0	103.856903	1.302900	0
9	23.111.236.124	18080	0	4.899500	52.382401	7
10	23.111.236.156	18080	0	4.899500	52.382401	2

5.2 Irudia: Exekuzioa amaitzean idazten den `logbzb` fitxategiaren hasiera

Datuak egiaztatu nahi baditugu, 5.1 irudian ikus daitekeen etiketan dagoen bigarren nodoaren eta `logbzb` fitxategian (5.2 irudian) ikus daitekeen 12. lerroan dagoen nodoaren informazioa bat datorrela konproba daiteke.

Hurrengo orriko proban hamar segunduz mantendu dira eskaerak, nodo gehiago lortzeko asmoarekin eta ondoren hiru ordu egon da transakzioak jasotzen eta *pipe*-aren bidez informazioa irudiko mapan islatuz.



5.3 Irudia: Hamar segundutan aurkitutako Monero nodoak mapan

Exekuzioa amaitzeko script-a exekutatu ondoren, ordu bat behar izan du bigarren programak exekuzioa modu egokian ixteko: programa nagusiak bidalitako lerro guztiak irakurri arte. Bukaeran, `kokatu_mapan.py` programak `mapinfo` fitxategia sortuko du exekuzioa modu egokian ixten denean, nodoen informazioa gordetzeko, baina erabiltzen den datu-egiturarengatik koordenatu pare bakoitzeko aurkituko ditugu nodoak (mapan zeudenak bakarrik) eta haien transakzio kopurua.

Irudian, Parisen 107 nodo aurkitu direla ikus daiteke, eta horietako batzuk transakzioak jaso dituzte. Exekuzio honen amaieran, 30944 transakzio jaso direla estimatu da, eta 1763 nodo atzigarri kokatu dira mapan. Gainera, 5 nodo ezabatu dira mapatik horietatik erantzunik jaso ez delako. Hala ere, espero zen bezala, geroz eta denbora gehiago eman eskaerak egiteari, orduan eta nodo gehiago aurkituko dira, nodo gehiago ezabatuko dira mapatik eta motelagoa izango da maparen programa, eta beraz, gehiago atzeratuko da datuak irakurtzen eta mapan islatzen. Atzerapen horri aurre egiteko, ordenagailu ahaltsua beharko da, exekuzioaren denbora asko murriztuko baita.

Proba hauek gama ertaineko ordenagailu eramangarri batekin egin baitira, 4 *core* edo nukleo dituena. Prozesadorearen dedikazioa % 95ean mapan kokatzeko programari zegokion, aldiz, programa nagusiak lan-karga txikia zuen.

Oraingoan, `top` komandoaren irteera aztertuko da bi programak exekuzioan ari diren bitartean, (`nagusia`) eta maparen programaren (`python`) errendimendua aztertzeko:

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1729 iperez    20   0 1558104 683836  56372 R  98,1  8,5   1:15.53 python
 2213 iperez    20   0 4195860 219048 110492 S   5,3  2,7  47:02.10 gnome-shell
 1598 iperez    20   0 1032504  71028  60300 S   2,8  0,9  42:31.05 Xorg
 5075 iperez    20   0   44512   4156   3348 R   2,8  0,1   0:00.30 top
 4084 iperez    20   0  796956  57272  15172 S   1,6  0,7   3:55.22 terminator
 1728 iperez    20   0 1630984   5428   1028 S   0,6  0,1   0:02.27 nagusia
```

5.4 Irudia: `top` komandoaren irteera exekuzioaren hasieran

Hala ere, argitu behar da 4 nukleo izanda, CPU erabileraren maximo teorikoa % 400 izango litzakeela, eta python prozesua CPU erabileraren % 75-105 tartean mugitzen da.

Gainera, denbora (`TIME+`) zutabean, prozesu bakoitzari eskaini zaion minutu:segundu.ehunen kopurua ikus daiteke; nahiz eta exekuzioa aldi berean hasi, mapan kokatzeko programan hogeita hamar aldiz gehiago aritu da. Horrez gain, memoriaren erabilera gehien duen prozesua Python programarena da, programa nagusiaren memoriaren erabilera oso txikia den bitartean.

Hainbat minutu igaro ondoren, komando bera exekutatuz, hurrengo irteera lortu da:

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1729 iperez    20   0 1806272 944832  55500 R  99,7 11,8  29:16.21 python
 1598 iperez    20   0 1033588  72244  61148 S  14,2  0,9  45:19.62 Xorg
 2213 iperez    20   0 4196068 220044 111264 R   9,9  2,7  50:14.34 gnome-shell
 1728 iperez    20   0 1632712  69552   1028 S   0,7  0,9   0:09.62 nagusia
```

5.5 Irudia: `top` komandoaren irteera hainbat minutu ondoren

Irteera horretan (5.6 irudian), mapan kokatzeko programaren lan-karga handiagotzen joan dela ikus daiteke, exekuzioan aritu beharreko denbora gehiago handiagotzen baita programa nagusiarekin eskainitako denborarekin konparatuta.

Amaitzeko, `mapinfo` fitxategiko lehen lerroak aztertuko dira, maparen programak exekuzioaren bukaeran idatzi dituenak:

```
Coords(lon=2.3387, lat=48.8582):       Paris { 212.83.175.67 43,  37.59.53.126 2,
   51.75.146.174 28,  54.38.249.59 28,  54.38.137.180 28,  82.65.235.190 0,  82.65.87.188 0,
   82.64.166.200 8,   82.65.73.222 17,  95.128.45.170 22,  87.98.224.123 0 (...)  }
Coords(lon=12.5126, lat=41.897999):    Rome { 2.236.4.226 35   }
Coords(lon=-74.460602, lat=40.551102): Piscataway { 45.77.77.49 12   }
Coords(lon=4.2158, lat=51.993401):     Naaldwijk { 31.210.173.79 40,  212.8.251.35 9   }
Coords(lon=-74.059998, lat=40.787601): Secaucus { 37.120.202.70 3   }
Coords(lon=-117.519798, lat=33.8241):  Corona { 45.50.45.46 10   }
Coords(lon=-114.2453, lat=51.202099):  Calgary { 50.99.141.164 12   }
Coords(lon=14.2976, lat=58.297699):    Hjo { 5.150.200.52 18   }
```

5.6 Irudia: `mapinfo` fitxategiaren hasiera

Argi ikus daitekeenez, nodo askotatik jaso dira transakzioak, eta sagua puntuaren gainean jarri ondoren, informazioa eguneratu da, maparen programa bukatu bitartean transakzio gehiago jaso direlako. Izan ere, programa nagusiak irteera estandarrean idatzi dituen datuak, exekuzioa ixterakoan (eta `q` eragiketa idaztean), maparen programari sarrera estandarrean geratuko zaizkio irakurtzeke, azken honen exekuzioa motelagoa baita, proba txikietan izan ezik.

Hainbat proben datuak bateratu dira eta Monero sarean 36553 nodo (IP helbide desberdinarekin) aurkitu dira, horietatik 48277 transakzio jaso dira guztira. Bi IP helbide berdin lortzen badira baina horiek portu desberdina erabiltzen badute, programak biak gordeko ditu nodo desberdintzat, eta hori kontuan hartuta, Monero sarean 38374 IP helbide eta portu desberdin aurkitu dira garatutako tresnekin.

6. KAPITULUA

Plangintza

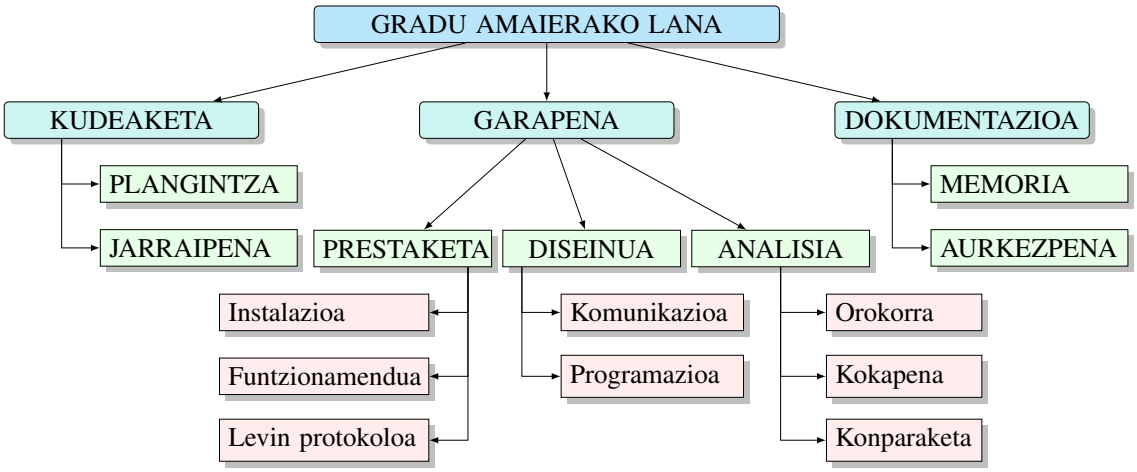
Proiektu hau egiteko, ezinbestekoa da plangintza zehatza gauzatzea, egin beharreko atazak eta arriskuak identifikatuz, lanaren garapena errazteko eta sor daitezkeen atzerapenak saihestu ahal izateko. Horrez gain, lan kargak eta atazen denbora estimazioak koherenteak eta orekatuak izan behar dira, proiektuaren helburuekin bat eginez.

Ildo berean, proiektuko oinarrizko eginbeharrak edo atazak sailkatzeko, hiru fase nagusi bereizi dira: kudeaketa, garapena eta dokumentazioa. Hau da, proiektuaren helburuak definitu eta antolakuntza aurrera eraman, proiektuaren ekintza praktikoak burutu finkatutako helburuak betez eta azkenik, horiek nola gauzatu diren eta lortutako emaitzak azaldu, hurrenez hurren.

Horietako fase bakoitza fase txikiagoetan sailkatu da, eta azken fase horiek, oinarrizko atazak barneratzen dituzte, hurrengo atalean ikus daitekeen bezalaxe.

6.1 LDE diagrama

Faseen sailkapena errazago ulertzeko, LDE (Lanaren Deskonposaketa Egitura) diagrama erabili da. Aipatutako faseak ondorengo diagraman ikus daitezkeenak dira:



6.1 Irudia: Proiektuaren faseen sailkapena (LDE diagrama)

6.2 Faseen deskribapena

Atal honetan, proiektuaren fase bakoitza xeheki azalduko da, zehaztasun guztiekin eta fase horietan barne diren atazak deskribatuz. Gainera, diagraman ikus daitezkeen hostoak, oinarrizko atazak izango dira, eta beraz, lanaren denbora estimazioa egiteko kontuan hartuko dira.

6.2.1 Kudeaketa

Kudeaketa faseak proiektuaren iraupen bera izango du. Hau da, proiektuarekin hasi bezain laster, lanaren antolakuntza eta plangintza egin beharko da, eta ondoren, egindako estimazioaren jarraipena eta kontrola eraman beharko da, ustekaberik gerta ez dadin. Izan ere, proiektuan zehar gerta daitezkeen denbora galtzeen arriskuak kontuan hartu behar dira, baina denbora gehien hasieran eskaini beharko zaio, plangintza egiterako orduan.

Irudian (6.1) ikus daitekeen bezalaxe, fase hau, bi fase txikiagotan bana daiteke. Alde batetik, plangintza, eta bestetik, jarraipena.

- **Plangintza:** atal honen lehen egitekoa proiektuaren helburuak identifikatzea da. Behin helburuak finkatuta daudela, proiektuaren ataza desberdinak zeintzuk diren adieraz daitezke, eta ondoren, horiei eskaini beharreko orduen estimazioaren kalkulua eta gerta daitezkeen desbideratzeen arriskuen zehaztapena egingo da.

- **Jarraipena:** atal honetan, plangintzan egindako estimazioak betetzen ari diren aztertu beharko da. Gainera, aurreikusi diren atzerapenak gertatzen ari diren, zenbateko eragina duten eta atzerapen horien arrazoiak azalduko dira. Honela, erraz jakingo da zenbat denbora eskaini zaion ataza bakoitzari momenturo.

  Horrez gain, proiektuaren tutoreekin astero biltzea adostu da. Horrela, sortu daitezkeen zalantzak azkar argitu ahalko dira, ustekabeak eta atzerapenak ekidinez, eta gainera, aurrera-pausoak maizago emango dira lan-karga hobeto banatuz.

6.2.2 Garapena

Fase honi eskainiko zaio denbora eta lan-karga gehien, proiektuaren helburuak lortzeko egin beharreko lan praktiko guztia barneratzen baitu.

Lehendabizi, fasea zati ezberdinetan sailkatu da: prestaketa, diseinua eta analisia. Gainera, zati horietako ataza bakoitza identifikatzea posible izan da eginiko bilerei esker.

- **Prestaketa:** proiektuan zehar erabiliko diren tresna desberdinei buruzko informazioa bilatu, ulertu eta praktikan jarriz barneratuko dira fase honetan. Hori dela eta, gaiari buruz egindako lan akademikoak eta baita webgune ofizialak dakarren dokumentazioa erabiliko dira.

  Prestaketa fase honetan ondorengo atazak identifikatu ditzakegu:

  - Monero CLI Wallet-aren instalazioa: instalazioaz gain, nodoa sarearekin konektatu beharra dago. Ataza honek beraz, informazioa bilatzea eta ekipoaren prestaketa barneratuko ditu.
  - Monero sarearen funtzionamendua: sarearen egiturari buruzko informazioaren bilaketa, P2P (ingeleseko *peer-to-peer* akronimotik) sarea eraikitzen da, parekoen arteko komunikazioak sortuz.

  - Levin protokoloa: Monero sareko nodoen artean informazioa partekatzeko bidaltzen diren mezuen ikerketa, dagoeneko existitzen den dokumentazioarekin. Mezuen goiburukoa eta helburuak aztertu, horiek nola sortu jakiteko.

- **Diseinua:** lortu nahi diren helburuak gauzatzeko, eraikiko diren programen lengoaiak eta diseinuak zehaztea premiazkoa da. Horren ostean, garapenean, hau da, programa horien inplementazioa egiterako orduan, eta egingo diren esperimentuekin lortuko diren emaitzak zuzenak izateko, diseinua zehatza eta eraginkorra izan beharko da.

  Diseinu fase honetan, zehaztasun maila handituz, ondorengo atazak gauzatuko dira:

  - Komunikazioa aztertu: Monero sarera konektatuta egonda, bidaltzen diren mezuak aztertzeko script-en programazioa: jasotako mezuak goiburukoaren arabera sailkatu eta goiburukoaren atal bakoitzari iruzkinak gehituko dizkio, bidalketa prozesua nola gauzatzen den ulertzeko.
  - Mezuak sortu: aztertutako Levin protokoloa jarraituz, mezu horiek sortu eta Monero sareko beste nodo batzuei eskaerak bidaltzeko tresnaren diseinua egin. Gainera, jaso nahi diren datuak lortu eta mapan kokatu.

- **Analisia:** fase honetan, diseinuarekin eta garatutako lanarekin eskuratutako emaitzen analisia aurrera eramango da. Emaitza zuzenak eskuratzea bezain garrantzitsua da horiek ondo interpretatzea eta azaltzea datu horiek testuinguru zehatz honetan daukaten esanahia.

  Fase honetan ataza hauek bereiz daitezke:

  - Datu orokorrak: Lortu diren emaitzak sailkatu, testuinguruan jarri, aztertu eta esanguratsuak izan daitezkeen datuak lortu.
  - IP helbideen mundu mailako kokapena: bizilaguntzat identifikatutako makina non dagoen jakingo dugu koordenatuekin, mapamundi batean kokatuz. Horrez gain, nodo bakoitzetik jasotako transakzioak irudikatuko dira, eta horren analisia egingo da.
  - Aurretik lortutako emaitzen konparaketa: Monero tresnaren inplementazioan aldaketak izan ondoren, aurreko lan batek zehazten zuena kontuan hartuz, lortutako emaitzek duten esanahia eta eragina analizatu sareari dagokionez.

6.2.3 Dokumentazioa

Fase honetan, proiektua dokumentatuko da, egin diren lan guztiak eta horiekin lortutako emaitzak azalduz. Helburua, lana edonori modu antolatuan aurkezteko aukera izatea da.

- **Memoria:** atal hau betetzeko, *Overleaf* tresna erabiliko da, proiektua era txukunean dokumentatu ahal izateko. Gainera, edizioa online denez, edozein momentuan, bai tutoreek eta baita ikasleak ere memoria atzitu ahalko dute.
- **Aurkezpena:** lanaren sintesia landuko da, aurkezpenean informazio esanguratsuena azaldu dadin, kontuan hartuz iraupena 15-20 minutu tartekoa izango dela. Hortaz, edukia eta hori aurkezteko era (estiloa, hizkuntza...) zainduko dira honakoan.

6.3 Lanen estimazioak eta desbiderapenak

Jarraian dagoen taulan ikus daiteke ataza bakoitzari eskaintzea pentsatu den (aurreikusitako) ordu kopurua eta benetan bakoitzarekin pasatako ordu kopurua.

	Aurreikusitako orduak	Benetako orduak
KUDEAKETA	**43 h**	**42 h**
Plangintza	28 h	30 h
Jarraipena	15 h	12 h
GARAPENA	**165 h**	**173 h**
Prestakuntza	22 h	26 h
Instalazioa	8 h	10 h
Funtzionamendua	7 h	7 h
Levin protokoloa	7 h	9 h
Diseinua	115 h	120 h
Komunikazioa	30 h	33 h
Programazioa	85 h	87 h
Analisia	28 h	27 h
Orokorra	7 h	8 h
Kokapena	12 h	10 h
Konparaketa	9 h	9 h
DOKUMENTAZIOA	**92 h**	**91 h**
Memoria	77 h	75 h
Aurkezpena	15 h	16 h
GUZTIRA	**300h**	**306 h**

6.1 Taula: Proiektuaren aurreikusitako eta benetako ordu kopurua

6.4 Arrisku plana

Proiektu bat egiterako orduan, ohikoa da atzerapenak sortuko dituzten ezustekoak gertatzea. Horregatik, garrantzitsua da dauden arriskuak identifikatzea, gertatzen badira ere, zer egin jakiteko eta atzerapenak sortzea saihesteko. Hau da, nahiz eta atzerapenak sortu, horiek ahalik eta azkarren konpontzea, proiektuaren hurrengo urratsen garapena ez baldintzatzeko.

Orokorrean, lan karga gehien hartzen duen atala gehien atzeratu daitekeena izaten da. Hortaz, atal horietan estimatzen den denbora eta ziurgabetasuna handitzen da, ordea ataza txikien estimazioa eta kontrola zehatzagoa izango dela espero da, eta beraz, horietan ez da hainbesteko arriskurik aurreikusten.

Gainera, hasierako faseetan garapeneko tresnak erabiltzen ikasi behar da eta hori egiteko ohituragatik, ezustekorik gertatzeko aukera txikia da. Horrez gain, lanak ongi dokumentatzeko ohitura eduki arren, gradu amaierako lanaren memoria egiterako orduan, luzera dela eta, atzerapenak sortzeko arriskua egon daiteke. Proiektuko datei dagokienez, ez da epeak betetzeko arriskurik aurreikusten.

Arrisku hauek proiektuaren garapenean ahalik eta eragin txikiena izan dezaten, desbiderapen bat gertatu bezain laster, ondorengo egunetako lana ere aurreratzeko saiakera egingo da, tutoreekin biltzerako orduan atazak eginak izateko. Atazaren batean denbora gehiegi dedikatu behar bada, atazaren bat laburtu beharko litzatekeen aztertuko da, proiektu osoaren iraupena kontrolatuz.

6.5 Lan-metodologia

Lan-metodologiari dagokionez, lan-karga asko handitu ez dadin eta arriskuak ekiditeko, tutoreekin astean behin biltzea erabaki zen. Horrela, zalantzak garaiz argitzeaz gain, egin beharreko hurrengo lanak denborarekin ulertu ahalko dira. Bilerak ostegun goizetan egitea erabaki zen, baina malgutasunarekin beste edozein momentuan biltzeko aukera izanez, zalantza puntualak sortzen badira, baita posta elektronikoz ere, ahal den neurrian zalantza horiek konpontzeko.

Proiektuaren ezaugarriak direla eta, garrantzi handia izango du ikerketa prozesuak. Hala ere, oztopo handiak sortzen badira, egitekoak gehiago zehaztuko dira atzerapenak txikituz.

Gainera, lanak egin ahala, programatu eta programa horiek ondo funtzionatzen dutela eta eraginkorrak direla egiaztatuko da, era praktikoan aplikatuz ikertuz graduan ikasi dena.

Gradu amaierako lana ekainean entregatzea erabaki zen, gainerako ikasgaiak une berean bukatzen direlako eta honela, lan guztiak lasaitasunez egiteko denbora egongo da. Izan ere, lanak egiteko beharrezkoa den astia hartzeak badu bere garrantzia, ikasitakoa gehiago barneratzen baita.

6.6 Desbiderapenen analisia

Aurreko ataleko taulan (6.1) ikus daitekeen bezalaxe, garapena atalerako estimatutakoa baino denbora gehiago behar izan da. Atal horren barruan, prestakuntza eta diseinuan egon da desbiderapen handiena, une hartan gaiari buruzko esperientzia faltarengatik. Gainera, diseinuak zuen proiektu osoko pisu handiena, eta ohikoa den bezala, pisu handiko atazek desbiderapenak gertatzeko arrisku handiagoa dute.

Horrez gain, Monero tresnarekin lehen kontaktua zenez, desbiderapen garrantzitsua egon da instalazioa modu egokian egin arte eta Levin protokoloak zehaztutakoa menperatu arte.

Tutoreekin bilerak hasieran astero baziren ere, azken hilabetean bi astean behin biltzea adostu zen, egin beharreko lana argiago baitzegoen. Hau da, geroz eta behar gutxiago egon da bilerak egiteko, eta hala ere, modu arrakastatsuan eta epeak errespetatuz garatu da lana.

7. KAPITULUA

Ondorioak

Monero aplikazioak sortzen duen P2P sarean nodoak nola komunikatu eta horien portaera hartzea lortu da lan honekin. Nodoen arteko komunikazioari buruzko informazio gutxi egon arren, probak egin ostean, une bakoitzean bidaltzen den informazioa lortu da. Hori kontuan hartuta, C lengoaiako sare-programazioarekin, komunikazioa era konkurrentean betetzea, eta Python liburutegi batzuk erabiliz, nodo horiek (eta bakoitzetik jasotako transakzio kopurua) mapan kokatzea lortu da. Hala ere, mapan kokatzeaz arduratzen den programaren exekuzioak lan-karga handia sortzen du prozesadorean (gama ertaineko ordenagailuan behintzat).

Sareari buruz, aipatzekoa da jasotako transakzioak nahiko banatuta egon direla munduan zehar, nahiz eta munduan zehar duten banaketa Estatu Batuetan eta Europan biziagotzen den. Korrelazioa gordetzen du munduaren kontaminazio luminikoarekin, Txinaren kasuan izan ezik. Nodo gutxi aurkitu dira Txinan, baina izan liteke Gobernu horrek dibisa digital hau erabiltzea debekatu izanagatik gertatzea, edo pribatutasun zerbitzuak erabiltzen egotea eta aurkitutako kokapena edo IP helbidea ez izatea nodoarena errealitatean, baizik eta segurtasun-maila baten bidez lortutakoa. Monero tresnak dirua modu anonimoan erabiltzea ahalbidetzen duenez, ohikoa da ezkutuko internet zerbitzuetan erabiltzea, Bitcoin bezalako kriptotxanponek arrastoa uzten baitute eta horiek jarraitzen erabiltzailea aurki daiteke.

Proiektuaren helburuak lortzeko hasierako informazioa urria izan arren, helburuak praktikoki lortzeko aukera izan dut. Neure burua kriptodiru teknologian murgiltzea eta C eta Python programazio lengoaietan ebazpena aurkitzeak, ikuspegi orokorra eman dit informatikako proiektuei buruz, eta programatzen trebatzea ahalbidetu dit.

Pribatutasuna mantentzen duen Monero tresna honen Levin aplikazio-mailako protokoloa barneratzea (atzeranzko ingeniaritza aplikatuz lortutako informazioarekin, Github-en dokumentatuta dagoenaz gain) eta beste nodoekin komunikatzen jakitea, positiboa iruditzen zait. Garapenean sareko programazioa aurrera eramaten trebatu naiz. Gainera, dibisakriptografikoek erabiltzen dituzten mekanismoei buruz ikastea interesgarria egin zait.

Informatika Ingeniaritza Graduko hainbat irakasgaietan ikasitakoa aplikatu ahal izan da proiektu hau garatzerako orduan. Programazio lengoaietan trebatzea ahalbidetu duten irakasgaiez gain, sareko protokoloak ulertzen lagundu duten funtsezko irakasgaiak Sare Zerbitzuak eta Aplikazioak (SZA) eta Konputagailuen Sareen Oinarriak (KSO) izan dira, proiektu honen kasuan oso ezaguna ez den Levin aplikazio-mailako protokoloa aztertzeko balio izan dutenak. Gainera, Programazio Konkurrentea (PRK) irakasgaian lortutako gaitasunekin, tresna konkurrente eta eraginkorra programatu ahal izan da.

Azkenik, gradu amaierako proiektua astero tutoreekin bilduz baina modu indibidualean egiteak ahalmen berriak eman dizkidala uste dut, informatikako proiektu edo baliabide berriak errazago barneratzea eta erabiltzen ikastea.

A. ERANSKINA

Moneroren hasieraketa

Jarraian, Monero erabiltzen hasteko eman beharreko urratsak azalduko dira; instalazioa eta sinkronizazioa bereiziz. Erabilitako sistema eragilea Linux-en Ubuntu 18.04 izan da, baina gainerako sistema eragileetarako eman beharreko urratsak oso antzekoak dira eta Moneroren webgune ofizialeko gida jarraitzearekin nahikoa da; instalazioaz gain, erabiltzaileen funtzionamenduaren alderdi garrantzitsuenak azaltzen dira.

Horrez gain, proiektu hau aurrera eramateko, Monero aplikazioaren komando lerroko interfaze bertsioa (CLI, ingelesez *command-line interface*) erabili da, interfaze grafikoa (GUI, ingelesez *graphical user interface*) eskaintzen duenaren ordez. Garrantzitsua da aplikazioa eguneratuta mantentzea azken bertsioak jaitsi eta erabiliz, aurrekoetan gerta daitezkeen erasoak saihesten direlako.

Helburua Monero martxan jartzea denez, GnuPG tresna dagoeneko instalatuta dagoela aintzat hartuko da, baina hau instalatzeko gida ere Moneroren web-orrialde ofizialean badago.

Garrantzitsua da urrats hauek hertsiki jarraitzea deskargatu dugun aplikazioa infektatua ez dagoela egiaztatzeko eta segurtasun osoz erabiltzeko.

A.1 Instalazioa

Monero erabiltzeko helburuak desberdinak izan daitezke (kontuak sortu eta kudeatu, transakzioak egin, tresnaren garapenean laguntza eman...), baina erasoetan ez erortzeko eta tresna segurtasunez erabiltzeko, deskargatu dena espero den fitxategia dela egiaztatu behar da.

Hori dela eta, deskargatzen diren fitxategiak egiaztatu behar dira; bost urrats nagusietan banatu da instalazio prozesua, baina behar bezala erabiltzeko sinkronizazio atalean zehaztutako urratsak ere eman beharko dira.

A.1.1 Gakoa lortu

Horretarako, hurrengo komandoa terminalean exekutatuz, binaryFate-en GPG gakoa uneko direktorioan deskargatuko da:

```
$ wget -O binaryfate.asc https://raw.githubusercontent.com/monero-project/monero/master
  /utils/gpg_keys/binaryfate.asc
```

Jaitsitako `binaryFate.asc` fitxategian Moneroren gako pribatua aurki daiteke eta inportatu aurretik horren baliozkotasuna egiaztatzea beharrezkoa da.

Ondoren, hatz-marka egiaztatzeko komando hau exekutatuko da:

```
$ gpg --keyid-format long --with-fingerprint binaryfate.asc
```

Emaitzak horrelako itxura izango du:

```
pub   rsa4096/F0AF4D462A0BDF92 2019-12-12 [SCEA]
      Key fingerprint = 81AC 591F E9C4 B65C 5806  AFC3 F0AF 4D46 2A0B DF92
uid                           binaryFate <binaryfate@getmonero.org>
```

Lortutako irteeran, *Key fingerprint* balioak bat etorri behar du hor agertzen denarekin, bestela `binaryFate.asc` fitxategia ezabatu eta hasierara itzuli beharko da.

A.1.2 Sinadura gakoa inportatu

Terminalean ondorengo komandoa exekutatuko da:

```
$ gpg --import binaryfate.asc
```

Exekutatzen den lehen aldia bada horrelako erantzuna lortuko dugu:

```
gpg: key F0AF4D462A0BDF92: 2 signatures not checked due to missing keys
gpg: key F0AF4D462A0BDF92: public key "binaryFate <binaryfate@getmonero.org>" imported
gpg: Total number processed: 1
gpg:               imported: 1
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
```

Aldiz, noizbait gakoa inportatu bada, horrelakoa izango da komandoaren irteera:

```
gpg: key F0AF4D462A0BDF92: "binaryFate <binaryfate@getmonero.org>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1
```

A.1.3 Hash kodeen fitxategia deskargatu eta egiaztatu

Komando hau exekutatuz, `hashes.txt` fitxategia lortuko dugu uneko direktorioan:

```
$ wget -O hashes.txt https://www.getmonero.org/downloads/hashes.txt
```

Egiaztatu deskargatu den fitxategia komando honen bidez:

```
$ gpg --verify hashes.txt
```

Hurrengo emaitza lortu beharko da aurrera jarraitzeko, ordea emaitza ondorengoaren desberdina bada, `hashes.txt` fitxategia ezabatu eta berriro deskargatuz saiatu beharko da:

```
gpg:                using RSA key 81AC591FE9C4B65C5806AFC3F0AF4D462A0BDF92
gpg: Good signature from "binaryFate <binaryfate@getmonero.org>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 81AC 591F E9C4 B65C 5806  AFC3 F0AF 4D46 2A0B DF92
```

A.1.4 Monero deskargatu eta egiaztatu

Monero fitxategi bitarra lortzeko hurrengo komandoa exekutatuko da, agian ondorengoa baino bertsio berriagoa egongo da webgunean atzigarri:

```
$ wget -O monero-linux-x64-v0.17.1.9.tar.bz2 https://downloads.getmonero.org/cli/linux64
```

Deskargatutako fitxategi trinkotua erauzi aurretik, komando honekin egiaztatuko da:

```
$ shasum -a 256 monero-linux-x64-v0.17.1.9.tar.bz2
```

Lortutako irteera honelakoa izango da:

```
0fb6f53b7b9b3b205151c652b6c9ca7e735f80bfe78427d1061f042723ee6381
monero-linux-x64-v0.17.1.9.tar.bz2
```

Lortutako irteera `hashes.txt` fitxategian dagoenarekin bat etorri behar du, eta beraz, adibide honen desberdina izan daiteke. Bestela, Moneroren fitxategi bitarra ezabatu eta berriro deskargatu beharko da, egiaztapen hau errepikatuz eta hash kode bera lortu arte.

A.1.5 Monero erauzi

Beraz, egiaztatu da deskarga zuzena izan dela eta fitxategi trinkotua erauziko da:

```
$ tar -jvxf monero-linux-x64-v0.17.1.9.tar.bz2
```

Azkenik, komando honekin lortu ditugun fitxategi exekutagarriak ikus daitezke:

```
$ ls monero-linux-x64-v0.17.1.9/monero-x86_64-linux-gnu-v0.17.1.9/
```

```
monero-blockchain-depth       monero-blockchain-import       monero-blockchain-prune
monero-blockchain-stats       monerod                        monero-gen-trusted-multisig
monero-blockchain-ancestry    monero-blockchain-export       monero-gen-ssl-cert
monero-blockchain-mark-spent-outputs            monero-blockchain-prune-known-spent-data
monero-blockchain-usage       monero-wallet-rpc              monero-wallet-cli
```

A.2 Sinkronizazioa

Hainbat modutan sinkronizatu daiteke nodoa sareko gainerako nodoekin funtzionatzeko. Jakina da, nodo hauek beraien artean informazio nahikoa partekatzen dutela, sarearen funtzionamendua modu autonomoan babesteko. Izan ere, Moneroren transakzio guztiak agertzen diren liburu digitala publikoa da, edonork kontsulta dezake eta transakzio berriak gauzatzean, hainbat nodoren artean zuzena den egiaztatzen da, sinadura digitalen bidez.

Horregatik, nodoak ahalik eta modu autonomoenean lan egiteko, makina lokalean transakzio liburu (*blockchain*) osoa izan beharko dugu deskargatuta. Prozesu honek, hainbat egun iraun ditzake, erabiltzen den biltegi gailuaren arabera; ikusiko den moduan, disko gogorra (HDD) erabiliz denbora gehiago hartuko du, egoera solidoko unitatea (SSD) erabiltzearekin konparatuta.

Bestetik, nodo lokala beste nodo batera konektatu daiteke *blockchain*a deskargatu behar gabe, baina modu honetan beste nodoaren menpe egongo da nodo lokalaren segurtasuna eta beharrezkoa litzake konfiantza osoa izatea kanpoko nodo horretan. Modu hau ondorioz, ez da gomendatzen.

Azken aukera, transakzio liburu zati bat deskargatzea da, sinkronizatzeko beharrezkoa den informazioa izango da nodo lokalean, baina hainbeste memoria erabili behar gabe. Izan ere, unean Moneroren *blockchain*ak 100 GB baino gehiago okupatzen ditu. Aldiz, modu honetan, egituraren informazio ez-kritikoa baztertzen da eta gutxi gorabehera 30 GB-eko tamaina izango du, espazio asko aurreztuz. Horrez gain, beharrezko informazio guztia izanda, nodo lokalean segurtasun osoarekin lan egitea lortuko da, eta denbora tarte txikiagoan sinkronizatuko da nodoa sarearekin.

A.2.1 Nodo osoa

Nodo honetan *blockchain*aren egitura osoa deskargatuko da, uneoro nodoaren segurtasuna bere menpe egon dadin.

Aipatzekoa da, `--data-dir direktorio` argumentuarekin *blockchain*-a non gordeko den zehaztu daitekeela, baina direktorio lehenetsia `"$HOME/.bitmonero/"` izango da Linux-en kasuan eta `"C:\ProgramData\bitmonero\"` direktorio ezkutua Windows-en kasuan.

Bigarren argumentuarekin, sinkronizazio prozesua bukatu bitartean (atzeko planoan), kan-

poko nodo baten informazioa erabil daiteke, tresna denbora tarte txikian erabiltzen hasi ahal izateko. Hala ere, sinkronizazioa burutu arte, ez da desiratutako segurtasun maila lortuko, baizik eta adierazitako nodoaren menpe egongo da.

Azkenik, `--restricted-rpc` argumentuarekin beste nodoak lokalera konektatzea galaraziko da.

```
$ ./monerod  --restricted-rpc  --bootstrap-daemon-address=node.moneroworld.com:18089
```

Lortuko den irteera honelakoa izango da sinkronizatzen dagoen bitartean:

```
2020-11-19 11:35:22.053 I Monero 'Oxygen Orion' (v0.17.1.3-release)
2020-11-19 11:35:22.053 I Initializing cryptonote protocol...
2020-11-19 11:35:22.053 I Cryptonote protocol initialized OK
2020-11-19 11:35:22.054 I Initializing core...
2020-11-19 11:35:22.054 I Loading blockchain from folder /media/iperez/TOSHIBA EXT/monero_bc/
 lmdb ...
2020-11-19 11:35:22.054 W The blockchain is on a rotating drive: this will be very slow, use
 an SSD if possible
2020-11-19 11:35:22.198 I Loading checkpoints
2020-11-19 11:35:22.199 I Core initialized OK
2020-11-19 11:35:22.199 I Initializing p2p server...
2020-11-19 11:35:22.210 I p2p server initialized OK
2020-11-19 11:35:22.210 I Initializing core RPC server...
2020-11-19 11:35:22.716 I Binding on 127.0.0.1 (IPv4):18081
2020-11-19 11:35:23.577 I core RPC server initialized OK on port: 18081
2020-11-19 11:35:23.578 I Starting core RPC server...
2020-11-19 11:35:23.578 I core RPC server started ok
2020-11-19 11:35:23.636 I Starting p2p net loop...
**********************************************************************
2020-11-19 11:35:24.638 I The daemon will start synchronizing with the network. This may take
  a long time to complete.
2020-11-19 11:35:24.638 I
2020-11-19 11:35:24.638 I You can set the level of process detailization through "set_log <
 level|categories>" command,
2020-11-19 11:35:24.638 I where <level> is between 0 (no details) and 4 (very verbose), or
 custom category based levels (eg, *:WARNING).
2020-11-19 11:35:24.639 I Use the "help" command to see the list of available commands.
2020-11-19 11:35:24.639 I Use "help <command>" to see a command's documentation.
**********************************************************************
2020-11-19 11:35:25.778 I [167.172.150.102:18080 OUT] Sync data returned a new top block
 candidate: 98568 -> 2233926 [Your node is 2135358 blocks (6.4 years) behind]
2020-11-19 11:35:25.778 I SYNCHRONIZATION started
2020-11-19 11:35:33.648 I Synced 98668/2233926 (4%, 2135258 left)
2020-11-19 11:35:34.113 I Synced 98768/2233926 (4%, 2135158 left)
2020-11-19 11:35:34.635 I Synced 98868/2233926 (4%, 2135058 left)
```

Argi adierazten da lehenengo mezuetan sinkronizazio prozesua luzea izan daitekeela, eta posible bada SSD unitatea erabiltzea. Irteeran, data eta gero agertzen den zutabean, mezuaren maila adierazten da. I: mezu informatiboa, W: kontuan hartu beharreko mezua eta E: errore mezua. Ondoren, mezuaren deskribapena irakur daiteke; sarearen hasieraketa,

nodoaren uneko egoera, etab. Gainera, komandoak exekutatzeko aukera ematen du, `help` komandoaren bidez gehiago ezagutzeko aukera eskainiz.

A.2.2 Inausitako nodoa

Modu honetan, ez dugu egitura osoa deskargatuko eta *bootstrap* modua gaitu daiteke, nahiz eta lan honetan aurreko sinkronizazio modua erabili den, hurrengo komandoarekin hasiko genuke sinkronizazio modu hau Moneroren kokapen lehenetsian:

```
$ ./monerod --bootstrap-daemon-address=node.moneroworld.com:18089 --prune-blockchain
  --restricted-rpc
```

Antzeko irteera lortuko da kasu honetan:

```
2021-01-23 18:34:36.287 I Monero 'Oxygen Orion' (v0.17.1.9-release)
2021-01-23 18:34:36.287 I Initializing cryptonote protocol...
2021-01-23 18:34:36.288 I Cryptonote protocol initialized OK
2021-01-23 18:34:36.296 I Initializing core...
2021-01-23 18:34:36.314 I Loading blockchain from folder /home/iperez/.bitmonero/lmdb ...
2021-01-23 18:34:36.317 W The blockchain is on a rotating drive: this will be very slow, use
 an SSD if possible
2021-01-23 18:34:36.665 I Loading checkpoints
2021-01-23 18:34:36.666 I Pruning blockchain...
2021-01-23 18:34:36.666 I Core initialized OK
2021-01-23 18:34:36.677 I Initializing p2p server...
2021-01-23 18:34:36.748 I p2p server initialized OK
2021-01-23 18:34:36.761 I Initializing core RPC server...
2021-01-23 18:34:37.729 I Binding on 127.0.0.1 (IPv4):18081
2021-01-23 18:34:38.105 I core RPC server initialized OK on port: 18081
2021-01-23 18:34:38.208 I Starting core RPC server...
2021-01-23 18:34:38.209 I core RPC server started ok
2021-01-23 18:34:38.261 I Starting p2p net loop...
**********************************************************************
2021-01-23 18:34:39.406 I The daemon will start synchronizing with the network. This may take
  a long time to complete.
2021-01-23 18:34:39.406 I You can set the level of process detailization through "set_log <
 level|categories>" command,
2021-01-23 18:34:39.406 I where <level> is between 0 (no details) and 4 (very verbose), or
 custom category based levels (eg, *:WARNING).
2021-01-23 18:34:39.406 I Use the "help" command to see the list of available commands.
2021-01-23 18:34:39.406 I Use "help <command>" to see a command's documentation.
**********************************************************************
2021-01-23 18:34:39.552 I [51.75.159.19:3706 OUT] Sync data returned a new top block
 candidate: 1 -> 2281075 [Your node is 2281074 blocks (6.8 years) behind]
2021-01-23 18:34:39.552 I SYNCHRONIZATION started
2021-01-23 18:34:47.472 I Synced 101/2281075 (0%, 2280974 left)
2021-01-23 18:34:47.535 I Synced 201/2281075 (0%, 2280874 left)
2021-01-23 18:34:47.602 I Synced 301/2281075 (0%, 2280774 left)
```

B. ERANSKINA

Garatutako tresnaren hasieraketa

Jarraian, garatutako tresna Ubuntu sistema eragilearen gainean erabiltzen hasteko instalatuta eduki behar diren pakete eta programen instalazioa nola gauzatu deskribatuko da.

Lehendabizi, pakete kudeatzaileak duen informazioa eguneratuko da hurrengo komandoa exekutatuz:

```
$ sudo apt-get update
```

Proiektua uneko makinan deskargatzeko git tresna beharko da, eta instalatuta ez badago, ondorengo komandoa exekutatuz instalatuko da:

```
$ sudo apt-get install git
```

Programa nagusia konpilatu ahal izateko gcc konpiladorea instalattuta egon behar da, ondorengo komandoarekin instalatuko da:

```
$ sudo apt-get install gcc
```

Hurrengo komandoarekin `geoiplookup` komandoa instalatuko da:

```
$ sudo apt-get install geoip-bin
```

Ondorengo komandoarekin proiektuaren informazio osoa deskargatuko da uneko direktorioan:

```
$ git clone https://github.com/iperezz97/monero-sarea-aztertzen
```

Proiektuaren ‘kodea’ karpetan kokatuta, ‘geoip’ karpeta gehitu:

```
$ cd monero-sarea-aztertzen/kodea/ && mv ../geoip ./
```

Programa nagusia konpilatzeko hurrengo komandoa exekutatu:

```
$ gcc nagusia.c bzb.c eskatu1001.c konprob1003.c kokatu_mapan.c jaso2002.c -lpthread -o nagusia
```

Dena prest egongo litzateke programa nagusia exekutatzeko, laguntza kasua ikusi nahi bada:

```
$ ./nagusia
```

Emaitzak horrelako itxura izango du:

```
Helburuko IP helbidea, portua eta eskaeren denbora muga (seg) beharrezkoak dira exekuzioa hasteko
Helburuko nodoa ondorengo zerrendatik aukeratu dezakezu:
  - 66.85.74.134 18080
  - 88.198.163.90 18080
  - 95.217.25.101 18080
  - 104.238.221.81 18080
  - 192.110.160.146 18080
  - 209.250.243.248 18080
  - 212.83.172.165 18080
  - 212.83.175.67 18080
```

Laguntza kasuan emandako informazioarekin hurrengo proba egin daiteke:

```
$ ./nagusia 212.83.175.67 18080 10
```

Hamar segundutan aurkitutako nodo kopurua eta haien koordenatuak pantailan inprimatuko dira, baina exekuzioaren helburua datu horiek *pipe*-aren bidez mapan kokatzeko programari pasatzea izango da. Denbora hori pasatzerakoan, atzigarri aurkitu diren nodoetatik transakzioak jasotzen saiatuko da, eta informazio hori ere mapan adierazi nahi da. Horretarako, Python programak dituen dependentziak instalatzen jarraituko da.

Oraingoan, `pip3` instalatzailea erabiliko dugu, ondorengo komandoa exekutatuz instalatuko dena (gida honetan python3 erabili bada ere, berdina da python-entzat):

```
$ sudo apt-get install python3-pip
```

```
$ pip3 install numpy
```

```
$ pip3 install pandas
```

```
$ pip3 install matplotlib
```

```
$ pip3 install libgeos-dev
```

```
$ sudo apt-get install python3-gi-cairo
```

```
$ pip3 install Cython
```

```
$ pip3 install --upgrade pip
```

```
$ pip3 install --user https://github.com/matplotlib/basemap/archive/master.zip
```

Proba txiki bat egin daiteke python programak funtzionatzen duen ikusteko:

```
$ python3 kokatu_mapan.py
```

Proiektua era zuzenean exekutatzeko, bi programen exekuzioa hasieratuko da:

```
$ ./nagusia 212.83.175.67 18080 10 | python3 kokatu_mapan.py
```

Exekuzioak sortuko duen irteerak honelako itxura izango du:

```
Koordenatuak irakurtzen...
<Eragiketa> <Longitude> <Latitude> <IP> <City/Tr>:
Transakzioak jasotzen...
Jasotako transakzioak:  14   ( 24.10.144.178 14 )
```

Bibliografia

[Tong Cao, Jiangshan Yu, Jérémie Decouchant, Xiapu Luo, and Paulo Verissimo, 2019] Tong Cao, Jiangshan Yu, Jérémie Decouchant, Xiapu Luo, and Paulo Verissimo (2019). Exploring the Monero Peer-to-Peer Network. https://eprint.iacr.org/2019/411.pdf. Noiz atzitua: 2021-01.

[Monero, 2014] Monero (2014). Monero proiektuaren web-orria. https://www.getmonero.org/. Noiz atzitua: 2021-01.

[Monero, 2020] Monero (2020). Monero proiektuaren dokumentazioa. https://monerodocs.org/. Noiz atzitua: 2021-02.

[Monero, 2021] Monero (2021). Monero hazi-nodoak. https://community.xmr.to/xmr-seed-nodes. Noiz atzitua: 2021-02.

[Monero, 2021] Monero (2021). Monero mundua. https://moneroworld.com/. Noiz atzitua: 2021-02.

[Monero, 2021] Monero (2021). Github-en Monero proiektua. https://github.com/monero-project/monero. Noiz atzitua: 2021-02.

[Monero, 2021] Monero (2021). Tcpflow tresnaren manuala. https://linux.die.net/man/1/tcpflow. Noiz atzitua: 2021-02.

[Brian “Beej Jorgensen” Hall, 2020] Brian “Beej Jorgensen” Hall (2020). Sare-programazio gida. http://beej.us/guide/bgnet/html/. Noiz atzitua: 2021-02.

[Monero, 2020] Monero (2020). Levin protokoloaren dokumentazioa. https://github.com/monero-project/monero/blob/master/docs/LEVIN_PROTOCOL.md.
Noiz atzitua: 2021-02.

[Robert S. Barnes, 2010] Robert S. Barnes (2010). Mezuak jasotzeko denbora muga ezarri. https://stackoverflow.com/questions/2876024/linux-is-there-a-read-or-recv-from-socket-with-timeout. Noiz atzitua: 2021-03.

[Jay Sullivan, 2020] Jay Sullivan (2020). Konexioa ezarri denbora mugarekin. https://stackoverflow.com/questions/2597608/c-socket-connection-timeout. Noiz atzitua: 2021-03.

[“Carun”, 2019] “Carun” (2019). Hainbat hariek portu bera erabili konexioetan. https://gist.github.com/carun/8146981. Noiz atzitua: 2021-04.

[“cdiv1e12”, 2020] “cdiv1e12” (2020). Levin bezeroa inplementatzen duen Python programa. https://github.com/cdiv1e12/py-levin/tree/9b213f94a72769592d444d17538e1bbca29db888. Noiz atzitua: 2021-04.

[“user36303”, 2020] “user36303” (2020). Monero transakzioen tamaina. https://monero.stackexchange.com/questions/211/how-much-larger-are-monero-transactions-compared-to-the-average-bitcoin-transact . Noiz atzitua: 2021-05.