eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

TELEKOMUNIKAZIO TEKNOLOGIAREN INGENIARITZAKO GRADUA

# GRADU AMAIERAKO LANA

# *IoT INGURUNEETAKO ZIBERSEGURTASUN TEKNIKEN ERABILGARRITASUNA HOBETZEKO MEKANISMOAK*

**Ikaslea**: Sanz, Rekalde, Ane

**Zuzendaria (1):** Astorga, Burgo, Jasone

**Zuzendaria (2):** Huarte, Arrayago, Maider

**Ikasturtea:** 2018-2019

**Data:** Bilbo, 2019ko ekainaren 26a

Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

# Laburpena

IoT soluzioen inplementazio orokorra egotea eragozten duen arazo nagusietako bat sarbide-kontrola da. Izan ere, segurtasun soluzio tradizionalak ezin dira zuzenean IoT inguruneetan erabili, gailuek dituzten baliabideen mugak direla eta. IoTrako sortu diren berariazko soluzioen artean, Hidra sarbide-kontrol protokoloak ikuspegi desberdin eta eraginkor bat eskaintzen du. Gainera, IoT inguruneetan beharrezkoak diren espresibitatea eta malgutasuna bermatzen ditu, egoera tradizionaletan lortzen direnen antzekoak. Hau guztia lortzeko, segurtasun politika konplexuak definitzen dira, eta politika hauek lengoaia espezifiko baten arabera kodifikatzen dira. Hala ere, lortzen den sistema oso konplexua da, eta segurtasunean ez adituak direnentzat erabiltzeko zailegia izan daiteke. Hori dela eta, soluzio honen hedapena oztopatzen da.

Ondorioz, proiektu honetan IoT inguruneetako segurtasun mekanismoak erabilgarriagoak bihurtzeko hainbat hobekuntza garatuko dira. Hobekuntza hauei esker, baimena duen edozein pertsonak erabili ahal izango ditu segurtasun mekanismoak, haien segurtasuneko jakintza-maila edozein izanda ere. Garatuko diren mekanismoak hurrengoak dira: erabiltzaile interfaze grafiko bat segurtasun politika berriak definitzeko; eta kodifikatzaile/dekodifikatzaile modulu bat politika horiek *“human-friendly”*-tik “*machine-friendly”*-ra itzultzeko eta alderantziz.

**Hitz gakoak:** IoT, segurtasuna, sarbide-kontrola, erabilgarritasuna

# Resumen

La seguridad, y el control de acceso en particular, es uno de los principales inconvenientes para una implementación masiva de las soluciones IoT, ya que las soluciones de seguridad tradicionales no pueden ser utilizadas directamente en los entornos IoT debido a las limitaciones de recursos de los dispositivos. Entre las soluciones creadas específicamente para IoT, el protocolo de control de acceso Hidra propone un enfoque eficiente y proporciona a los entornos IoT la flexibilidad y expresividad necesarias, equiparables a las conseguidas en entornos tradicionales. Esto se consigue gracias a la definición de políticas de seguridad complejas y a su codificación acorde a un lenguaje de política específico. Aun así, el sistema resultante es muy complejo y resulta muy difícil para ser usado por personas no expertas en seguridad, dificultando así su adopción.

Por lo tanto, en este proyecto se desarrollan varias mejoras para dotar de usabilidad a los mecanismos de seguridad, y así conseguir que puedan ser utilizados por cualquier persona sea cual sea su nivel de conocimientos en seguridad. Los mecanismos que se van a desarrollar son una interfaz gráfica *"user-friendly"* para definir nuevas políticas de seguridad y un módulo codificador/decodificador para convertir esas políticas de *"human-friendly"* a *"machine friendly"* y viceversa.

**Palabras clave:** IoT, seguridad, control de acceso, usabilidad

# Abstract

Security, and access control in particular, is one of the main drawbacks to the broad adoption of IoT solutions, as traditional solutions cannot be directly used in IoT environments due to the significant resource constraints of the targeted devices. Among security solutions specifically tailored to IoT, the Hidra access control mechanism stands out as an efficient approach to bring to the IoT world the flexibility and expressiveness achieved in traditional computing environments. This is fulfilled by the definition of complex security policies and their codification using a specific policy language. Unfortunately, the resulting system becomes complex and difficult to use by non-security experts, hindering its broad adoption.

Therefore, this project presents some enhancements to make powerful security mechanisms usable by human beings whatever their expertise: a user-friendly mechanism to create and manage security policies, and an encoder/decoder module to convert security policies from human-friendly to machine-friendly format and vice versa.

**Key words:** IoT, security, access control, usability

# Aurkibidea

# Irudien zerrenda

# Taulen zerrenda

# Akronimoen zerrenda

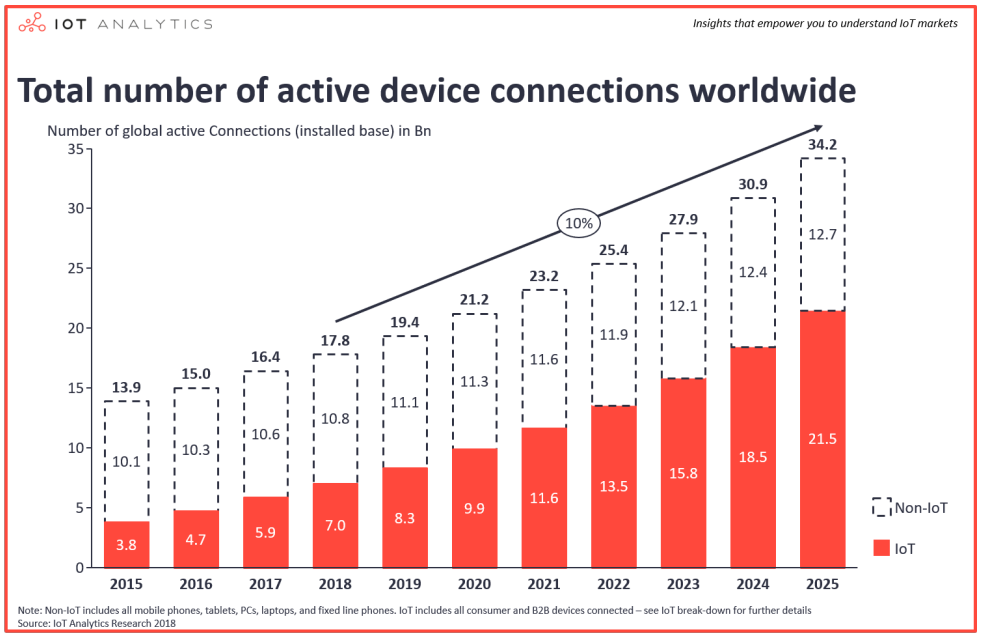
| | |
|---|---|
| 6LoWPAN | IPv6 over Low power Wireless Personal Area Network |
| ACS | Access Control Server |
| APBR | Authorization Policy Binary Representation |
| CapBAC | Capability Based Access Control |
| CBOR | Concise Binary Object Representation |
| CDS | Constrained Device Sensor |
| CM | Credential Manager |
| DAO | Destination Advertisement Object |
| DAO-ACK | Destination Advertisement Object Acknowledgement |
| DCAF | Delegated CoAP Authentication and authorization Framework |
| DCapBAC | Distributed Capability Based Access Control |
| DIO | DODAG Information Object |
| DIS | DODAG Information Solicitation |
| DODAG | Destination Oriented Directed Acyclic Graph |
| DTLS | Datagram Transport Layer Security |
| E2E | End to End |
| FFD | Full Function Device |
| HC | Header Compression |
| IEEE | Institute of Electrical and Electronics Engineer |
| IETF | Internet Engineering Task Force |
| IoT | Internet of Things |
| ISM | Industrial, Scientific and Medical |
| IPv4 | Internet Protocol version 4 |
| IPv6 | Internet Protocol version 6 |
| JavaEE | Java Enterprise Edition |
| JDBC | Java Database Connectivity |
| LLN | Low power and Lossy Networks |

| | |
|---|---|
| LR-WPAN | Low Rate Wireless Personal Area Network |
| LoWPAN | Low power Wireless Personal Area Network |
| MAC | Media Access Control |
| MTU | Maximum Transmission Unit |
| OF | Objective Function |
| PAN | Personal Area Network |
| PHP | Hypertext Preprocessor |
| RAM | Random Access Memory |
| RFD | Reduced Function Device |
| RPL | Routing Protocol for LLNs |
| TGT | Ticket Granting Ticket |
| UDP | User Datagram Protocol |
| WSN | Wireless Sensor Network |
| XACML | Extensible Access Control Markup Language |
| XML | Extensible Markup Language |

# 1. Sarrera

IoT (*Internet of Things*) datuak era masiboan bidaltzeko, jasotzeko eta biltegiratzeko gaitasuna duten gailu desberdinak elkar konektatzen dituen kontzeptua da. IoT-ren oinarri nagusietako bat datuak jasotzeko, prozesatzeko eta komunikatzeko erabiltzen diren gailuak gaitasun txikiko gailuak direla da. Gaitasun mugatu horiek alderdi desberdinetakoak izan daitezke: memoria gaitasuna, prozesadorearen gaitasuna, energia-iturria… Gailu hauen artean gaitasun mugatuko sentsoreak (CDS; *Constrained Device Sensor*) aipatzekoak dira, sentsore hauei esker era askotako datuak jaso daitezkeelako, ondoren hainbat aplikaziotan erabili ahal izateko.

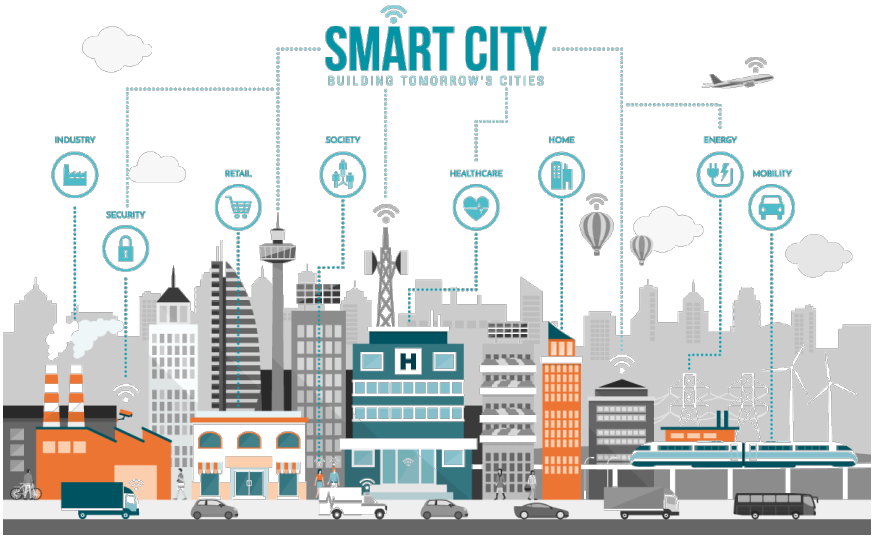
Gaur egun gero eta gehiago erabiltzen da IoT, izan ere, gero eta gailu gehiago daude Internetera konektatuta, eta kopuru horrek datorren urteetan gora egiteko joera izango duela uste da. Izan ere, IoT Analytics-ek argitaratutako datuen arabera [1] 2018. urtean 17 bilioi gailu zeuden konektatuta Internetera, eta horietatik 7 bilioi IoT gailuak ziren. Horrez gain, 2025. urterako 21 bilioi IoT gailu konektatuta egotea espero da, beraz, argi geratzen da IoT jasaten ari den igoera oso nabarmena dela.




*Irudia 1: Internetera konektatuta dauden gailu kopurua*

Aipatu den bezala, IoT sisteman CDS kopuru handia dago inplementatuta, eta hauek informazioa jaso eta bidaltzen dute haririk gabeko sareak sortuz. Sare hauei WSN (*Wireless Sensor Network*) deitzen zaie, eta sare hauen inplementazio eta hedapenari esker, mota askotako IoT aplikazioak ahalbidetu dira. Adibide moduan, *SmartCity* [2] teknologian mota honetako sareak erabiltzen dira, eta horiei esker posible da osasun zerbitzuak hobetzea, etxeetako gailuen urruneko kontrola izatea edota hirietako semaforo guztien kontrol automatikoa izatea, beste hainbat gauzen artean. *SmartCity*-en aplikazio eremuak 2. irudian adierazten dira. Beste aplikazio baten adibidea *Industry*

*4.0* da, eta honek produktuak denbora tarte laburragoan ekoiztea, bezeroen beharrizanetara hobeto egokitzea edo produktuen zerbitzuak hobetzea ahalbidetzen du, adibidez.



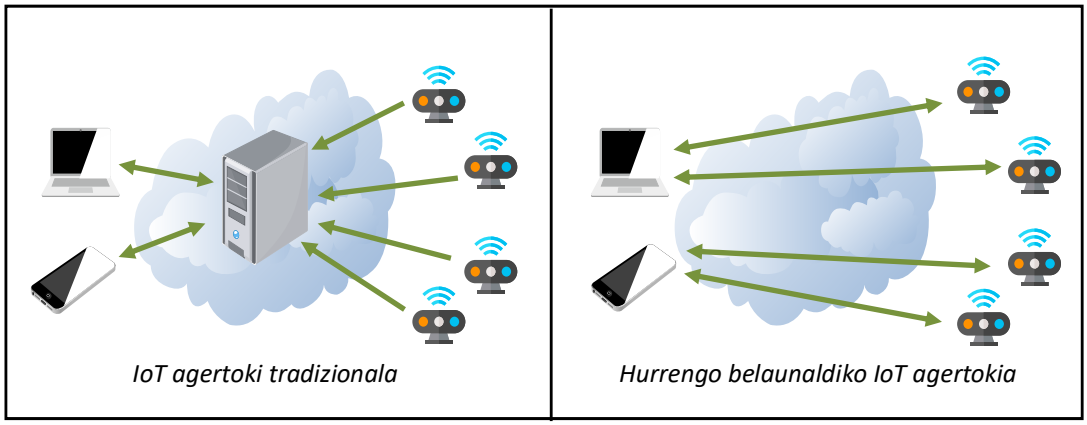

*Irudia 2: SmartCity baten aplikazio arloak*

Beraz, argi dago IoT-k jadanik gizartera egin dituen eta etorkizunerako espero diren ekarpenak oso garrantzitsuak izango direla arlo desberdinetan. Hala ere, izan ditzakeen potentzial guztia erabili ahal izateko, guztiz beharrezkoa da aplikazio hauetan segurtasun mekanismo egokiak inplementatzea. Izan ere, transmititzen diren datuek sentikortasun handia izan ditzakete, beraz, segurtasun ezaugarri guztiak betetzen direla bermatu behar da, hala nola, konfidentzialtasuna, eskuragarritasuna, osotasuna, autentifikazioa eta autorizazioa. Segurtasun hau nola inplementatzen den ulertu ahal izateko, beharrezkoa da IoT egoera desberdin bi aztertzea.

Alde batetik, IoT egoera tradizionaletan sentsoreen lan bakarra datuak biltzea eta datu hauek zentralizatutako zerbitzari batera bidaltzea da. Kasu hauetan, bezeroek eskaerak zerbitzari nagusira egiten dituzte, sentsorearekin muturretik muturrerako komunikazioa ezarri gabe, 3. irudiko ezkerreko eskeman adierazten den legez. Beraz, eskaerak zerbitzariak jasotzen dituenez eta aldez aurretik ezaguna denez jasoko dituen eskaeren datuak (IP helbideak adibidez), segurtasuna inplementatzeko gaur egun Interneten erabiltzen diren segurtasun mekanismo berdinak erabiltzen dira. Adibidez, bezeroaren eta zerbitzariaren artean gako bat partekatzea izan daiteke erabiltzen den segurtasun mekanismo bat.

Beste alde batetik, hurrengo belaunaldiko IoT-k egoera konplexuagoak ahalbidetuko ditu, *smart* CDSak erabiliz. Sentsore adimentsu hauek zerbitzari txiki moduan lan egiteko ahalmena izango dute. Egoera hauetan, CDSek bezeroen eskaerak jaso eta prozesatu ahalko dituzte jarraian *end-to-end* (E2E) komunikazio bat ezarriz haien artean, inolako zerbitzari zentral baten beharrik gabe, 3. irudiko eskumako eskeman ikusten den moduan. Horrez gain, bezeroak sentsoreetako parametroak konfiguratzeko edo aldatzeko aukera ere izan dezake. Hori posible egiteko, oso garrantzitsua da segurtasun ezaugarrien artean sarbide kontrol sendo bat izatea, sentsorera atzipena baimena duen bezeroak bakarrik izan dezan. Hala ere, gaur egun Interneten erabiltzen diren segurtasun neurriak ez

dira aplikagarriak kasu honetarako, sentsoreek dituzten gaitasun-mugak direla eta. Adibidez, ezin daiteke partekatutako gakoaren mekanismoa erabili, ezin delako aldez aurretik jakin zeintzuk izango diren komunikazioaren bi muturrak.




*Irudia 3: IoT agertoki bien eskemak*

Beraz, egoera hori kontuan harturik eta IoT aplikazioeek segurtasun egokia izan dezaten, beharrezkoa da IoT-rentzat berariazko segurtasun mekanismoak garatzea. Mekanismo hauek, C0 eta C1 motako gailuetan (memoria aldetik existitzen diren gailurik mugatuenetan) aplikatzeko modukoak izateaz gain, bukaerako erabiltzaileentzat erabilterrazak ere izan behar dira.

Hori dela eta, lan honetan gai honi buruzko analisi bat garatuko da, eta planteatzen den arazoari soluzio bat emango zaio, IoT inguruneetako zibersegurtasun mekanismoak erabilgarriagoak bihurtzeko. Kasu honetan, Hidra segurtasun protokoloaren erabilgarritasuna hobetzeko egingo da garapena. Horretarako, alde batetik interfaze grafiko bat garatuko da, segurtasun-politikak era erraz batean sortu ahal izateko, eta interfaze hau baita segurtasunean ez-adituak direnak ere erabiltzeko modukoa izango da. Beste alde batetik, segurtasun politikak kodifikatzeko eta dekodifikatzeko modulu batzuk diseinatu eta garatuko dira. Kodifikatzaileak segurtasun politiken adierazpen bitarra lortu ahal izateko balioko du, eta dekodifikatzaileak kontrako prozesua egin beharko du, hau da, adierazpen bitarretik politikaren esanahia interpretatu.

# 2. Testuingurua

Atal honetan gaiaren testuingurua azalduko da, gaia kokatzeko eta hobeto ulertzeko. Horretarako, lehenengo eta behin IoT gailuen ezaugarriak azalduko dira. Ondoren, gailu hauen arteko komunikazioa nola gauzatzen den azalduko da, eta bukatzeko, gailu hauetan exekutatzen diren aplikazioetan segurtasunaren inplementazioa nolakoa den azalduko da.

## 2.1. Gaitasun mugatuak dituzten gailuak

Sarreran aipatu den bezala, IoT-ren oinarria gaitasun txikiko gailuak Internetera konektatzea da. Gailu horien artean, gaitasun mugatuko sentsoreak, CDSak, erabiltzen dira. Sentsore hauek normalean RAM eta FLASH memoria txikiak izaten dituzte, eta haien prozesadoreen gaitasuna ere ez da handia izaten. Horrez gain, bateria moduan pilak erabiltzen dituzte normalean. Hori dela eta, gailu hauen baliabideak nahiko mugatuak direnez, hauek erabiltzen direnean oso garrantzitsua da baliabideak ahalik eta era eraginkorrenean erabiltzea, baliabideak xahutu gabe.

Esan bezala, gailu hauen mugak baliabide desberdinetan egon daitezke: memoria gaitasunean, bateria gaitasunean, prozesaketa gaitasunean… Hori kontuan izanik, IETFk memoriaren gaitasunaren araberako sailkapen bat egin du, CDSak hiru klase desberdinetan sailkatuz [3]. Klase hauek C0, C1 eta C2 dira, klase batetik bestera memoria baliabideen mugak handituz.

C0 klaseko gailuak mugatuenak dira, izan ere, haien gehienezko RAM eta FLASH memoriak nahiko txikiak dira (10 KB baino askoz gutxiago RAMerako eta 100 KB baino askoz gutxiago FLASHerako). Hortik abiatuz, C1 klaseko sentsoreen memoria gaitasunak handitu egiten dira C0-koekin konparatuz (gutxi gora behera 10 KB RAMerako eta 100 KB FLASHerako). Era berean, berdina gertatzen da C2 klaseko memoria gaitasunekin, C1 klasekoekin konparatuz memoria gaitasunak handiagoak dira (gutxi gora behera 50 KB RAMerako eta 250 KB FLASHerako).

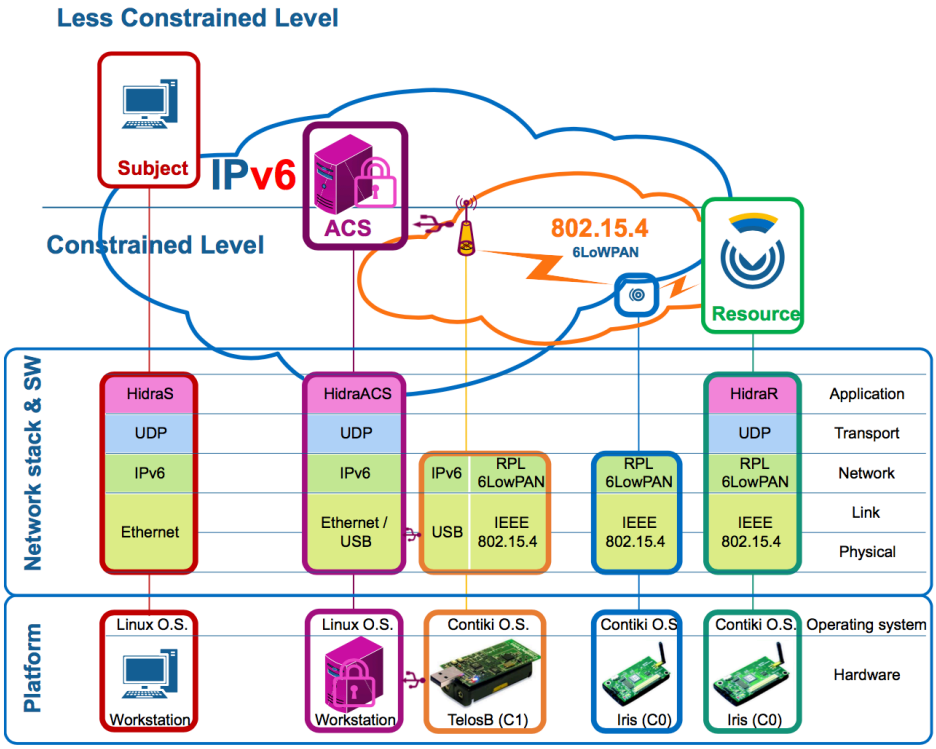
1. taulan klase bakoitzari dagozkien RAM eta FLASH memoria gaitasunak adierazten dira.

| | RAM (Datu-memoria) | FLASH (kode-memoria) |
|---|---|---|
| **C0** | << 10 KB | << 100 KB |
| **C1** | ≈ 10 KB | ≈ 100 KB |
| **C2** | ≈ 50 KB | ≈ 250 KB |

*Taula 1: CDSen memoriaren araberako sailkapena*

## 2.2. Komunikazioa

IoT aplikazioetan aurreko atalean deskribatutako sentsoreak erabili ahal izateko, beharrezkoa da erabiltzaileen eta sentsoreen arteko komunikazioa nolakoa den ulertzea. Horretarako, 4. Irudian komunikazioaren eskema eta erabiltzen diren protokolo pila adierazten dira.




*Irudia 4: komunikazio eskema eta protokolo pila*

Alde batetik, lotura mailan erabiltzen den protokoloa IEEE 802.15.4 da, I. eranskinean era zehatzean azaltzen dena. Izan ere, protokolo honek potentzia baxuko PANetan (*Personal Area Network*) haririk gabeko lotura espezifikatzen du, hori dela eta, sentsoreen sareetan erabiltzen den protokoloa da. Potentzia baxuko sareetarako denez, beharrizan berezi batzuk behar ditu, IoT aplikazioek betetzen dituztenak: bateria kontsumo txikia izatea, kostu baxuko nodo askok multisalto moduan lan egitea, banda-zabalera txikia erabiltzea, transmititutako potentzia txikia izatea, gailuetako memoria gaitasuna txikia izatea… Beraz, espezifikazio hauen ondorioz, IEEE 802.15.4 protokoloa egokiena da IoT aplikazioetako komunikazioan erabiltzeko.

Beste alde batetik, aipatu beharra dago IoT aplikazioetarako gailu kopuru oso handia konektatu behar dela sarera, sarreran azaldu den bezala. Gainera, gailu hauek zuzenean gaur egungo Internetera konektatu ahal izateko sare mailako protokoloa IP izan behar denez, beharrezkoa da konektatutako gailu bakoitzari IP helbide bat esleitzea. Hala ere, ezinezkoa da IPv4 (*Internet Protocol version 4*) protokoloko helbideak erabiltzea, ez baitaude gailu guztientzat helbide libre nahikoak.

Beraz, aplikazio hauetarako erabiltzen den sare mailako protokoloa IPv6 (*Internet Protocol version 6*) da. Protokolo hau IPv4-ren jarraipena da, eta horrekiko hainbat aldaketa eskaintzen ditu. Aldaketarik esanguratsuena helbideen hedapenean dago. Izan ere, IP helbideak 32 bitetik 128 bit izatera pasatzen dira. Honi esker, posible da sarean dauden nodo kopurua era esanguratsuan handitzea. Horrez gain, IPv6 protokoloak duen beste ezaugarri garrantzitsu bat, eta gailuak era masiboan desplegatzeko oso erabilgarria dena, IPv6 helbideak autosortzeko (MAC helbidea abiapuntutzat hartuta) eta autokonfiguratzeko gaitasuna duela da.

Hala ere, IPv6 eta IEEE 802.15.4 protokoloak ez dira zuzenean bateragarriak. Adibidez, bateragarritasun falta hau datagramaren tamainarekin ematen da. Izan ere, IEEE 802.15.4 protokoloan bidaltzen diren paketeak nahiko txikiak izaten dira, eta ez dira IPv6-ren MTUra *(Maximum Transmission Unit*) heltzen.

Horrez gain, IEEE 802.15.4-ko *throughput*-a eta potentzia oso txikiak dira, loturan interferentziak eta akatsak egon daitezke, beraz, sare mailako protokoloa egoera horretara egokitu behar da. Gainera, IEEE 802.15.4 protokoloaren beharrizanak bete ahal izateko, beharrezkoa da IPv6 protokoloko eta bere gaineko geruzetako goiburuak ahalik eta gehien konprimatzea, IPv6 protokoloak *overhead* handiak gehitzen dituelako datagrametara, eta hau ez da CDS aplikazioetarako egokia.

Beraz, geruza bien arteko bateragarritasun ezari aurre egiteko, IEEE 802.15.4 eta IPv6ren artean egokitzapen geruza bat behar da, bien arteko bateragarritasuna lortu ahal izateko. Egokitzapen geruza hau 6LoWPAN (IPv6 *over* LoWPAN) da, eta hainbat funtzio betetzen ditu: goiburuen konpresioa, fragmentazioa…. Gainera, IPv6 goiburuetako eremuek asko aldatzen ez diren balioak daramatzate. Beraz, hori da 6LoWPAN-en oinarria: lehenetsitako balioak zehazteen ditu eta aldaketarik ez badago, eremu horiek ez dira bidaltzen eta lehentsitako balioa erabiltzen da.

6LoWPAN-i esker, komunikazioan parte hartzen duten nodoen energia kontsumoa eta sentsoreen memoria kontsumoa era esanguratsuan murrizten dira. Gainera, lotura mailako eta sare mailako protokoloen arteko bateragarritasuna eskaintzeaz gain, 6LoWPAN protokoloak garraio maila ere inplementatzen du, horretarako normalean UDP (*User Datagram Protocol*) protokoloa erabiliz. 6LoWPAN-en funtzionamendu zehatza III. eranskinean azaltzen da.

## 2.3. Segurtasuna

IoT aplikazioek era egokian funtziona dezaten, eta batez ere hurrengo belaunaldiko IoT aplikazioak martxan jarri ahal izateko, guztiz beharrezkoa da segurtasun sendo eta eraginkorra izatea. Aurretik esan bezala, CDSek gaitasun mugatuak dituzte, bai memoria aldetik, bai bateria aldetik... Beraz, oso garrantzitsua da segurtasun mekanismoak bateragarriak izatea muga hauekin, eta baliabideak ahalik eta era eraginkorrenean erabiltzea. Hala ere, gaur egun Interneten erabiltzen diren segurtasun mekanismoak ez dira aplikagarriak IoT agertokietan, gailuek dituzten mugak direla eta. Hori dela eta, zaila da segurtasun mekanismo egokiak eta bideragarriak inplementatzea aplikazio hauetan, eta horren ondorioz, IoT aplikazioen erabateko hedapena ez da oraindik lortu.

Gainera, orain arte soluzio desberdinak proposatu izan dira segurtasunaren arloan, batez ere lotura mailan (edo LoWPAN sarearen barruan) autentifikazioa eta datuen konfidentzialtasuna bermatzeko. Hala ere, autorizazioari ez zaio beharrezko garrantzia eman, eta sarbide kontrola ez da asko eboluzionatu izan, nahiz eta garrantzitsua izan. Izan ere, IoT aplikazioen izaera dinamikoaren ondorioz, sarbide kontrol soluzio sendo, espresibo eta erabilgarriak behar dira.

Alde batetik, esan bezala, Interneteko segurtasun soluzio tradizionalak ez dira IoT aplikazioekin bateragarriak, aurretik aipatutako gailuen mugak direla eta. Soluzio tradizionalak IoT-ra egokitzeko ahaleginak egin dira, adibidez, *Authorization Framework for the IoT* [4] soluzioak XACML [5] (*Extensible Acces Control Markup Language*) soluzioaren egokitzapena egiten du IoT aplikazioetarako. XACML-k sarbide kontrolerako politikak definitzen ditu XML-rekin, baina sortzen diren politika hauek astunegiak dira gailu mugatuek prozesatu ahal izateko. Beraz, ez da soluzio aplikagarria IoT ingurunetarako. Beste aukera bat IoT-ra egokitutako Ucon (*Usage based access control*) [6] izango litzateke. Hala ere, soluzio honek arkitektura zentralizatua jarraitzen du, CDSetako baldintza lokalak kontuan hartu gabe. Ondorioz, behar diren dinamikotasuna eta espresibitatea ez da lortzen, eta ezin da alternatiba hau oso gaitasun mugatuak dituzten gailuetan, C0 gailuetan, erabili.

Arkitektura banatua erabiltzen duten beste alternatiba batzuk ere existitzen dira, hala nola CapBAC for IoT [7]. Soluzio honek gaitasunetan oinarritutako sarbide kontrola eskaintzen du, Javan oinarrituz. Hala ere, Javak behar duen *framework*a nahiko astuna da, beraz, ez da egokia C0 eta C1 gailuetarako, haien gaitasun-mugak oso handiak direlako. Beste alternatiba bat, DCapBAC [8] da, baina soluzio honetan baimenak gaitasunetan doaz sartuta, eta gaitasun hauek ezin dira gaurkotuak izan, beraz, soluzioaren malgutasuna eta dinamismoa mugatuak dira.

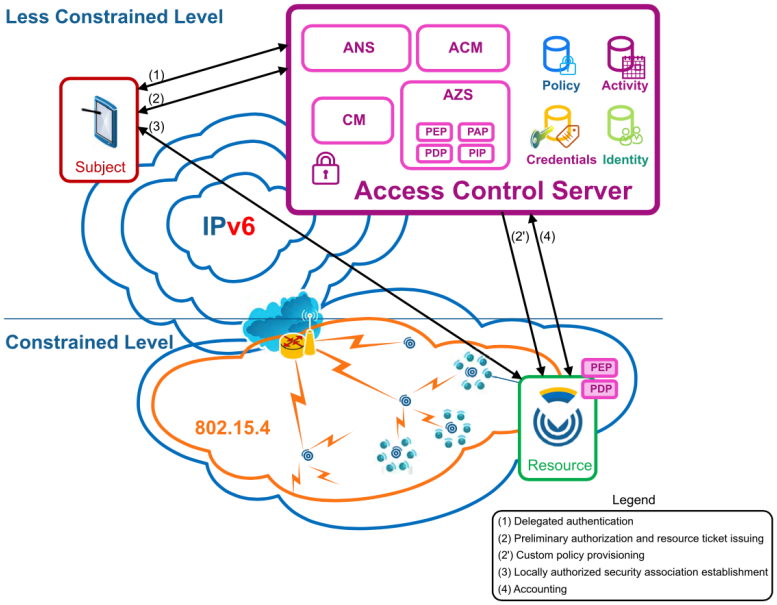
Existitzen den beste soluzio bat *Delegated CoAP Authentication and authorization Framework* (DCAF) [9] da. Soluzio honetan autorizazio politikak erabiltzen dira, baldintza lokalak kontuan hartuz, eta politika hauek CBOR [10] (*Concise Binary Object Representation)* serializazioa erabiliz konprimatzen dira. Hala ere, soluzio hau inplementatzeko DTLS (*Datagram Transport Layer Security*) kanalak ezarri behar dira, eta honek *overhead* handia izatea suposatzen du. Gainera, CBOR-ek ez du behar bezalako konpresio maila altua eskaintzen, konpresore generiko bat delako.

Aztertutako aukeren mugei aurre egiteko, batez ere oso gaitasun txikiak dituzten gailuetan erabilgarria izatearen aldetik, Ladon [11] protokoloa sortu zen. Protokolo honi esker, muturretik muturrerako autentifikazioa eta autorizazioa inplementatu daiteke baita C0 gailuetan ere.

Hala ere, Ladon protokoloak erabiltzen dituen politikak sarbide kontrolerako nahiko estatikoak dira, gainera, ez du *accounting* funtziorik egiten. Beraz, Ladonek duen mugarik handiena sarbide kontrolari espresibitaterik ez diola ematen da.

Hori dela eta, arazo horri aurre egiteko, Hidra [12] protokoloa sortu zen, IV. eranskinean zehatzago azaltzen dena. Protokolo honek bezeroaren eta bukaerako gailuaren arteko autentifikazio eta autorizazio espresiboa gauzatzen du, CDS oso mugatuetarako ere aplikagarria dena. Sarbide kontrol prozedurari adierazkortasuna eman ahal izateko, protokolo honek arkitektura zentralizatua eta banatua konbinatzen ditu, autorizazioa pauso bitan gauzatuz. Lehenengo eta behin, zerbitzari zentral batek autentifikazioa eta hasierako autorizazio prozedurak gauzatzen ditu, baimenik gabeko

eskaera gehienak deuseztatuz. Ondoren, testuinguruan eta baldintza lokaletan oinarritutako sarbide-erabakia hartu ahal izateko, CDSak era lokalean bigarren autorizazio prozedura bat gauzatzen du. Era horretan, sarbide-kontrol espresiboa lortzen da.




*Irudia 5: Hidraren funtzionamendu-eskema*

Beraz, CDSak berak baldintza lokaletan oinarritutako sarbide erabakia hartu ahal izateko, kasu bakoitzean segurtasun-politika oso espresibo bat aplikatu behar du. Hala ere, deskribatutako espresibitate maila altu hori lortzeko, gaur egun Interneten lortzen denaren baliokidea, protokoloaren konplexutasuna igotzen da. Izan ere, CDSak aplikatu behar dituen politiken egitura nahiko konplexua da, eta erabiltzen den kodifikazio bitarrak ere konplexutasuna handitzen dio, eta horren ondorioz, segurtasunean adituak ez diren pertsonentzat zailegia izango litzateke protokolo honekin lan egitea. Horregatik, Hidra protokoloak duen arazoetako bat erabilgarritasun falta da. Nahiz eta protokolo oso eraginkorra izan CDS oso mugatuetan erabiltzeko, ez da guztiz erabilgarria pertsona ez-adituentzat.

Horrez gain, aipatutako segurtasun politikak sentsoreetan aplikatu ahal izateko, politika horiek era eraginkorrean garraiatu behar dira zerbitzaritik sentsorera. Horretarako, kodifikazio bitar berezi bat proposatu da, baina beharrezkoa da horretarako kodifikatzaile eta dekodifikatzaile bat garatzea sentsoreek politika horiek ulertu eta aplikatu ahal izateko.

# 3. Lanaren helburuak eta irismena

Lan honen helburu nagusia Hidra protokoloaren erabilgarritasuna hobetzea da. Horretarako, hainbat funtzionaltasun diseinatu eta garatu behar dira. Alde batetik, erabiltzaile-interfaze grafiko bat sortu behar da, bertan segurtasun politika desberdinak definitu ahal izateko. Interfaze grafiko hau pertsona ez adituek erabili ahal izango dute, beraz, erabiltzeko erraza izan beharko da.

Beste alde batetik, sortutako segurtasun politikak sentsoreetara bidali ahal izateko kodifikazio bitar egokian egon behar dira. Beraz, adierazpen bitar hori sortzeko kodifikatzaile eta dekodifikatzaile modulu batzuk sortu behar dira.

Horrez gain, bigarren mailako helburu batzuk ere definitu dira lan honetan, hurrengo zerrendan deskribatzen direnak:

1. **Hidra protokoloa, politikak definitzeko lengoaia eta kodifikazio bitarra aztertu.**
   Lan honen diseinu eta garapenarekin hasi aurretik, beharrezkoa da lanaren oinarria diren hainbat kontzeptu ulertzea. Hasteko, Hidra protokoloaren funtzionamendua, transmititzen diren mezuen funtzioak eta edukia eta protokoloaren inplementazioa ulertu behar dira. Izan ere, lan honetan garatu behar diren atalak Hidra protokoloarekin batera funtziona dezaten garatu behar dira.
   Ondoren, segurtasun politikak definitu ahal izateko, erabiliko den politika lengoaia ikasi behar da, bere egiturak, eremuak eta haien arteko erlazio guztiak barneratuz. Bukatzeko, definitutako politikak sentsorera bidali ahal izateko aldez aurretik kodifikatu behar dira, dagokion kodifikazio bitarra jarraituz. Gainera, sentsoreak jasotzen duen politika kodifikatua interpretatu ahal izateko, dekodifikazio prozesu bat egin behar du. Beraz, beharrezkoa da aukeratutako kodifikazio bitarraren funtzionamendua ere ulertzea.

2. **Interfaze grafikoaren diseinua eta garapena.**
   Behin Hidra, politika lengoaia eta kodifikazio bitarra aztertuta daudenean, Hidra protokoloa erabilgarria egiteko falta diren funtzionaltasunen diseinuarekin hasi behar da. Horretarako, segurtasun politika berriak definitzeko erabiltzaile interfaze grafiko bat diseinatu eta garatu behar da. Interfaze honetan era askotariko politikak definitu ahalko dituzte baimen egokia duten erabiltzaileek. Interfaze grafikoaren diseinua egiterakoan, guztiz beharrezkoa da politika lengoaiaren egitura jarraitzea, eta sortutako politikak formatu egokian biltegiratzea.

3. **Kodifikatzaile eta dekodifikatzailearen diseinua eta garapena.**
   Interfaze grafikoa sortu ondoren, hurrengo helburua kodifikatzaile eta dekodifikatzailearen diseinua eta garapena egitea da. Esan bezala, politikak sentsorera bidali aurretik beharrezkoa da adierazpen bitar egokira kodifikatzea. Era berean, sentsoreak jasotako politikaren adierazpen bitarra dekodifikatu behar du, politikaren edukia era egokian interpreta dezan eta sarbide erabaki bat hatu ahal izateko. Bai kodifikatzailea eta bai dekodifikatzailea diseinatzeko eta garatzeko, bietan aukeratutako kodifikazio bitar berdinaren arauak jarraitu behar dira.

4. **Politiken biltegiratzea**
   Interfaze grafikoan sortzen diren segurtasun politika guztiak era egokian biltegiratu behar dira. Politika bera biltegiratzeaz gain, politikarekin erlazionatutako informazioa ere biltegiratu behar da, adibidez, politika sortu duen erabiltzailearen informazioa, sorrera dataren informazioa… Gainera, biltegiratzen den informazioa berreskuratzea posible izan behar da, datuak erabili ahal izateko geroago analisi desberdinak egiteko.

5. **Balidazio funtzionala gauzatzeko maketaren diseinua eta inplementazioa**
   Diseinatu eta garatuko diren modulu guztiak balidatu aurretik, beharrezkoa da balidazio-agertokia diseinatzea eta inplementatzea. Kasu honetan, balidazio funtzionala egiteko maketa erreal bat diseinatuko da, benetako sentsoreekin WSN sare bat sortuz. Ondoren, maketa hori inplementatu beharko da, sare osoa martxan jarriz.

6. **Interfaze grafikoaren, kodifikatzailearen eta dekodifikatzailearen balidazio funtzionala.**
   Bukatzeko, azken helburua interfaze grafikoaren eta kodifikatzaile/dekodifikatzaile moduluen balidazio funtzionala egitea da. Horretarako, alde batetik interfaze grafikoan luzera eta eduki desberdineko politikak sortu beharko dira haien sorrera eta biltegiratzea ondo gauzatzen dela egiaztatuz.
   Ondoren, sortutako politika horiekin proba batzuk egin beharko dira kodifikatzaile eta dekodifikatzailearekin. Kodifikatzerakoan adierazpen bitar egokia lortzen dela bermatu behar da, eta era berean sentsoreak dekodifikatu ondoren jatorrizko politikaren informazio egokia berreskuratzen duela ere bermatu behar da.

# 4. Onurak

## 4.1. Onura teknikoak

Proiektu honek alde teknikoan izan ditzakeen onurak oso nabarmenak dira. Izan ere, IoT aplikazioak gero eta gehiago ari dira erabiltzen, eta WSN sareak era masiboan ez hedatzearen arrazoietako bat segurtasun mekanismo sendo eta erabilgarrien falta da. Ondorioz, lan honetan Hidra segurtasun protokoloari falta zaizkion funtzionaltasunak diseinatuko eta garatuko direnez, era horretan falta den erabilgarritasuna lortuz, posible izan daiteke WSN sareak hedatzea eta erabiltzea. Izan ere, komunikazio segurua bermatuko da, batez ere sarbide-kontrolaren aldetik, eta segurtasun mekanismo horiek erabilgarriak izango dira segurtasunean ez-adituak diren pertsonentzat. Hori dela eta, proiektu honen emaitza aurrerapauso bat izan daiteke IoT aplikazioak eta WSN sareak hedatzeko eta erabiltzeko orduan.

Horrez gain, segurtasun mekanismo erabilgarri honi esker zerbitzu berriak inplementatu ahalko dira. Adibidez, medikuntza arloan erabiltzen diren sentsoreak, gaixotasun batekin erlazionatutako parametro batzuk neurtzeko balio dutenak, medikuek konfiguratu ahal izango dituzte zuzenean, momentu bakoitzean pazientearentzat onena dena aukeratuz, eta segurtasun arloko inolako adituren beharrik gabe.

Beste alde batetik, nahiz eta lan honetan garatuko diren mekanismoak Hidra protokoloarentzat diseinatuta egon, beste mekanismo batzuetara hedatzea posiblea da. Adibidez, politikak sortzeko interfaze-grafikoaren erabilera segurtasun-politiken erabileran oinarritzen den edozein mekanismorentzat ere hedatu daiteke.

## 4.2. Onura ekonomikoak

Alde ekonomikoari dagokionez ere, lan honek hainbat onura ekar ditzake. Alde batetik, WSN sareen inplementazioan gehikuntza nabarmen bat egon daiteke, eta sare hauek haririk gabeko sareak direnez, instalazioaren prezioan aurreztea posible izango da, ez delako instalazio kableatu bat sortzeko materialik behar. Horrez gain, aipatu beharra dago sare hauetan inplementatzen diren CDSen energia-kontsumoa oso baxua dela, beraz, bateria moduan erabiltzen dituzten pilen autonomia nahiko handia izaten da, ete honek ere ekonomikoki onura nabarmena suposatzen du.

Gainera, lan honetan segurtasun politikak definitzeko sortuko den interfaze grafikoak ere onura ekonomikoa suposatzen du. Izan ere, mekanismo honi esker posible da edozein segurtasun-politika inplementatzea CDSetan momentu bakoitzen behar denaren arabera, pertsona bat sentsorea programatzen ibili beharrean konfigurazioa aldatu behar den bakoitzean.

Beste alde batetik, gaur egun IoT inguruneetan babestu gabe dauden sistema asko existitzen dira, babesteko oso zailak direlako. Proiektu honetan landuko diren segurtasun mekanismoen erabilgarritasuna hobetuz, mekanismo hauek babestu gabeko sistema horiek babesteko erabili ahal izango dira. Horri esker, sistema horiek jasan ditzaketen segurtasun erasoak eta horren ondoriozko galera ekonomikoak ekidingo dira.

## 4.3. Onura sozialak

Bukatzeko, proiektu honek alderdi sozialean ere onura nabarmenak ditu. Alde batetik, lan honen oinarrietako bat segurtasun mekanismo konplexuak edozein pertsonarentzat, segurtasunean ez adituak direnentzat ere, erabilgarri egitea da. Honen ondorioz, teknologia aurreratuak, kasu honetan IoT inguruneetako segurtasun mekanismo aurreratuak, gizartera hurbiltzea lortzen da.

Horrez gain, lan honetan garatuko diren mekanismoei esker hainbat zerbitzu posible egingo dira, eta zerbitzu hauek pertsonentzat onuragarriak izango dira. Adibidez, osasunarekin lotuta, gaixotasun kardiobaskularrak detektatzeko zerbitzu bat inplementatu daiteke, gaixotasun horiek izateko arriskua duten pertsonei hainbat sentsore jarriz bihotzaren maiztasuna, gorputzaren tenperatura eta arterietako presioa neurtzeko, besteak beste. Ondoren, medikuak sentsore horietara sarbidea izan dezake, horien parametroak konfiguratzeko behar denean pertsonaren arabera. Ondorioz, mota honetako zerbitzuak desplegatuz gero, pertsonen segurtasuna bermatu ahal izango da.

# 5. Baldintzen deskribapena

Lan honetan diseinatuko eta garatuko den sistema oso baldintzatuta dago inplementatuko duen segurtasun mekanismoarengandik. Izan ere, garatuko diren atal guztiak jadanik sortuta dagoen segurtasun mekanismo horrekin integratu behar dira, eta honen ondorioz baldintza batzuk ezartzen dira. Beraz, beharrezkoa da lan honek dituen baldintzak deskribatzea.

Alde batetik, garatuko den sistema osoa Hidra protokoloarentzat egin behar da. Hau da, erabiliko den sarbide-kontrol mekanismoa Hidra izango denez, lanean zehar garatuko den modulu bakoitza protokolo honen funtzionamenduarekin bateragarria izan behar da. Ondoren, modulu horiek guztiak protokoloarekin integratu beharko dira, batera lan egin dezaten. Ondorioz, Hidra protokoloa C programazio lengoaian garatuta dagoenez, dekodifikatzailea ere C-n programatu beharko da, Hidrarekin batera exekutatuko delako. Gainera, kodifikazailea ere lengoaia berean garatu behar da, kodifikatzailearen eta dekodifikatzailearen arteko bateragarritasuna bermatzeko.

Beste alde batetik, garatuko den sistemak bukaerako gailuetako baliabideengan izango duen inpaktua ere kontuan izan behar da. Testuinguruan aipatu den legez, IoT-ren oinarria gaitasun txikiko gailuek, kasu honetan CDSek, Internetera konektatzeko gaitasuna dutela da. Gailu hauek baliabide mugatuak dituzte, memoria aldetik, bateria aldetik, prozesatzeko gaitasunaren aldetik… Gainera, muga horien artean memoriaren araberako CDSen sailkapen bat egin da, C0tik C2ra, C0 gailuak mugatuenak izanik. Beraz, lan honetan garatuko den sistema IoT-rako aplikagarria izan dadin, beharrezkoa da gailurik mugatuenetan ere inplementagarria izatea. Hau da, sistema osoak C0 gailuetan funtzionatzeko gaitasuna izan beharko du. Hori kontuan harturik, eta 1. taulako informazioa jarraituz, sortutako sistema osoak 10 KB baino askoz gutxiago izan beharko ditu RAM memoriarako, eta 100 KB baino askoz gutxiago FLASH memoriarako.

Hala ere, CDSen memoria gaitasuna ez da sistema baldintzatzen duen bakarra, beste gaitasunek ere muga batzuk ezartzen baitituzte. Alde batetik, sentsoreetan exekutatuko den programa osoaren kontsumo energetikoa baxua dela bermatu behar da, sentsoreen bateria-autonomia optimizatu ahal izateko. Ondorioz, garatzen den dekodifikatzaile moduluaren kontsumo energetikoa CDSetan inplementatzeko egokia dela egiaztatu behar da.

Beste alde batetik, CDSak prozesamendu osoa denbora tarte onargarri batean burutu behar du, erabiltzaileak erantzuna denbora laburrean jaso dezan. Hau da, erabiltzaile batek sarbide bat edota datu bat eskatzen duenean, erantzuna denbora errealean jaso beharko luke. Beraz, erabiltzaileak eskaera hasten duen momentutik erantzun bat jasotzen duen arte pasatzen den denbora ezin izango da segundo bat baino luzeagoa izan.

Horrez gain, Hidra protokoloak sarbide-kontrol espresiboa ahalbidetzeko segurtasun-politika oso espresiboak erabiltzen ditu. Beraz, politika horiek definitzeko beharrezkoa da Hidra protokoloaren definizioan deskribatutako politika lengoaia zehatza jarraitzea. Horretarako, bertan adierazten diren zehaztasun guztiak bete beharko dira politika guztietan.

Bukatzeko, politikak sentsoreetara adierazpen bitarrean bidaltzen dira, eta horretarako beharrezkoa da politikaren informazio guztia adierazpen bitar jakin batera kodifikatzea. Era berean, sentsoreak adierazpen bitarretik jatorrizko informazioa berreskuratu behar du sarbide-erabaki bat hartzeko. Horretarako, jasotzen duen politika kodifikatua dekodifikatu behar du. Beraz, lan honetan garatuko diren kodifikatzaileak eta dekodifikatzaileak kodifikazio bitar berdina jarraitu behar dute, eta kasu honetan, APBR *(Authorization Policy Binary Representation)* kodifikazio bitarra inplementatu behar da, Hidra protokoloaren definizioan deskribatzen dena.

# 6. Alternatiben analisia

Atal honetan lana garatzeko behar diren baliabideen aukeraketa egingo da, horretarako existitzen diren alternatiba desberdinen analisi bat eginez. Kasu honetan, 6 analisi desberdin garatuko dira: ebaluazioa burutzeko mekanismoa, CDS plataformak, CDSetako sistema eragileak, interfaze grafikoa garatzeko tekonlogia, politikak biltegiratzeko modua eta kodifikatzailea/dekodifikatzailea garatzeko programazio lengoaia.

## 6.1. Ebaluaketa burutzeko mekanismoa

Helburuetan azaldu den legez, lan honetan diseinatuko eta garatuko diren atalak Hidra protokoloarekin integratu behar dira, eta horien balidazio funtzionala egin behar da. Horretarako, sistema osoaren ebaluaketa burutzea beharrezkoa da, mekanismorik egokiena aukeratuz. Gainera, oso garrantzitsua da aukeratutako mekanismoarekin lortzen diren emaitzak errealak eta fidagarriak izatea. Ebaluazioa burutzeko mekanismoen artean, existitzen diren alternatiba desberdinak hurrengoak dira:

- **Simulazioa:** Alternatiba honetan sare oso bat simula daiteke SW batzuk erabiliz. Programa honek sarearen eta gailuen portaera simulatzen du, errealitateko egoera batean gerta daitekenera hurbilduz. Beraz, Hidra protokoloaren programa osoa simulatutako gailuetan kargatu behar da. Alternatiba honetan lortzen diren emaitzak agian ez dira guztiz errealak edo fidagarriak, baina simulazio bat gauzatzeko erraztasuna eta arintasuna kontuan hartzekoa da.
- **Eredu analitikoa:** Alternatiba honetan eredu analitiko bat erabiliz (Markov-en kateak adibidez) egin beharreko prozesamendu guztiak eta ematen diren mezu trukeak deskribatzen dituen eredu bat garatzen da. Kasu honetan lortzen diren emaitzak ez dira egoera errealeko emaitzak, eta agian fidagarritasun txikiagoa izan dezakete.
- **Maketa erreala:** Aukera honetan sentsore fisiko errealak erabili behar dira maketa bat sortzeko. Horretarako, hainbat sentsore behar dira haien artean WSN bat eratzeko. Maketa erreala sortzeak zailtasun handiagoa suposa dezakeen arren denbora eta konplexutasun aldetik, lortzen diren emaitzak guztiz fidagarriak dira, egoera erreal batean oinarritzen direlako.

Alternatiba horiek kontuan harturik, lan honetarako ebaluazio-mekanismorik egokiena aukeratzeko erabiliko diren irizpideak hurrengoak dira:

- **Emaitzen fidagarritasuna (%40):** Lortzen diren emaitzak egoera errealera hurbiltzen diren eta horren ondorioz emaitzek daukaten fidagarritasun maila. Egoera zenbat eta errealagoa izan, emaitzak orduan eta fidagarriagoak izango dira.
- **Baliabideen eskuragarritasuna (%35):** Aztertzen den alternatiba bakoitza inplementatzeko eskuragarri dagoen ala ez.
- **Inplementazio denbora (%25):** Alternatiba bakoitza inplementatzeko eta martxan jartzeko behar den denbora.

Ondorioz, alternatiba bakoitza ebaluatu behar da irizpide horiekin, irizpide bakoitzari nota bat emanik 0-10 artean. Ondoren, bukaerako puntuazio bat lortuko da puntuazio partzialak ponderatuz, eta puntuaziorik altuena aukerarik egokiena izango da.

| | Emaitzen fidagarritasuna (%40) | Baliabideen eskuragarritasuna (%35) | Inplementazio-denbora (%25) | Puntuazio totala |
|---|---|---|---|---|
| Simulazioa | 6 | 10 | 9 | 8,15 |
| Eredu analitikoa | 5 | 8 | 5 | 6,05 |
| Maketa erreala | 9 | 10 | 6 | **8,6** |

*Taula 2: ebaluazio-mekanismoen analisia*

Alternatiba bakoitza irizpideekiko ebaluatu ondoren, 2. taulan ikus daiteke aukerarik egokiena ebaluazioa gauzatzeko maketa erreala dela. Alde batetik, egoera erreal batean lan egiten duenez, lortzen diren emaitzak oso fidagarriak dira. Gainera, maketa sortzeko behar diren baliabideak (CDSak, sistema eragilea…) eskuragarri daudenez laborategian, aukera hau bideragarria da, nahiz eta inplementazio-denbora altuagoa izan.

## 6.2. CDSak: hardware plataformaren aukeraketa

Aurreko analisian erabaki den legez, lan honetan maketa erreala erabiliko da ebaluaziorako, benetako CDSak erabiliz. Beraz, diseinatuko eta garatuko diren atalak aukeratutako CDSetan inplementagarriak izan behar dira, hori baita lan honen oinarria. Beraz, erabili behar diren sentsoreak aukeratzea beharrezkoa da. Horretarako, merkatuan existitzen diren eta gure beharrizanetara gehien egokitzen diren plataformen analisi bat egin da, eta aztertuko diren 4 alternatibak hurrengoak dira:

- **TelosB:** Gailu hauek oso memoria gaitasun baxua dute orokorrean. Haien RAM memoriaren gaitasuna 10 KB-ekoa da eta FLASH memoriarena 48 KB-ekoa. Horrez gain, CDS honek 3 sentsore ditu integratuta, argia, tenperatura eta hezetasuna neurtzeko.
  IoT inguruneetan erabiltzen diren sistema eragileei dagokionez, mota honetako plataformak TinyOS, Contiki eta Riot sistema eragileekin bateragarriak dira.
- **Iris:** Gailu hauek memoria gaitasun handiagoa eskaintzen dute aurrekoekin konparatuz, izan ere, 8 KB-eko RAM gaitasuna eta 128 KB-eko FLASH gaitasuna dute. Hala ere, ez dute sentsorerik integratuta, eta sentsoreak erabiltzeko MTS310 plaka gehitu behar zaio.
  IoT inguruneetan erabiltzen diren sistema eragileei dagokionez, Iris plataformak TinyOS eta Contiki sistema eragileekin bateragarriak dira.
- **MicaZ:** Gailu hauen RAM gaitasuna 4 KB-ekoa da eta FLASH gaitasuna 128 KB-ekoa. Sentsoreei dagokienez, Iris gailuekin gertatzen den legez, ez du sentsorerik eskaintzen eta MTS310 plaka gehitu behar zaio sentsoreak erabiltzeko.
  IoT inguruneetan gehien erabiltzen diren sistema eragileei dagokionez, MicaZ plataformak TinyOs eta Contiki sistema eragileekin dira bateragarriak.

- **Zolertia Z1:** Gailu honen RAM gaitasuna 8 KB-ekoa da eta FLASH gaitasuna 92 KB-ekoa. Horrez gain, CDS honek tenperatura neurtzeko sentsore bat dauka integratuta.
  IoT inguruneetan gehien erabiltzen diren sistema eragileei dagokionez, plataforma hau TinyOS, Contiki eta Riot sistema eragileekin bateragarria da.

Alternatiba horiek kontuan harturik, lan honetarako plataformarik egokiena aukeratzeko ebaluatuko diren irizpideak eta haien garrantzia hurrengoak dira:

- **FLASH memoria (% 40):** FLASH memorian programaren kodea kargatzen da, ondorioz, beharrezkoa da gaitasun nahikoa izatea programa osoa biltegiratzeko. Hori dela era, nahiz eta CDSek memoria gaitasun mugatua izan, garrantzitsua da gaitasun hori behar baino baxuagoa ez izatea.
- **RAM memoria (% 30):** Plataformaren RAM memorian datuak biltegiratu behar dira. Horretarako FLASH memoriak behar duena baino gaitasun gutxiago behar da, baina hala ere, datu guztiak biltegiratzeko nahikoa izan beharko da.
- **Erabiltzeko erraztasuna (% 20):** CDS plataformak konfiguratzea eta bertan programak kargatzea erraza izatea bilatzen da. Izan ere, sistema osoa martxan jartzeko orduan proba desberdinak egingo dira, eta posible programak hainbat alditan kargatu behar izatea sortzen doazen kode-erroreak zuzentzea. Ondorioz, zenbat eta errazagoa izan, orduan eta denbora gutxiago galduko da lan hori egiteko.
- **Soportatzen dituzten sistema eragileak (%10):** CDS plataforma bakoitza hainbat sistema eragilerekin da bateragarria. Ondorioz, zenbat eta malguagoa izan zentzu honetan eta zenbat eta sistema eragile gehiago jasan, errazagoa izango da sistema eragilearen aukeraketa egitea.

Beraz, alternatiba bakoitza ebaluatu behar da irizpide horiekin, irizpide bakoitzari nota bat emanik 0-10 artean. Ondoren, bukaerako puntuazio bat lortuko da puntuazio partzialak ponderatuz, eta puntuaziorik altuena aukerarik egokiena izango da.

| | FLASH (% 40) | RAM (% 30) | Erabiltzeko erraztasuna (% 15) | Sistema eragileak (%10) | Puntuazio totala |
|---|---|---|---|---|---|
| TelosB | 5 | 8 | 7 | 9 | 6,35 |
| Iris | 9 | 7 | 6 | 8 | **7,4** |
| MicaZ | 9 | 3 | 5 | 8 | 6,05 |
| Zolertia Z1 | 7 | 7 | 3 | 9 | 6,25 |

*Taula 3: sentsoreen analisia*

Alternatiba bakoitza irizpideekin aztertu ondoren, 3. taulan aukerarik onena Iris sentsorea dela ikus daiteke. Izan ere, CDS plataforma honek memoria gaitasunik altuena eskaintzen du, eta hau oso onuragarria da Hidra protokoloaren kode osoa kargatu ahal izateko. Beraz, lan honetan CDS bezala erabiliko diren plataformak hauek izango dira.

Hala ere, beharrezkoa izango da *router* moduan lan egiten duen beste plataforma bat erabiltzea, honek 6LoWPAN interfazea eskaini behar duelako, eta horretarako TelosB motakoa erabiliko da, bigarren puntuaziorik altuena lortu duena baita. Plataforma honek memoria FLASH

gaitasun txikiena du, baina *border router* funtzioak betetzeko ez da gaitasun oso alturik behar. Beraz, TelosB plataformak eskaintzen dituen prestazioak nahikoak dira funtzio hori betetzeko.

Ondorioz, lan honetan 2 motako CDS plataformak erabiliko dira. Alde batetik, Iris plataformak WSN sarea eratzeko eta Hidraren kodea exekutatzeko. Beste alde batetik, TelosB *border router* funtzioak gauzatzeko.

## 6.3. CDSetako sistema eragilea

Lan honetan garatuko den sistema martxan jarri ahal izateko, beharrezkoa da sentsoreetan Hidra protokoloaren software-a kargatzea. Horretarako, sentsoreek sistema eragile bat behar dute, IoT aplikazioekin bateragarria den sistema eragile bat, hain zuzen ere. Arlo honetan hainbat alternatiba existitzen dira, baina analisi honetan 3 sistema eragile erabilienak aztertuko dira: tinyOS, Contiki eta Riot. Bakoitzaren espezifikazio orokorrak hurrengo tauletan adierazten dira:

- **TinyOS:** Sistema eragile honek CDSetan okupatzen duen memoria gaitasuna <1 KB-ekoa da RAMaren kasuan, eta <4 KB-ekoa ROMaren kasuan. Horrez gain, sistema eragile honekin sortzen diren programak nesC lengoaian egon behar dira idatzita. Urte askotan zehar sistema eragile oso erabilia izan da, baina gaur egun zaharkituta geratu da beste alternatiba berriago batzuen sorrera dela eta.
- **Contiki:** Sistema eragile honek CDSetan okupatzen duen memoria gaitasuna <2 KB-ekoa da RAMerako, eta <30 KB ROMerako. Ikus daitekeen legez, ROM askoz ere handiagoa behar du aurreko alternatibarekin konparatuz gero. Gainera, sistema eragile honekin sortzen diren programak C lengoaian idatzi behar dira. IoT inguruneetan asko erabiltzen den sistema eragilea da.
- **Riot:** Sistema eragile honek CDSetan okupatzen duen memoria gaitasuna 1,5 KB-ekoa da RAMerako, eta 5 KB-ekoa ROMerako. Ikusten den moduan, memoria kontsumo oso txikia du sistema eragile honek. Gainera, programazio lengoaia desberdin bi jasaten ditu: C eta C++. Gaur egun gero eta erabiliagoa ari da izaten sistema eragile hau.

Alternatiba horiek kontuan harturik, lan honetarako sistema eragilerik egokiena aukeratzeko ebaluatuko diren irizpideak eta haien garrantzia hurrengoak dira:

- **Iris eta TelosB plataformekin bateragarritasuna (%40):** CDS plataformaren aukeraketa egiteko analisian deskribatu den legez, proiektu honetan Iris eta TelosB plataformak erabiliko dira. Beraz, beharrezkoa da aukeratzen den sistema eragilea plataforma hauekin bateragarria izatea. Hori kontuan izanik, irizpide honekiko ebaluazio posibleak 0 edo 10 izango dira, 0 ez denean bateragarria, eta 10 bateragarria denean.
- **Memoria kontsumoa (% 30):** Sistema eragile bakoitzaren instalazioak CDSan okupatzen duen memoria gaitasuna da. Memoria kontsumoa ahalik eta baxuena izatea bilatzen da, leku libre gehiago edukitzeko programak kargatzeko.
- **Onartzen dituzten programazio lengoaiak (%20):** Sistema eragile bakoitzak onartzen dituen programazio lengoaien malgutasuna eta erraztasuna.

- **Erabilera maila (% 10):** Sistema eragile bakoitza zenbat erabiltzen den gaur egun IoT inguruneetarako. Zenbat eta gehiago erabili, errazagoa izango da sortzen diren akatsen soluzioak aurkitzea.

Beraz, alternatiba bakoitza ebaluatu behar da irizpide horiekin, irizpide bakoitzari nota bat emanik 0-10 artean. Ondoren, bukaerako puntuazio bat lortuko da puntuazio partzialak ponderatuz, eta puntuaziorik altuena aukerarik egokiena izango da.

| | Bateragarritasuna (% 40) | Memoria kontsumoa (% 30) | Programazio lengoaiak (%20) | Erabilera maila (% 10) | Puntuazio totala |
|---|---|---|---|---|---|
| TinyOs | 10 | 8 | 5 | 5 | 7,9 |
| Contiki | 10 | 6 | 7 | 9 | **8,1** |
| Riot | 0* | 9 | 8 | 7 | 5 |

*Taula 4: sistema eragileen analisia*

**Iris plataformarekin ez bateragarria*

Alternatiba bakoitza irizpideekin aztertu ondoren, 4. taulan ikus daiteke sistema eragilerik egokiena kasu honetan Contiki dela, CDS plataformekin bateragarria izateaz gain memoria kontsumo onargarria eta programazio lengoaia erraza duelako. Beraz, lan honetan sistema eragile hori erabiliko da sentsoreetan. Sistema eragilearen aukeraketak eta honek onartzen duen programazio lengoaiak kodifikatzailearen eta dekodifikatzailearen programazio lengoaiak baldintzatzen ditu, guztietan berdina izan behar delako. Ondorioz, kodifikatzaile eta dekodifikatzailearen programazio-lengoaia aukeratzeko ez da analisirik egingo, zuzenean C lengoaia erabiliko baita.

Hala ere, aipatu beharra dago beste sentsore batzuk erabiliko balira, Riot sistema eragilearekin bateragarriak direnak, Riot izango litzatekeela aukerarik onena, gaur egun gero eta gehiago erabiltzen baita.

## 6.4. Interfaze grafikoa garatzeko teknologia

Helburuetan azaldu den legez, Hidra protokoloa erabilgarria izateko proiektu honetan egingo den lehenengo gauza erabiltzaile-interfaze grafiko bat diseinatzea eta garatzea da. Interfaze honek segurtasun politika berriak sortzeko aukera emango du era erraz batean, ez adituak diren pertsonek ere erabili ahal izateko. Ondorioz, hori garatu ahal izateko beharrezkoa da erabiliko den teknologia eta programazio lengoaia aukeratzea. Existitzen diren alternatiba desberdinak hurrengoak dira:

- **JavaEE (*Java Enterprise Edition*):** Teknologia hau Javan oinarritzen da, eta web aplikazioak garatzeko oso erabilia da. Gainera, hainbat estandar inplementatzen ditu proiektu honetarako oso erabilgarriak izan daitezkeenak. Adibidez, JDBC (*Java Database Conectivity*) estandarrari esker posible da aplikazioa datu-basearekin konektatzea eta sinkronizatzea.
- **ASP.NET:** Teknologia hau ere web aplikazioak garatzeko erabiltzen da, eta MySQL datu-baseekin sinkronizazio arina eskaintzen du. Gainera, hainbat programazio-lengoaiekin bateragarria da: C#, J#... Hala ere, ez da kode irekiko teknologia, Microsoft-ena baita.

- **PHP:** Web aplikazioak sortzeko teknologia honek PHP *(Hypertext Preprocessor)* programazio lengoaia erabiltzen du. Teknologia honen abantaila nagusia MySQL datu-baseekin era erraz batean konektatzeko gaitasuna eskaintzen duela da. Hala ere, konfigurazioa oso zehatza izan behar da, eta programak oso ondo babestu behar dira segurtasun arazoak ekiditeko.

Alternatiba horiek kontuan harturik, aukerarik egokiena aukeratzeko irizpide desberdinak hartuko dira kontuan:

- **Programazio lengoaia (%40):** Alternatiba bakoitzerako menperatu behar den programazio lengoaia kopurua eta lengoaien zailtasun-maila aztertzen ditu irizpide honek.
- **Eskuragarritasuna (%35):** Alternatiba bakoitzaren eskuragarritasuna neurtzeko, kode irekikoak edo kode itxikoak diren aztertuko da. Izan ere, kode irekiko teknologiak egokiagoak dira proiektu honetarako, eskuragarriago baitaude.
- **Datu-baseekin sinkronizazioa (%25):** Garatuko den interfaze grafikoa datu-base batekin egon behar da konektatuta, beraz, beharrezkoa da teknologiak datu-baseekin sinkronizazio arina eskaintzea.

Beraz, alternatiba bakoitza irizpide horiekiko ebaluatu behar da, irizpide bakoitzari nota bat emanik 0-10 artean. Ondoren, bukaerako puntuazio bat lortuko da puntuazio partzialak ponderatuz, eta puntuaziorik altuena aukerarik egokiena izango da.

| | Programazio lengoaia (%40) | Eskuragarritasuna (%35) | Datu-baseekin sinkronizazioa (%25) | Puntuazio totala |
|---|---|---|---|---|
| JavaEE | 9 | 10 | 9 | **9,35** |
| ASP.NET | 5 | 0 | 9 | 4,25 |
| PHP | 7 | 10 | 9 | 8,55 |

*Taula 5: interfaze grafikorako programazio lengoaiaren analisia*

Ondorioz, 5. taulan ikus daitekeen legez, interfaze grafikoa garatzeko teknologiarik egokiena JavaEE da. Izan ere, teknologia hau erabiltzeko behar den programazio-lengoaia Java da, oso ezaguna eta erabiltzeko erraza dena. Gainera, MySQL datu-baseekin sinkronizazioa eskaintzen du, eta hori lan honetarako beharrezkoa da. Ondorioz, teknologia hori erabiliko da interfaze-grafikoaren garapena egiteko.

## 6.5. Politikak biltegiratzeko modua

Erabiltzaile-interfaze grafikoan segurtasun politikak definitzeaz gain, beharrezkoa da politika hauek biltegiratzea, aurrerago erabili ahal izateko. Alde batetik, biltegiratutako politikak kodifikatzaileak irakurri beharko ditu, bertako informazioa kodifikatu ahal izateko. Beste alde batetik, biltegiratutako informazio guztia eskuragarri egon beharko da aurrerago maila altuagoko analisiak egiteko eta ondorioak ateratzeko. Horretarako, hainbat modu egon daitezke:

- **Testu fitxategia:** Sortzen diren politikak testu fitxategi batean gorde daitezke, eta modu hau sinpleena eta arinena da, hori izanik bere abantailarik handiena. Hala ere, testu

fitxategi bat ez da datuak biltegiratzeko oso modu egokia geroago datu horiek erabili edo aztertu nahi badira.

- **Datu-basea:** Sortutako politikak datu base batean gordetzeko hainbat gauza izan behar dira kontuan. Datu-base erlazional bat erabiltzekotan, adibidez, beharrezkoa da taula desberdinak sortzea eta taulen arteko erlazio zuzena ezartzea. Datu basea erabiltzearen abantaila nagusia politikaren edukiaz gain politikarekin erlazionatutako informazio gehigarria ere biltegiratu daitekeela da. Hau oso erabilgarria izan daiteke politikak sortzeko baimenak dituzten erabiltzaileen informazioa biltegiratzeko, politika bakoitza noiz eta nork sortu duen adierazteko… Horri esker, datuekin analisi desberdinak egin daitezke, maila altuagoko ondorioak ateratzeko.
- **XML fitxategia:** Alternatiba honetan politika bakoitza XML formatuko fitxategi batean biltegiratuko litzateke. Formatu honetan biltegiratzearen abantailarik nagusiena politika bakoitzaren datuak ondo egituratuta eta ordenatuta biltegiratzen direla da, egitura bakoitzari etiketa bat dagokiolako. Gainera, politika lengoaiak kateatutako egiturak dituenez, XML fitxategietan posible da kateatze hori mantentzea biltegiratzeko orduan. Beraz, politika baten edukia irakurri nahi bada era egokian, alternatiba ona izan daiteke modu arinean eta errazean lor daitekelako. Hala ere, alternatiba honetan ez da posible politikarekin erlazionatutako informazio gehigarria era errazean biltegiratzea ezta informazioaren arteko erlazioak ezartzea. Era berean, biltegiratutako datuak analisi desberdinak egiteko erabili nahi badira, ez da modurik egokiena.

Alternatiba horiek kontuan harturik, aukerarik egokiena aukeratzeko irizpide desberdinak hartuko dira kontuan:

- **Politiken edukiaren egitura mantentzea (%40):** Sortzen diren politikek lengoaia espezifiko bat jarraitu behar dute, eta honek politika bakoitza egitura desberdinetan banatzen du, egiturak haien artean kateatuz. Beraz, garrantzitsua da biltegiratzen diren politikek egitura hori ez galtzea, beranduago kodifikatu ahal izateko ezinbestekoa baita egitura zuzena berreskuratzea. Ondorioz, aukeratutako alternatibak politiken egitura mantentzeko gaitasuna izan beharko luke.
- **Datuak berreskuratzeko eta erabiltzeko erraztasuna (%30):** Biltegiratzen diren politiken informazioa berreskuratzeko erraza izan behar da. Izan ere, momenturen batean politikei buruzko informazioaren analisi bat egin nahi bada, maila altuagoko ondorioak ateratzeko politiken sorrerari buruz, beharrezkoa da biltegiratuta dagoen informazio guztia era erraz batean berreskuratu ahal izatea.
- **Informazio gehigarria gehitzeko eta erlazionatzeko gaitasuna (%20):** Esan bezala, politikak biltegiratzen direnean ez da bakarrik haien edukia biltegiratu behar. Politikekin erlazionatutako informazio gehigarria ere biltegiratzea beharrezkoa da, eta informazio hori politika bakoitzarekin era egokian erlazionatu beharko litzateke. Adibidez, politikak sortzeko baimena duten erabiltzaileen artean, beharrezkoa izango litzateke politika bakoitza zein erabiltzailek sortu duen adieraztea. Beraz, aukeratutako alternatibak aukera hau eman beharko luke.

- **Biltegiratze prozesuaren erraztasuna (%10):** Interfaze grafikoan sortzen diren politikak biltegiratzeko prozesua oso astuna ez izatea komeni da, beraz, hori egiteko erraztasuna eta arintasuna ere kontuan hartu behar dira.

Beraz, alternatiba bakoitza irizpide horiekiko ebaluatu behar da, irizpide bakoitzari nota bat emanik 0-10 artean. Ondoren, bukaerako puntuazio bat lortuko da puntuazio partzialak ponderatuz, eta puntuaziorik altuena aukerarik egokiena izango da.

| | Politiken egitura (%40) | Datuak berreskuratzea eta erabiltzea (%30) | Informazio gehigarria gehitzea eta erlazionatzea (%20) | Erraztasuna (%10) | Puntuazio totala |
|---|---|---|---|---|---|
| Testu fitxategiak | 4 | 4 | 4 | 10 | 4,6 |
| Datu-basea | 8 | 10 | 10 | 8 | **9** |
| XML fitxategiak | 10 | 7 | 4 | 7 | 7,6 |

*Taula 6: politikak biltegiratzeko moduaren analisia*

Ondorioz, 6. taulan ikus daitekeen legez, aukerarik onena segurtasun politikak biltegiratzeko datu-basea da. Izan ere, alternatiba honek informazio gehigarria gehitzea eta politika bakoitzarekin erlazionatzea ahalbidetzen du. Adibidez, erabiltzaileen taula bat sortuz gero, oso erraz erlazionatu daiteke politika bakoitza zein erabiltzailek sortu duen. Horrez gain, biltegiratuta dauden datuak berreskuratzeko gaitasuna ere ematen du alternatiba honek. Horri esker, edozein momentutan existitzen diren datuei buruzko analisi bat egin daiteke, baita beharrezkoak diren ondorioak atera ere.

Hala ere, alternatiba hau ez da egokiena politikaren jatorrizko egitura mantentzeko, XML fitxategietan hau oso ondo lortzen baita. Hori dela eta, proiektu honetan alternatiba horiek biak erabiltzea erabaki da: datu-basea eta XML fitxategiak.

Alde batetik, biltegiratze iraunkor moduan datu-basea erabiliko da, bai politiken edukia eta bai erlazionatutako informazio gehigarria biltegiratzeko. Alternatiba-hau ondoren datuak berreskuratzeko eta behar diren analisiak egiteko erabiliko da batez ere. Gainera, interfazea erabiltzeko baimena duten erabiltzaileen informazioa ere datu-base berdinean biltegiratuko da.

Beste alde batetik, politika bakoitzaren edukia XML fitxategi batean ere gordeko da, politikaren egitura oso ondo mantentzen duelako. Fitxategi honen funtzioa kodifikatzailetik irakurtzea izango da, horrela, kodifikatzaileak politika bakoitzaren egitura zehatza berreskuratuko du zuzenean, eta bertako informazioa adierazpen bitarrera kodifikatu ahalko du.

# 7. Arriskuen analisia

Atal honetan proiektuan zehar sor daitezkeen arazoen analisi bat egingo da. Arazo hauek gertatzekotan, proiektuak atzerapena jasan dezake, edota proiektuaren garapen egokia arriskuan jar daiteke. Ondorioz, arriskuen analisi sakon bat egitea gomendagarria da, kasu bakoitzerako prebentzio-neurriak ezarriz edota arazoa gertatzekotan nola konponduko litzatekeen pentsatuz.

Arrisku bat era egokian baloratzeko, arriskuak gertatzeko duen probabilitatea eta gertatzekotan suposatuko lukeen inpaktua aztertu behar dira. Proiektu honetan arrisku desberdinen analisia egin da.

## 7.1. Arriskuak

### A1 - Atzerapenak

Arriskuetako bat plangintzan ezarritako eperen bat ez betetzea da. Ataza baten epeak ez badira betetzen, horren menpe dauden gainontzeko atazak ere beranduago hasiko dira. Honen ondorioz, proiektu osoak atzerapena jasan dezake, eta hori saihestu beharreko zerbait da.

- Gertatzeko probabilitatea: %50.
- Inpaktua: %50

Nahiko probablea da plangintza osoko atazen baten epeak ez betetzea, edozein atazetan arazoak sor daitezkeelako. Arazo honek izango lukeen inpaktua ere ertaina dela esan daiteke. Alde batetik, posiblea da ataza batean edukitako atzerapena beste ataza batekin berreskuratzea, uste baino arinago betetzen bada.

Arrisku hau ekiditeko plangintza oso ondo garatu behar da, gertatu daitezkeen ezusteak aurreikusiz eta ezuste horientzat denbora kontuan harturik, finkatutako epeetan tarte gehigarriak ezarriz badaezpada.

### A2 – CDSan memoria nahikoa ez izatea SW osoa sartzeko

Proiektu honetan 3 modulu garatzen dira: interfaze grafikoa, kodifikatzailea eta dekodifikatzailea. Lehenengo 2 moduluekin ez da memoria-arazorik egongo, zerbitzari zentralean exekutatzen baitira. Hala ere, dekodifikatzaileak memoria-arazoak izan ditzake, CDSan exekutatzen baita. Izan ere, CDSek memoria oso mugatua dute, beraz, dekodifikatzailearen kodea Hidra protokoloaren kodera gehitzerakoan ezin izango da CDSaren memoria-gaitasuna baino altuagoa izan. Ez bada kode osoa CDSan sartzen, ezin izango da sistema martxan jarri.

- Gertatzeko probabilitatea. %10
- Inpaktua: %90

Ikusten den moduan, ez da oso probablea arazo hau ematea. Izan ere, Hidra protokoloaren kodearen tamaina ikusita, tarte handia dago oraindik Iris plataformen memoria-gaitasun maximora heltzeko. Beraz, aurreikuspenen arabera, dekodifikatzailearen tamainak ez du muga hori gainditukо.

Hala ere, hori gertatzekotan egongo litzatekeen inpaktua oso handia da, sistema ezin izango delako martxan jarri. Ondorioz, hori gertatzekotan dekodifikatzailearen kodea arintzeko metodoak bilatu beharko lirateke, beharrezko tamainara egokitu ahal izateko.

### A3 - CDS plataformekin arazoak

Proiektu honen balidazioa egiten denean maketa erreal bat inplementatuko da, alternatiben analisian erabaki den legez. Beraz, maketa honetan CDS errealak erabiliko dira. Maketaren inplementazioa egiterakoan posible da CDSekin arazoak izatea: haien funtzionamendua ez izatea egokia, sareak ez detektatzea CDSak, zerbitzariarekin ondo ez komunikatzea, sentsoreen neurketak egokiak ez izatea…

- Gertatzeko probabilitatea: %50
- Inpaktua: %70

Ikus daitekeen legez, arrisku hau gertatzeko probabilitatea ertaina. Hala ere, izango lukeen inpaktua nahiko handia da, sistema osoaren funtzionamendua arriskuan jartzen delako eta ezin izango litzateke sistema martxan jarri. Hala ere, arrisku honetan eman daitezkeen arazoek normalean soluzio erraza izatea espero da. Arazoren bat gertatzekotan, CDSak apurtuta dauden konprobatu beharko da lehenengo eta behin. Apurtuta badaude, soluzio bakarra CDS berriak erostea izango da. Ez badaude apurtuta baina funtzionamendua ez bada egokia, plataformak berrabiarazi behar dira, barruan daukaten programak ezabatuz eta berriro zerotik hasiz.

## 7.2. Konparaketa

Hurrengo taulan 3 arriskuen arteko konparaketa ikusten da. Bertan, arrisku bakoitza taularen puntu batean kokatzen da bere, inpaktuaren eta probabilitatearen arabera.

| | | **Inpaktua** | | | | |
|---|---|---|---|---|---|---|
| | | %10 | %30 | %50 | %70 | %90 |
| **Probabilitatea** | %10 | | | | | **A2** |
| | %30 | | | | | |
| | %50 | | | **A1** | **A3** | |
| | %70 | | | | | |
| | %90 | | | | | |

*Taula 7: arriskuen analisiaren konparaketa*

7. taulan arrisku bakoitza bere gertatzeko probabilitatearen eta inpaktuaren larritasunaren arabera kokatu da. Era horretan, arriskurik garrantzitsuenak taularen atal gorrian kokatuko dira, eta garrantzirik txikiena dutenak atal berdean. Horri esker, proiektuan zehar atal gorrietan dauden arriskuei arreta handiagoa jarri beharko zaie, ekiditeko asmoarekin.

Kasu honetan 3 arriskuak atal laranjan kokatzen dira, hau da, garrantzi ertaina dute hirurek. Hala ere, A3 arriskuak beste biek baino garrantzi handiagoa du. Ondorioz, arreta handiagoa jarri beharko da A3 arriskua saihesteko edota gertatzekotan eraginak ahalik eta arinen zuzentzeko.

# 8. Diseinua

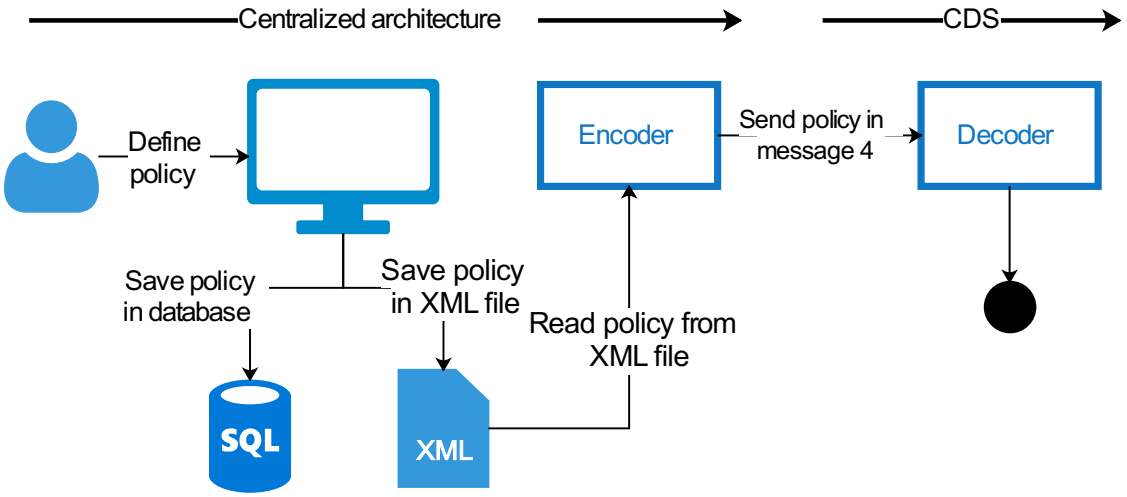
## 8.1. Erabilgarritasuna hobetzeko moduluen diseinua

Atal honetan Hidra protokoloa erabilgarria egiteko behar diren funtzionaltasunen diseinua eta garapena azalduko dira. 6. irudian garatuko diren modulu desberdinak eta haien arteko erlazioa deskribatzen dira.

Esan bezala, alde batetik erabiltzaile interfaze grafiko bat sortu behar da, bertan segurtasun adituak ez diren pertsonek ere sarbide kontrolerako politikak definitu ahal izateko. Modulu honi esker, Hidrari falta zaion erabilgarritasuna lortzen da. Ondoren, politika sortuta dagoenean, formatu desberdin bitan gorde behar da. Alde batetik, datu-base batean biltegiratu behar da politikaren eduki guztia eta sortu duen pertsonaren informazioa. Beste alde batetik, XML fitxategi batean ere gordetzen da politika. Era horretan, beranduago sentsorera politika bat bidali behar denean XML fitxategi horretatik irakurriko da politika.

Behin politika zerbitzari zentraletik irakurri denean, eta sentsorera bidali aurretik, adierazpen bitar egokira kodifikatu behar da. Horretarako, kodifikatzaile modulu bat sortu behar da, eta hau izango da politika era bitarrera kodifikatzearen arduraduna. Ondoren, politika sentsorera bidali ahalko da dagokion mezuan, bertan aplikatu dadin sarbide erabaki bat hartu ahal izateko.

Bukatzeko, sentsoreak politika kodifikatua jasotzen duenean, politikaren edukia interpretatu behar du, eta horretarako dekodifikatzaile modulu bat behar da, kodifikatzailearen kontrako prozesua egiten duena. Politika dekodifikatu denean, sentsorea gai izango da bere edukia ulertzeko eta dagokion sarbide erabakia hartzeko.



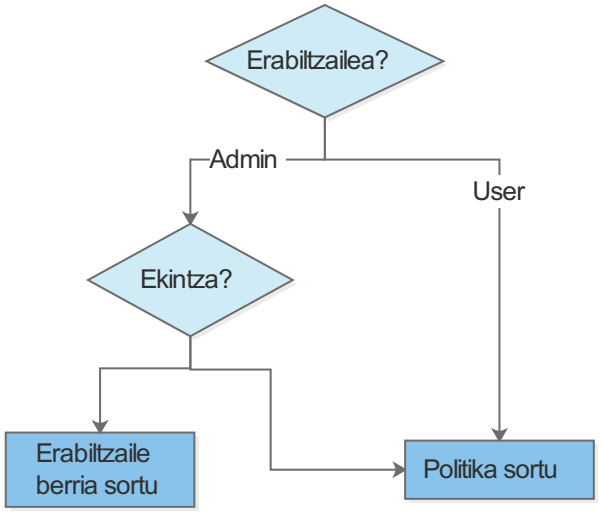

*Irudia 6: moduluen arteko erlazioa*

Beraz, lan honetan garatuko diren hiru moduluak hurrengoak dira: erabiltzaile interfaze grafikoa, kodifikatzailea modulua eta dekodifikatzaile modulua.

### 8.1.1. Erabiltzaile-interfaze grafikoa

Aurretik aipatu den legez, lan honen garapenaren lehen modulua erabiltzaile interfaze grafiko bat izango da, segurtasun politikak definitu eta gorde ahal izateko. Interfaze honen helburua Hidra protokoloa erabilgarria izatea da, beraz, interfazea erabilerraza izatea bermatu behar da. Era horretan, baimen egokia duten pertsonek interfaze hau erabili ahalko dute beharrezkoak diren politikak sortzeko.

Alde batetik, interfaze honetara sarbide kontrola izatea garrantzitsua da, izan ere, pertsona baimenduak dira politikak sortu eta kudeatzeko gaitasuna izan behar duten bakarrak. Era horretan, rol desberdin bi definitu dira interfaze honetara sarbidea edukitzeko. Alde batetik, erabiltzaile rola duten pertsonak egon daitezke, eta hauen funtzio bakarra politika berriak sortzea izango da. Beste alde batetik, administratzaile rola duten erabiltzaileak egon daitezke, eta hauek, politika berriak sortzeaz gain, erabiltzaile berriak sortzeko gaitasuna ere izango dute, bai erabiltzaile normalak, bai administratzaileak. 7. irudian interfaze grafikoaren hasierako pausoak adierazten dira, bertan rol desberdin biak eta haien funtzio posibleak azalduz.

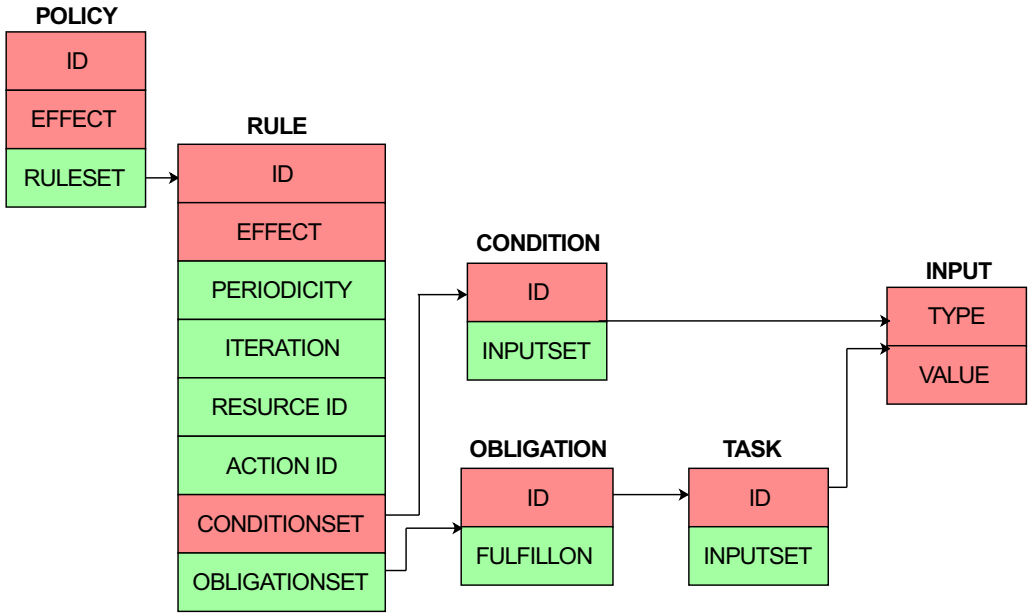



*Irudia 7: erabiltzaile motak eta funtzioak*

Beste alde batetik, esan bezala, interfaze grafikoaren funtzio nagusia politika berriak definitzea da, politika bakoitzean arau edota baldintza desberdinak sartuz. Gainera, sortuko diren politika guztiek politika-lengoaia jakin bat jarraitzen dute.

Segurtasun politikak sortzeko erabiltzen den lengoaia egitura desberdinez osatuta dago. Politika baten barruan arau multzo bat sartzen da, baldintza jakin batzuk egiaztatzeko eta dagozkion eraginak adierazteko. Horretarako, politika lengoaia egitura desberdinetan antolatzen da. Horietako egitura batzuk derrigorrezkoak dira politikaren barruan, beste batzuk, ordea, hautazkoak. Gainera, egitura batzuk beste batzuen barruan kateatzen dira, oinarrizko *policy* egituratik abiatuz. Horrela, luzera eta eduki desberdinetako politikak definitzea ahalbidetzen da.

8. irudian politika lengoaiak dituen egitura desberdinak eta haien arteko erlazioak adierazten dira. Esan bezala, egitura edota eremu batzuk derrigorrezkoak dira politikaren barruan, eta beste batzuk, aldiz, ez. Beraz, desberdintasun hori adierazteko, kolore desberdinak erabili dira. Alde batetik, gorriz adierazitako eremuak derrigorrezkoak dira, nahitaez agertu behar direnak. Beste alde batetik, berdez adierazita daudenak hautazkoak dira, erabiltzaileak erabaki ditzakeenak gehitu nahi dituen ala ez.

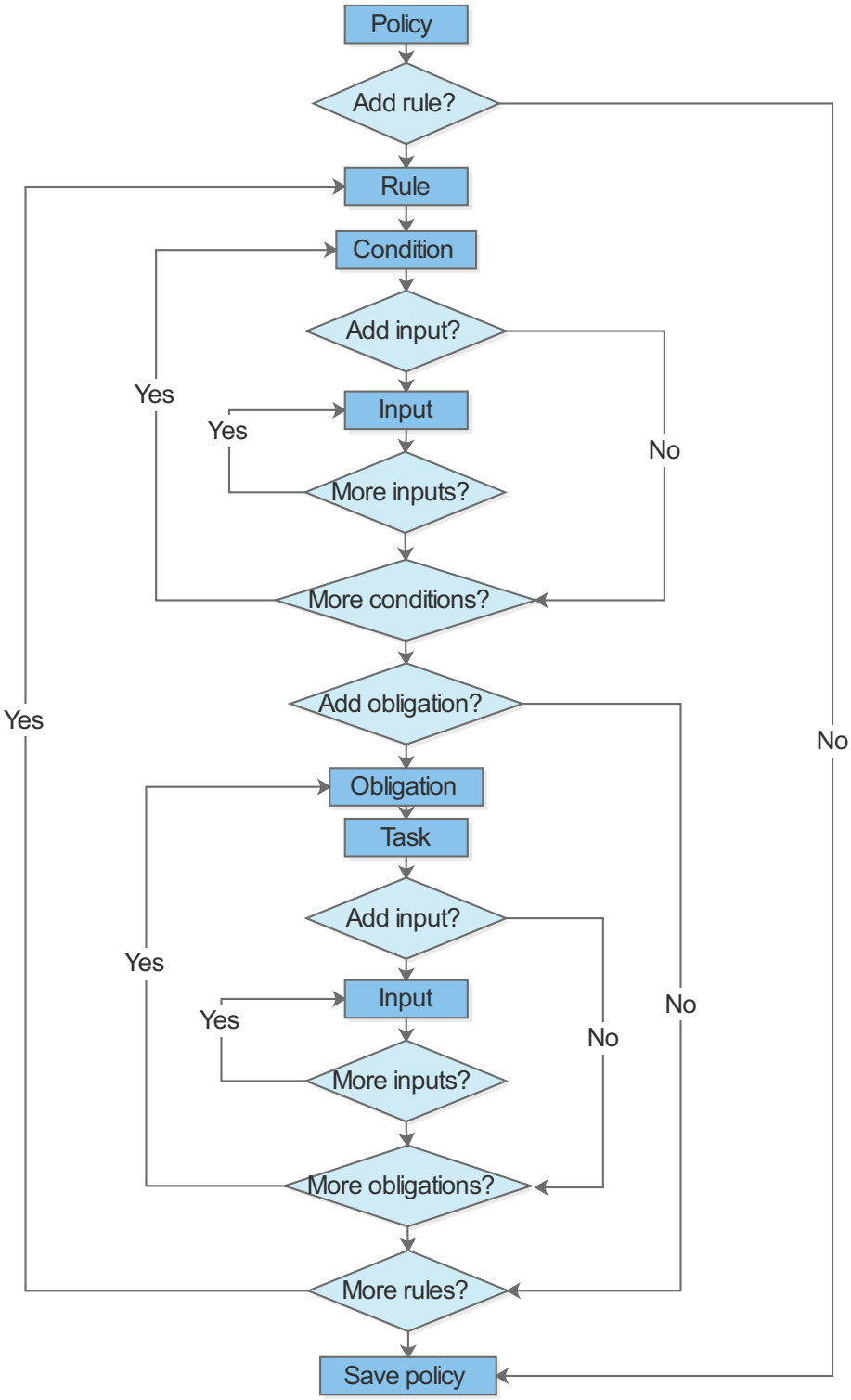



*Irudia 8: politika lengoaiaren egiturak*

Beraz, 8. irudiko eskeman adierazten den egitura osoa kontuan harturik, politika berrien definizio-prozesua diseinatu daiteke, interfaze-grafikoak jarraitu behar duena.

**POLITIKA BERRIEN DEFINIZIOA**

Esan bezala, interfaze-grafikoak segurtasun-politika berriak definitzea ahalbidetzen du, 8. irudiko lengoaia aplikatuz eta beharrezkoa den ordena jarraituz. Beraz, hori kontuan harturik, 9. irudian politika berri bat sortzeko prozesuaren fluxu-diagrama adierazten da. Bertan, prozesuan zehar ematen diren pausoak zeintzuk diren erakusten da, eta oso garrantzitsua da bertan adierazitako pauso guztiak eta haien arteko ordena zehaztasunez jarraitzea interfaze grafikoa garatzeko orduan.

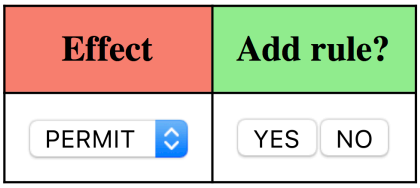



*Irudia 9: politiken sorreraren fluxu-diagrama*

Fluxu diagraman ikus daitekeen legez, politika bat sortzen hasi bezain laster eman behar den lehen pausoa erabiltzaileari *policy* egituraren eremuen balioak sartzeko eskatzea da. Egitura hau lengoaiaren oinarrizko egitura da, edozein politika bertatik abiatzen delako eta politikarik laburrenak ere gutxienez egitura hau izan behar duelako.

Egitura honek 2 eremu nagusi ditu, biak derrigorrezkoak. Lehen eremua *id* eremua da, politika bakoitza zenbaki batekin identifikatzen duena. Bigarren eremua *effect* da, eta honek politikaren lehenetsitako eragina zein izan behar den zehazten du. Politika baten ez bada beste araurik definitzen edota agertzen diren arauetan kontraesanen bat agertzen bada, sentsoreak eragin hau jarraitu beharko du. Eremu honen balio posibleak onartu (*permit*) edo ezeztatu (*deny*) dira. Horrez gain, *policy* egiturak *ruleset* izeneko egitura bat izan dezake bere barruan, hainbat arau desberdinez osatua.

Beraz, pauso honetan erabiltzaileari eskatzen zaion derrigorrezko eremu bakarra *effect* izango da, *id* eremua automatikoki sortzen baita, politika bere osotasunean identifikatzeko. Horrez gain, erabiltzaileari *rule*-ren bat gehitu nahi al duen galdetu behar zaio. Ezezko kasuan, politika bukatutzat ematen da, datu-basean eta XML fitxategian biltegiratuz. Hurrengo irudian interfaze grafikoaren lehen pauso hau adierazten da.



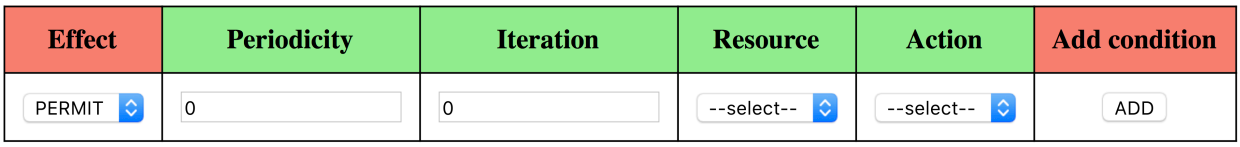

*Irudia 10: interfaze grafikoa, policy egitura*

Baiezko kasuan, interfaze grafikoaren bigarren pausoan *rule* egiturako eremuak sartu beharko ditu erabiltzaileak, horien bitartez arau desberdinak gehituz politikara. Rule bakoitzak hainbat eremu eta azpiegitura ditu bere barruan:

- ***Id*** (derrigorrezkoa): Eremu honek arau bakoitza identifikatzen du.
- ***Effect*** (derrigorrezkoa): *Rule* baten lehenetsitako eragina adierazten du.
- ***Periodicity*** (hautazkoa): Araua zenbatero ebaluatu behar den adierazten du.
- ***Iteration*** (hautazkoa): Araua zenbat aldiz ebaluatu behar den adierazten du.
- ***Resource*** (hautazkoa): Atzitu nahi den sentsoreko baliabidea adierazten du.
- ***Action*** (hautazkoa): Atzitu nahi den sentsoreko baliabideari dagokion akzioa adierazten du. Aukera posibleak *GET*, *POST*, *PUT*, *DELETE* eta *ANY* dira.
- ***Conditionset*** (derrigorrezkoa): Baldintza edo espresio desberdinen array bat da. Derrigorrezkoa da *rule* bakoitzak gutxienez baldintza bat edukitzea.
- ***Obligationset*** (hautazkoa): Betebehar desberdinen array bat da, nahitaez bete behar diren funtzioak identifikatzeko.

Beraz, hori kontuan harturik, interfaze grafikoan hautazkoak diren eremuak (*iteration, periodicity, resource* eta *action*) hutsik uzteko aukera egongo da. Eremu horiez gain, *effect* eremua

derrigorrez bete behar da. Behin behar diren eremuak bete direnean, hurrengo pausoa nahitaezkoa den *condition*-aren egitura gehitzea izango da. Bigarren pausoaren itxura interfaze grafikoaren barruan hurrengo irudian adierazten da.

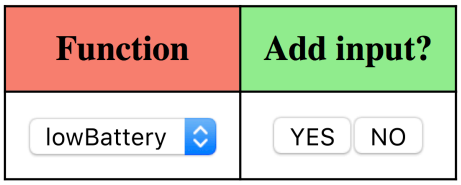



*Irudia 11: interfaze grafikoa, rule egitura*

Hirugarren pausoa, gutxienez *condition* edo baldintza bat gehitzea derrigorrezkoa denez, egitura honen eremuak betetzea izango da. Kasu honetan, baldintza bakoitzak izan ditzakeen eremuak hurrengoak dira:

- ***Function*** (derrigorrezkoa): Ebaluatu behar den baldintzari dagokion identifikazio zenbakia adierazten du. Existitzen diren funtzio guztiak .txt fitxategi batean biltegiratzen dira, eta interfazeak bertan dauden funtzioak adierazi behar ditu erabiltzaileak bat aukera dezan. Adibidez, existitu daitezkeen funtzio posibleak hurrengoak dira: *lowBattery, contains, isTrue…*
- ***Inputset*** (hautazkoa): Funtzioari gehitu ahal zaizkion sarrerako parametroen array bat da.

Beraz, baldintza bakoitzaren eremuak zeintzuk diren kontuan hartuz, interfaze grafikoan *function* eremua betetzeaz gain, erabiltzaileari *input* bat gehitzeko aukera emango zaio, hurrengo irudian adierazten den legez:

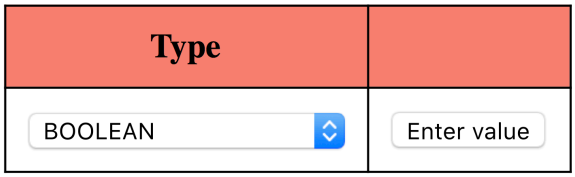



*Irudia 12: interfaze grafikoa, condition egitura*

Baiezko kasuan, hurrengo pausoa input egitura betetzea izango da, baldintzari sarrerako parametroak gehitzeko. Egitura honek 2 eremu desberdin ditu, biak derrigorrezkoak:

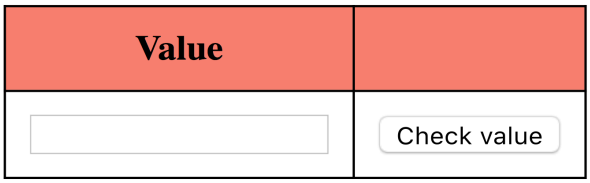
- ***Type*** (derrigorrezkoa): Parametroaren mota identifikatzen du eremu honek. Mota posibleak *boolean, byte, integer, float, string, request reference, system reference* eta *local reference* dira.
- ***Value*** (derrigorrezkoa): Parametroak hartzen duen balioa identifikatzen du eremu honek.

Beraz, interfaze grafikoak egin behar duen lehenengo gauza erabiltzaileari *input*-aren mota eskatzea da, hurrengo irudian adierazten den bezala.




*Irudia 13: interfaze grafikoa, input egituraren mota*

Ondoren, interfazeak erabiltzaileak aukeratutako mota identifikatu behar du, eta horren arabera parametroari balio bat ematea eskatuko dio. Oso garrantzitsua da mota bakoitzaren arabera balio egokia sartzen duela bermatzea, adibidez, ezin da *integer* mota aukeratu eta balio moduan 4,5 zenbakia sartu. Beraz, frogapen hau egitea beharrezkoa da. Hurrengo irudian balioak sartzeko era bat adierazten da, mota *integer* den kasuetarako.




*Irudia 14: interfaze grafikoa, policy egituraren balioa*

Behin *input* bat sortu denean, erabiltzaileari *input* gehiago gehitu nahi dituen galdetuko zaio. Baiezko kasuan, berriro ere prozesu hau errepikatuko da, erabiltzaileak *input* gehiago ez dituela nahi erabaki arte.

Beste alde batetik, *input*-ik sartu nahi ez bada, edo erabiltzaileak *input* gehiago ez dituela sartu nahi erabakitzen duenean, *condition* baten egitura amaitutzat ematen da. Horren ondoren, erabiltzaileari beste *condition* bat gehitzeko aukera ematen zaio, hala erabakiz gero berriro egitura honen sorreraren hasierara bueltatuz.

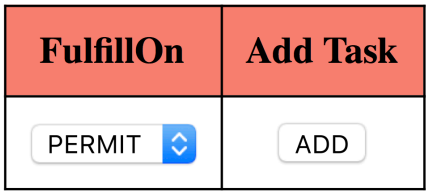
*Condition* gehiago ez dituela nahi erabakitzen badu, ordea, *obligation* egituraren bat gehitzeko aukera egongo da. Egitura hau hautazkoa denez, erabiltzaileari galdetuko zaio berak erabakitzeko gehitu nahi duen ala ez, hurrengo irudian ikusten den moduan.


*Irudia 15: obligation egitura gehitzeko aukera*

*Obligation*-en bat gehitzeko erabakia hartzen badu, bere eremuei balioa eman behar zaie. *Obligation* bakoitzak izan ditzakeen eremuak hurrengoak dira:

- ***Task*** (derrigorrezkoa): Egitura honek *obligation*-aren barruan zer egin behar den adierazten du.
- ***FulfillOn*** (hautazkoa): Eremu honek egitura zein kasutan bete behar den adierazten du. Eremuaren balio posibleak onartu (*permit*) eta ezeztatu (*deny*) dira.

Beraz, lehenengo eta behin erabiltzaileak *fulfillOn* eremua betetzeko aukera izango du. Ondoren, *task* egitura derrigorrezkoa denez, hau da, *obligation* bakoitzak *task* bat izan behar duenez, *task* egitura sortzeko botoi bat egongo da, hurrengo irudian adierazten den legez.

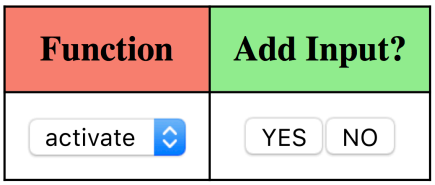



*Irudia 16: interfaze grafikoa, obligation egitura*

Hurrengo pausoa, ondorioz, *task* egitura betetzea da, *obligation* bakoitzak *task bat izan behar duelako nahitaez. Task* egituraren barruan dauden eremuak hurrengoak dira:

- ***Function*** (derrigorrezkoa): Eremu honek *obligation-ean* bete behar den funtzioa zein den identifikatzen du id baten bitartez.
- ***Inputset*** (hautazkoa): Sarrerako parametroko atributuen array bat da, baldintzen egituran onartzen zenaren berdina.

Beraz, *task* egituraren eremuak kontuan harturik, interfaze grafikoan *function* eremua bete beharko da lehenengo eta behin. Ondoren, *input*-en bat gehitzeko aukera emango zaio erabiltzaileari, hurrengo irudian adierazten den legez.




*Irudia 17: interfaze grafikoa, task egitura*

Baiezko kasuan, *input*-ak gehitzeko *condition* egitura ematen zen prozesu berdina ematen da. Hau da, lehenengo eta behin *input* baten mota aukeratu behar da. Ondoren, mota bakoitzaren arabera balio egokia sartu behar da, balioa motarekin bat datorrela bermatuz. Bukatzeko, erabiltzaileari *input* gehiago gehitu nahi dituen galdetuko zaio. Baiezko kasuan, *input* bat sortzearen prozesua berriro hasiko da.

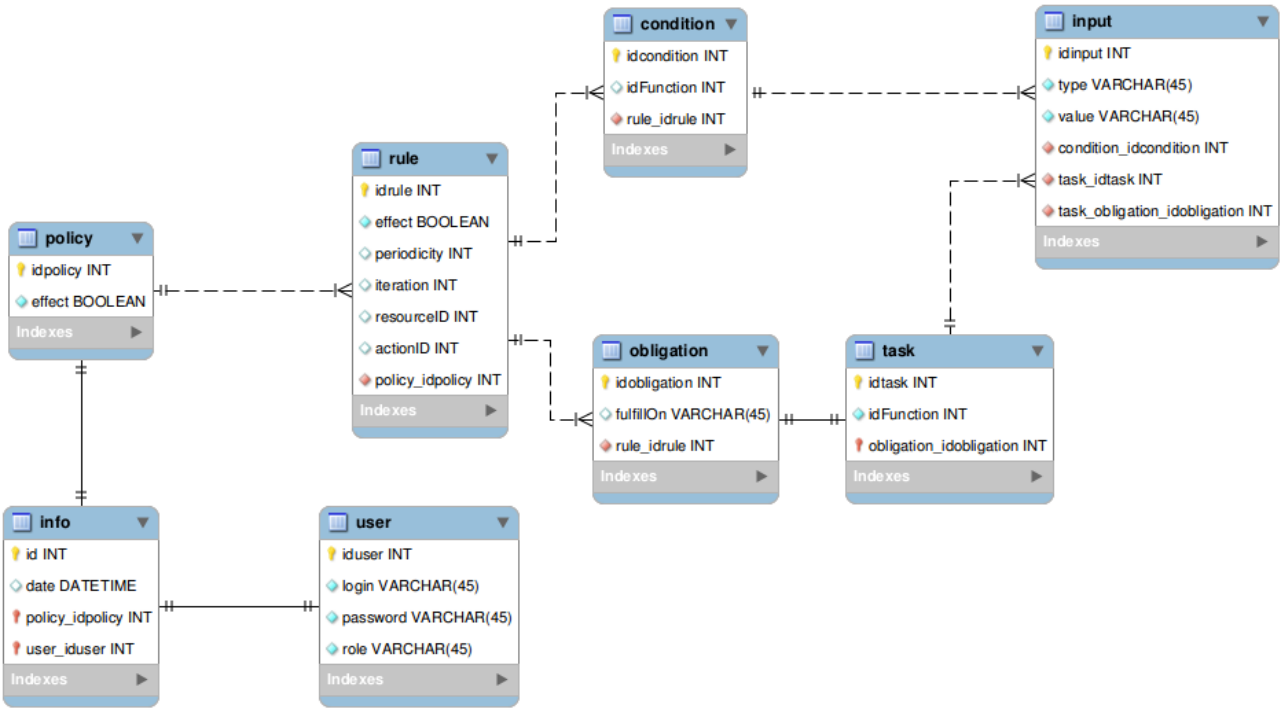
Ezezko kasuan, edota *input* gehiago gehitu nahi ez bada, *obligation* egitura amaitutzat ematen da, eta beste *obligation* berri bat gehitzeko aukera egongo da.

Erabiltzaileak ez badu *obligation* egiturarik nahi edo *obligation* nahikoa gehitu dituenean, *rule* osoa amaitutzat ematen da. Honen ondoren, erabiltzaileari beste *rule* egituraren bat nahi al duen galdetuko zaio. Baiezko kasuan, berriro *rule* oso baten sorreraren hasierako puntura bueltatu behar da, beste arau-egitura berri bat sortzeko. Ezezko kasuan, politika osoa bukatutzat ematen da, beraz, politika datu-basean eta XML fitxategian gorde behar da.

**POLITIKEN BILTEGIRATZEA DATU-BASEAN**

Esan bezala, sortzen den politika formatu desberdin bitan biltegiratu behar da. Alde batetik, politikaren informazioarekin XML fitxategi bat sortu behar da, ondoren kodifikatzeko erabiliko dena. Beste alde batetik, politika datu-base batean gorde behar da, politikarekin erlazionatutako informazio guztia biltegiratzeko: edukia, sortu duen erabiltzailea, sorrera-data… Horrez gain, datu-base berdinean interfaze-grafikoa erabiltzeko baimena duten erabiltzaileen informazioa ere biltegiratu behar da. Taula horri esker, interfaze grafikoan sarbide kontrola gauzatu ahalko da, baimenik gabeko erabiltzaileen sarbideak ezeztatzeko.

Kasu honetan, erabiltzaile-interfaze grafikoan definitzen diren politika guztiak biltegiratzeko MySQL datu-basean aukeratu da. 18. irudian ikus daitekeen legez, datu-basea sortzeko 8. irudiko lengoaiaren egitura jarraitzen da, egitura bakoitzeko taula desberdin bat sortuz eta haien artean behar diren erlazio zuzenak esleituz. Horrez gain, egitura guztietako eremu bakoitzaren datu-mota egokia zehaztea ere beharrezkoa da.




*Irudia 18: datu-basearen eskema*

Irudi honetan ikus daiteke ez direla *conditon-etako input* egiturak eta *task*-etako *input* egiturak gordetzeko bi taula bereiztu, taula bakarra baizik. Kasu bietan *input* egiturako eremuak berdinak direnez, taula berdina erabiliz gero memoria aldetik optimizazioa lortzen da. Hala ere, taula berdina erabiltzen denez, beharrezkoa da gehitzen den *input* bakoitza zein egiturakoa den adieraztea. Beraz, horretarako Foreign Key-ak (FK) erabili dira, kasu bakoitzean FK bakarrak izango baitu balioa (*condition* edo *task*), eta bestea hutsik egongo da.

Gainera, 18. irudian ikus daitekeen legez, politika bakoitzaren informazioa sartzeaz gain, politika horren sorrerari buruzko informazioa ere gorde behar da. Horretarako, beste bi taula gehitu dira datu basera.

Lehen taula *user* izeneko taula da. Taula honek baimena duten erabiltzaileen informazioa gordetzeko balio du, haien erabiltzaile-izena, pasahitza eta rola adieraziz. Rol eremuan aukera posible bi existitzen dira: administraria edo erabiltzailea. Rol biek dute politika berriak definitzeko aukera, baina, administrariak bakarrik dauka erabiltzaile berriak sortzeko baimena.

Bigarren taula *info* izeneko taula da. Taula honek politika bakoitzaren informazio orokorra gordetzen du, politika bakoitza noiz eta zein erabiltzailek sortu duen adieraziz. *Policy* taularen eta *info* taularen arteko erlazioa derrigorrezkoa eta 1:1 motakoa da, sortzen den politika bakoitzeko beharrezkoa delako informazioa gehitzea. Beste alde batetik, *info* taularen eta *user* taularen arteko erlazioa ere derrigorrezkoa eta 1:N motakoa da, erabiltzaile berdinak hainbat politika sortu ahal dituelako.

### 8.1.2. Kodifikatzailea eta dekodifikatzailea

Behin segurtasun politikak sortuta daudenean, sentsorera bidali aurretik beharrezkoa da politikak adierazpen bitar jakin batera kodifikatzea. Horretarako, kodifikazio bitar jakin bat erabili behar da, XML formatuan dagoen politikaren informazio guztia irakurri eta interpretatu ondoren, edukiaren arabera eta kodifikazio bitar bat jarraituz bit sekuentzia jakin bat sortzeko. Era berean, kontrako prozesua egiten duen dekodifikatzaile modulu bat ere behar da, sentsoreak heltzen zaion politikaren adierazpen bitarra interpretatu ahal izateko, eta horri esker sarbide-erabaki bat hartu ahal izateko.

Funtzio honetarako aukeratu den kodifikazio bitarra APBR (*Authorization Policy Binary Representation*) da, izan ere, honek konpresio maila altua eskaintzen du beste adierazpen bitar batzuekin konparatuz gero. Konpresio maila altu honi esker, politikei dagozkien adierazpen bitarrak laburragoak izango dira, eta honek CDSetako memoria gaitasunean inpaktu txikiagoa izango du, baliabideak hobeto aprobetxatuz.

Kodifikazio mota honek politikako egiturak era bitarrean adierazten ditu, eremu bakoitzarentzat bit kopuru desberdinak erabiliz. Horrez gain, eremu eta egitura batzuk hautazkoak direnez, *flag* batzuk gehitzen dira bit sekuentziara (bit injektatuak), hautazkoak diren egiturak agertzen direnean aktibatzeko, edota mota bateko egitura kopurua zein den adierazteko.

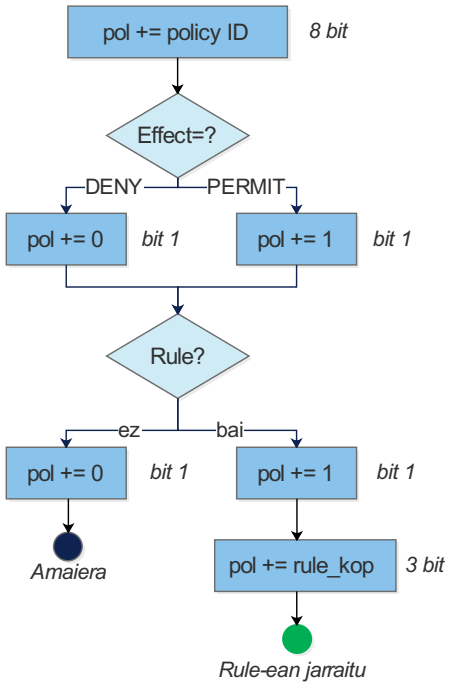
Lehenengo eta behin, policy egitura kodifikatu behar da. Horretarako, 11. taulan adierazten diren arauak jarraitu behar dira. Hasteko, politikaren IDa bitarrera itzuli behar da, horretarako 8 bit erabiliz. Ondoren, 9. bita *effect* eremuarentzat izango da, 0=DENY eta 1=PERMIT izanik.

9 bit horiekin *policy* egitura kodifikatuta dago. Hala ere, beste bit bat gehitu behar da sekuentziara, eta bit honek *rule*-rik existitzen den adieraziko du. Bit hori 0 bada, *rule*-rik ez dagoela esan nahi du, eta adierazpen bitarra bertan bukatuko da. Bit hori 1 bada, ordea, *rule*-ren bat dago, eta sekuentzia sortzen jarraitu behar da. Horretarako, 3 bit gehitzen zaizkio sekuentziari, eta horien bitartez *rule* kopurua zein den adierazten da, 0-7 arteko zenbaki batekin era bitarrean.

| Eremua | Izaera | Adierazpen bitarra | Deskribapena |
|---|---|---|---|
| ID | Derrigorrezkoa | 0b00000000 - 0b11111111 | [0-255] |
| *Effect* | Derrigorrezkoa | 0b0 - 0b1 | 0=DENY, 1=PERMIT |
| | Injektatua | 0b0 - 0b1 | *Rule*-ak existitzen diren |
| | Injektatua | 0b000 – 0b111 | *Rule* kopurua [0-7] |
| *Ruleset* | Hautazkoa | ***9. taulan*** | |

*Taula 8: policy egituraren kodifikazioa*

*Policy* egituraren kodifikazio-prozesuan azaldutako pauso guztiak 19. irudiko fluxu-diagraman deskribatzen dira. Gainontzeko egitura guztien kodifikazio prozesuaren fluxu-diagramak V. eranskinean adierazita daude.




*Irudia 19: policy egituraren kodifikazioaren fluxu-diagrama*

Ondoren, aurreko 3 bitetan adierazitako kopuruko *rule* bakoitza kodifikatu behar da, beraz, orain azalduko den prozesua behar beste aldiz errepikatu behar da, *rule* bakoitza dagokion moduan kodifikatuz.

*Rule* bat kodifikatzeko arauak 9. taulan adierazten dira. Lehenengo eta behin, *rule*-aren identifikatzaile zenbakia era bitarrean adierazi behar da, horretarako 8 bit erabiliz. Ondoren, arauaren lehenetsitako *effect* eremurako bit bat erabiltzen, da, 1=PERMIT eta 0=DENY izanik. Jarraian, 5 bit injektatzen dira hautazkoak diren 5 eremuak existitzen diren adierazteko. Beraz, bit bakoitzak hautazko eremu bati egiten dio erreferentzia, eremu bat existitzen bada, horri dagokion bita aktibatu behar da. Adibidez, *rule* batek *periodicity* eta *obligationset* badauka, bit hauen balioa 10001 izango da.

Ondoren, *periodicity*, *iteration*, *resource* eta *action* eremuak kodifikatu behar dira, existitzen direnean. Lehen 3 kasuetan 8 bit erabiltzen dira, eta *action* eremurako 3 bit bakarrik, 5 aukera posible onartzen dituelako: GET, POST, PUT, DELETE eta ANY.

Hautazkoak diren 4 eremu horiek kodifikatu ondoren, *conditionset* egitura kodifikatu behar da, gutxienez beti baitago *condition* bat *rule* bakoitzaren barruan. Beraz, lehenengo eta behin 3 bit erabiltzen dira exisititzen diren baldintza kopurua adierazteko. Ondoren, *condition* egitura kodifikatu behar da behar beste aldiz, eta horretarako 10. taulan adierazten diren arauak jarraitzen dira, aurrerago azalduko den moduan.

Bukatzeko, *obligation*-en bat egonez gero, egitura hori kodifikatzea izango da azken pausoa. Hala ere, hori kodifikatzen hasi aurretik, 3 bit erabiltzen dira existitzen diren *obligation* kopurua adierazteko. Ondoren, *obligation* bakoitza kodifikatu behar da behar beste aldiz, horretarako 11. taulako arauak jarraituz, aurrerago azalduko den legez.

*Obligation* egituraren kodifikazio osoa bukatzerakoan, *rule* egituraren kodifikazioa bukatutzat ematen da. Hala ere, prozesu hau errepikatu beharko da existitzen den *rule* bakoitzarentzat. Horrez gain, hurrengo ataletan *condition* eta *obligation* egituren eta haiei dagozkien azpiegituren kodifikazioa azalduko da.

| Eremua | Izaera | Adierazpen bitarra | Deskribapena |
|---|---|---|---|
| ID | Derrigorrezkoa | 0b00000000 - 0b11111111 | [0-255] |
| *Effect* | Derrigorrezkoa | 0b0 - 0b1 | 0=DENY, 1=PERMIT |
| | Injektatua | 0b00000 – 0b11111 | *Periodicity* = 0b10000<br>*Iteration* = 0b01000<br>*Resource* = 0b00100<br>*Action* = 0b00010<br>*Obligations* = 0b00001 |
| *Periodicity* | Hautazkoa | 0b00000000 - 0b11111111 | 0-255 minutu |
| *Iteration* | Hautazkoa | 0b00000000 - 0b11111111 | 0-255 errepikapen |
| *Resource* | Hautazkoa | 0b00000000 - 0b11111111 | Baliabidearen IDa |
| *Action* | Hautazkoa | 0b000 – 0b111 | GET=000<br>POST=001<br>PUT=010<br>DELETE=011<br>ANY=100 |
| | Injektatua | 0b000 – 0b111 | Baldintza kopurua |
| *Conditionset* | Derrigorrezkoa | ***10. taulan*** | |
| | Injektatua | 0b000 – 0b111 | *Obligation* kopurua |
| *Obligationset* | Hautazkoa | ***11. taulan*** | |

*Taula 9: rule egituraren kodifikazioa*

Esan bezala, 13. taulan *condition* egituraren kodifikazioa nola egin behar den deskribatzen da, eta pauso hauek existitzen den baldintza bakoitzeko jarraitu behar dira. Lehenengo eta behin, 8 bit erabiltzen dira baldintzaren funtzioaren identifikazio zenbakia adierazteko. Ondoren, bit bat injektatzen da *input*-en bat existitzen den ala ez adierazteko. Baiezko kasuan, 3 bit erabiltzen dira *input* kopurua zein den adierazteko, eta ondoren, *input* bakoitza kodifikatu behar da, 13. taulan adierazten den moduan, aurrerago azalduko den legez.

| Eremua | Izaera | Adierazpen bitarra | Deskribapena |
|---|---|---|---|
| Funtzioa | Derrigorrezkoa | 0b00000000 - 0b11111111 | Funtzioaren IDa |
| | Injektatua | 0b0 - 0b1 | *Input*-ak existitzen diren |
| | Injektatua | 0b000 – 0b111 | Input kopurua |
| *Inputset* | Hautazkoa | ***13. taulan*** | |

*Taula 10: condition egituraren kodifikazioa*

Beste alde batetik, 11. taulan *obligation* bakoitza nola kodifikatu behar den adierazten da. Lehenengo eta behin, *obligation* bakoitzak *task* bat izan behar duenez, egitura hori kodifikatu behar da, 6. taulan agertzen diren arauak jarraituz. *Task* egitura kodifikatu ondoren, bit bat injektatzen da *fulfillOn* eremua existitzen den ala ez adierazteko. Existitzen bada, beste bit bat erabiliko da bere balioa adierazteko, 0=DENY eta 1=PERMIT izanik. Azken bit honekin *obligation* egitura bukatutzat ematen da, baina prozesu hau errepikatu beharko da existitzen den *obligation* bakoitzerako. Azken *obligation*-a kodifikatu denean, *rule*-aren kodifikazioa ere bukatutzat ematen da. Hala ere, hurrengo tauletan eta azalpenetan falta diren azpiegituren kodifikazioa deskribatuko da.

| Eremua | Izaera | Adierazpen bitarra | Deskribapena |
|---|---|---|---|
| *Task* | Derrigorrezkoa | ***12. taulan*** | |
| | Injektatua | 0b0 - 0b1 | *FulfillOn* existitzen den |
| *FulfillOn* | Hautazkoa | 0b0 – 0b1 | |

*Taula 11: obligation egituraren kodifikazioa*

Aurreko azalpenean deskribatu den legez, *obligation* egitura kodifikatzerakoan egin behar den lehen gauza *task* egitura kodifikatzea da, 12. taulan adierazten den moduan. Lehenengo eta behin, 8 bit erabiltzen dira aplikatu behar den funtzioa era bitarrean identifikatzeko. Ondoren, bit bat injektatzen da *input*-en bat existitzen den ala ez adierazteko. Ezezko kasuan, bertan amaitzen da *task* egitura baiezko kasuan, ordea, beste 3 bit gehitzen zaizkio *input* kopurua adierazteko. Ondoren, existitzen den *input* bakoitza kodifikatu behar da, 13. taulan azaltzen diren arauak jarraituz.

| Eremua | Izaera | Adierazpen bitarra | Deskribapena |
|---|---|---|---|
| Funtzioa | Derrigorrezkoa | 0b00000000 – 0b11111111 | Funtzioaren IDa |
| | Injektatua | 0b0 - 0b1 | Input-ak existitzen diren |
| | Injektatua | 0b000 – 0b111 | Input kopurua |
| *Inputset* | Hautazkoa | ***13. taulan*** | |

*Taula 12: task egituraren kodifikazioa*

13. taulan *input* egituraren kodifikazioa nola egin behar den adierazten da. Lehenengo eta behin, dagokion mota adierazteko 3 bit erabiltzen dira, 8 mota posible existitzen direlako: *boolean, byte, integer, float, string, request reference, system reference* eta *local reference.* Beraz, mota posible bakoitzari zenbaki bitar desberdin bat dagokio.

| Eremua | Izaera | Adierazpen bitarra | Deskribapena |
|---|---|---|---|
| Mota | Derrigorrezkoa | 0b000 – 0b111 | BOOLEAN=000<br>BYTE=001<br>INTEGER=010<br>FLOAT=011<br>STRING=100<br>REQUEST_REFERENCE=101<br>SYSTEM_REFERENCE=110<br>LOCAL_REFERENCE=111 |
| Balioa | Derrigorrezkoa | ***14. taulan*** | |

*Taula 13: input egituraren kodifikazioa*

Ondoren, *input*-aren balioa kodifikatu behar da, baina atal hau ez da era berean kodifikatzen mota guztietarako, hau da, motaren arabera arau bat edo beste bat jarraitzen da. Beraz, mota bakoitzari dagokion kodifikazio hurrengoa da:

- ***Boolean*:** Bit bakarra erabili behar da kodifikatzeko, 0=*false* eta 1=*true* izanik.
- ***Byte*:** 8 bit erabiltzen dira byte zenbakiaren balioa adierazteko, byte-etik bitetara konbertsioa eginez. Balio posibleen tartea 0-255 da.
- ***Integer*:** 16 bit erabiltzen dira mota honetako zenbakien balioa adierazteko, beraz, dagokion balioa 16 biteko formatura itzuli behar da. Balio posibleen tartea 0-65535 da.
- ***Float*:** 32 bit erabiltzen dira mota honetako zenbakien balioa adierazteko. Beraz, dagokion balioa 32 biteko formatura itzuli behar da. Balio posibleen tartea 0d-$1E^{-16}$ – 0d$1E^{16}$ da.
- ***String*:** Kasu honetan, lehenengo eta behin 3 bit erabiltzen dira karaktere kopurua zein den adierazteko, karaktere kopuru maximoa 6 dela kontuan harturik. Ondoren, karaktere bakoitzeko 8 bit erabiltzen dira. Beraz, luzera totala 3+8*karaktere_kopurua izango da.
- ***Request reference:*** Mota honetako balioak 8 bitekin adierazten dira. Mota hauetako atributuei esker, eskaerarekin erlazionatutako parametro batzuk konprobatzea ahalbidetzen da.
- ***System reference*:** Mota honetako balioak 8 bitekin adierazten dira.
- ***Local reference*:** Mota honetako balioak 3 bitekin adierazten dira, aurretik sortutako eta ebaluatutako *condition* bati egiten baitio erreferentzia. Horri esker, baldintza bat sarrerako parametro bezala erabiltzea ahalbidetzen da.

| Mota | Adierazpen bitarra | Deskribapena |
|---|---|---|
| BOOLEAN | 0b0 – 0b1 | False/True [0 - 1] |
| BYTE | 0b00000000 - 0b11111111 | [0 - 255] |
| INTEGER | 0d0 – 0d65535 (16 bit) | [0 - 65535] |
| FLOAT | $0d\text{-}1E^{-16}$ – $0d1E^{16}$ (32 bit) | |
| STRING | | 3+8*karaktere_kop |
| REQUEST_REFERENCE | 0b00000000 - 0b11111111 | IDa |
| SYSTEM_REFERENCE | 0b00000000 - 0b11111111 | IDa |
| LOCAL_REFERENCE | 0b000 – 0b111 | Baldintzaren IDa |

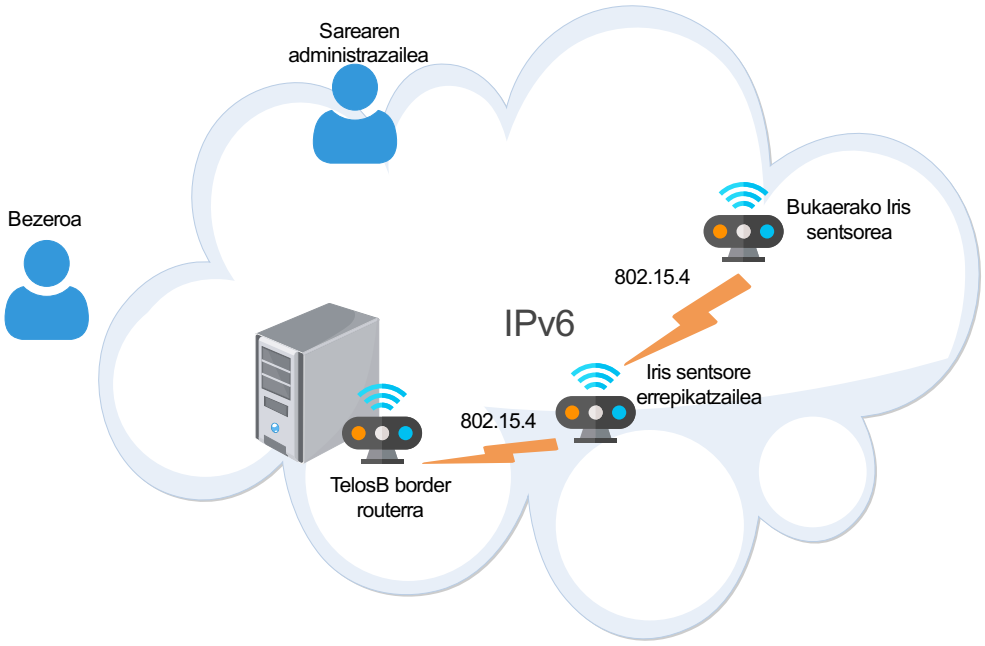
*Taula 14: input-eko balioen kodifikazioa*

Beraz, deskribatutako pauso guztiak jarraituz, eta pausoen arteko ordena egokia mantenduz, politikak kodifikatuko dira, bakoitzarentzat dagokion adierazpen bitar bat lortuz. Ondoren, politika hori sentsorera bidaltzeko prest egongo da.

Era berean, dekodifikatzailea diseinatzeko arau berdinak jarraitu behar dira, baina kontrako norabidean. Hau da, sentsoreak sekuentzia bitar bat jasoko du, eta deskonposatze prozesu bat hasiko du, bit multzo bakoitza informazio desberdin baterako erabiliz. Adibidez, jasotzen dituen lehen 8 bitak politikaren ID bezala interpretatuko ditu, hurrengo bita lehenetsitako efektu bezala, eta horrela sekuentzia osoa interpretatu arte.

## 8.2. Ebaluazio maketaren diseinua

Alternatiben analisiaren atalean deskribatu den bezala, lan honetan diseinatu eta garatuko den sistema osoaren ebaluazioa gauzatzeko, maketa erreal bat erabiliko da. Horri esker, lortzen diren emaitzak errealak eta fidagarriak izango dira, eta sistemaren balidazio zuzena egitea ahalbidetzen da. Horretarako, 20. irudian maketaren egitura orokorra deskribatzen da. Ondoren, 21. irudian interfaze grafikoarekin erlazionatutako prozesuak maketan nola erlazionatzen diren adierazten dira. Bukatzeko, 22. irudian bezero baten eskaerarekin erlazionatutako prozesuak maketaren gainean ikus daitezke.




*Irudia 20: ebaluazio-maketaren eskema*

Esan bezala, 20. irudian maketaren egitura ikus daiteke. Aurrerago azalduko diren prozesuen funtzionamendua ulertu ahal izateko, beharrezkoa da maketan parte hartzen duten elementuak ezagutzea.

Hasteko, maketa inplementatzeko WSN sare bat eratu behar da CDS desberdinekin. Horretarako, Iris sentsore-plataformak erabiliko dira, haien artean sare bat era dezaten. Kasu honetan, 2 plataforma erabiltzea erabaki da. Urrunago kokatzen den Iris sentsorea bukaerakoa izango da, hau da, bezeroak atzitu nahi duena E2E komunikazio seguru baten bitartez. Sentsore honetan Hidra protokoloa inplementatu behar da, bera izango delako sarbide kontrola gauzatzearen arduraduna. Hala ere, sentsore horretara heltzeko komunikazioa ezin da zuzena izan distantzia dela eta, beraz, tartean beste Iris plataforma bat kokatuko da, errepikatzaile funtzioak beteko dituenak bukaerako sentsorera atzitu ahal izateko. Horretarako, bezeroaren eskaera bukaerako sentsorera bideratzeko, tartean dagoen sentsoreak RPL bideratze protokoloa erabiltzen du. RPL protokoloari esker paketeak WSN sarean zehar bideratzen dira, eta bere funtzionamendu zehatza II. eranskinean deskribatzen da.

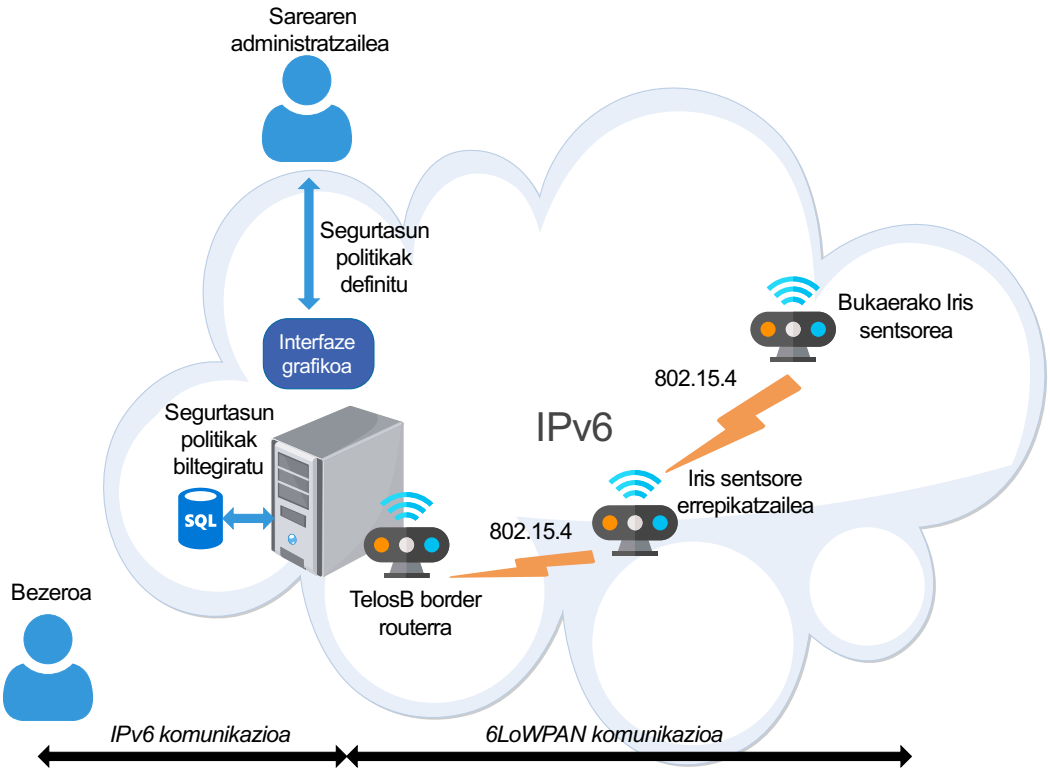
Hala ere, nahiz eta maketa honetako WSN sarean Iris sentsore bi bakarrik egon, maketa nahi beste handitu daiteke, RPL protokoloari esker sareko sentsore guztiak bezeroarentzat atzigarriak izango liratekeelako.

Eskeman ikusten den legez, CDSen arteko komunikazioa lotura mailan IEEE 802.15.4 protokoloaren bitartez egiten da, eta sare mailan IPV6 erabiltzen denez, 6LoWPAN protokoloa erabiltzen dute.

Beste alde batetik, zerbitzari bat behar da maketa honetan. Zerbitzari hau hainbat funtzio betetzearen arduraduna izango da, aurrerago azalduko den legez. Horrez gain, WSN sarerako *gateway* funtzioak egiten dituen gailu bat ere behar da, kasu honetan TelosB plataforma. Gainera, gailu honek IPv6 eta 6LoWPAN protokoloen arteko interfazea eskaini beharko du, bezeroaren eta sentsoreen arteko komunikazioa ahalbidetzeko. Horrez gain, *border router* funtzioak egin behar ditu gailu honek, heltzen diren paketeak WSNra bideratzeko. Maketa honen kasuan, TelosB *border routerra* zerbitzarian bertan konektatzen da.

Bukatzeko, rol bi desberdintzen dira. Alde batetik, bezero bat existitzen da, eta eskaerak egingo dituena izango da. Beste alde batetik, sarearen administratzaile bat ere existitzen da, eta hau izango da politikak sortzearen arduraduna.

Beraz, behin maketaren egitura orokorra deskribatuta, maketan parte hartzen duten prozesuen funtzionamendua azaldu behar da.

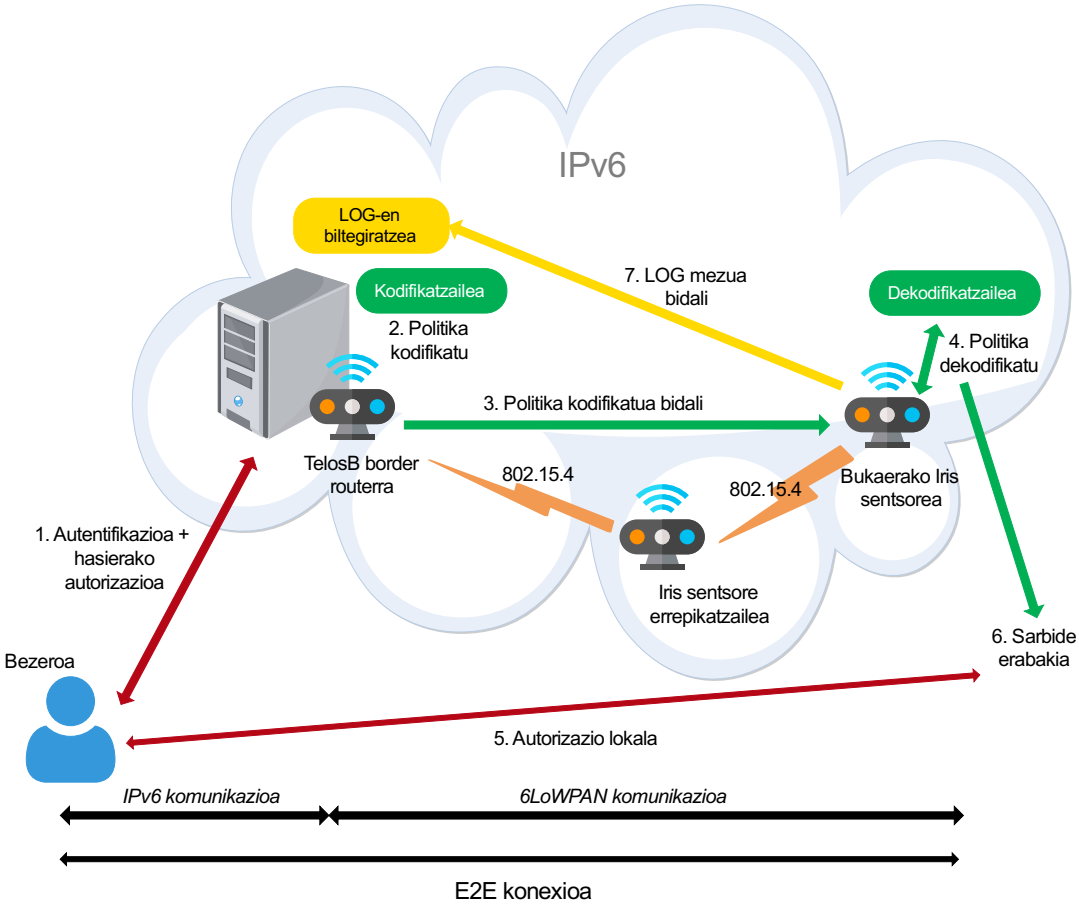



*Irudia 21: interfaze grafikoa maketan*

21. irudian interfaze-grafikoaren erabilera maketaren zein zatitan exekutatzen den eta nola ematen den prozesu hori adierazten da. Bertan ikus daitekeen legez, segurtasun politikak definitzeko balio duen interfaze grafikoa zerbitzarian exekutatzen da. Ondoren, erabiltzeko baimen egokia duen sarearen administratzaile batek interfaze grafiko hori erabiliko du behar duen segurtasun politika sortzeko.

Behin segurtasun politika definituta dagoenean, zerbitzariak politika hori datu-basean biltegiratu behar du. Horrez gain, beharrezkoa da politika XML fitxategi batean ere biltegiratzea ondoren kodifikatu ahal izateko. Bai datu-basea eta bai XML fitxategien biltegia zerbitzarian bertan daude.

Interfaze grafikoaren exekuzioan ez dute sentsoreek parte hartzen, hau da, zerbitzari zentralean ematen da prozesu osoa.




*Irudia 22: bezero baten eskaera maketan*

22. irudian bezero baten eskaerarekin erlazionatutako prozesua maketan nola ematen den adierazten da. Prozesu honetan, bezeroak urrunago dagoen Iris plataformatik datu batzuk lortu nahi ditu.

Hasteko, bezeroa zerbitzarira konektatu behar da, bertan exekutatzen delako Hidra protokoloaren ACS zerbitzaria. Beraz, 1. pausoan zerbitzariaren atal honek bezeroaren autentifikazioa eta hasierako autorizazioa gauzatuko ditu. Hau da, Hidra protokoloaren lehen eta bigarren faseak zerbitzari honetan gauzatzen dira. Ondorioz, zerbitzari honetan bezeroen eta CDSen kredentzialak biltegiratu behar dira, beharrezkoak diren ticketak eman ahal izateko.

Autentifikazioaren eta hasierako autorizazioaren emaitza positiboa izan dela suposatuz, CDSak bezeroaren autorizazio lokala gauzatu beharko du baldintza lokal batzuen arabera eta segurtasun politika bat aplikatuz. Beraz, lehenengo eta behin zerbitzarian kodifikatzaile modulua exekutatu behar da (2. pausoa), horretarako dagokion politika XML fitxategitik irakurriz. Kodifikatzaile honek dagokion segurtasun politika irakurriko du, eta adierazpen bitar egokira kodifikatuko du.

Ondoren, Hidra protokoloaren mezu baten barruan, beste informazio gehiagorekin batera, kodifikatutako politika bidaliko dio bezeroaren helburua den CDSari (3. pausoa). Beraz, CDS honek bezeroaren eskaera jasotzen duenean, dekodifikatzaile modulua exekutatuko du jasotako politikaren adierazpen biterretik jatorrizko edukia interpretatu ahal izateko (4. pausoa). Dekodifikatutako politika hori aplikatu beharko du, eta horren arabera sarbide-erabaki bat hartu beharko du bezeroak autorizazio lokalaren eskaera gauzatzen duenean (5. eta 6. pausoak).

Sarbidea onartzen bada, bezeroaren eta bukaerako CDSaren artean E2E komunikazio segurua ezartzen da. Ez bada onartzen, ostera, bezeroari ukapenaren berri ematen zaio eta ez da komunikaziorik ezarriko.

Hala ere, edozein kasutan CDSak zerbitzarira LOG mezu bat bidaltzen du, jasotako eskaeraren berri emateko. Zerbitzariak LOG modulua exekutatzen du jasotako informazio guztia biltegiratzeko (7. pausoa). Horrela, sarean ematen diren sarbide-eskaera guztien berri dauka zerbitzariak, bai onartutako eskaerenak eta bai ezeztatutakoenak.

# 9. Emaitzen deskribapena

Atal honetan proiektuan garatutako mekanismoen emaitzak deskribatuko dira. Horretarako, lehenengo eta behin modulu guztien balidazio funtzionala egingo da. Ondoren, sistema osoaren errendimenduaren balidazioa gauzatuko da.

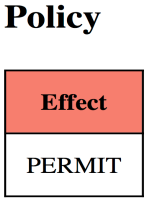
## 9.1. Balidazio funtzionala

Proiektu honen balidazio funtzionala egin ahal izateko, garatutako modulu bakoitzaren balidazioa egin behar da. Horretarako, proba desberdinak egingo dira interfaze grafikoaren, kodifikatzailearen eta dekodifikatzailearen funtzionamendua egiaztatzeko.

Balidazio hau egiteko 3 proba desberdin egingo dira, luzera desberdineko politikak erabiliz. Lehenengo eta behin, interfaze grafikoan politika bakoitza sortuko da, behar den moduan biltegiratzen dela bermatuz, bai datu-basean eta bai XML fitxategian. Ondoren, bezero baten eskaera martxan jarriko da, eta zerbitzariak sortu berri den politika hori kodifikatuko du, hortik lortzen den adierazpen bitarra aztertuz eta egokia dela egiaztatuz. Bukatzeko, sentsoreak politika kodifikatua jasotzen duenean, dekodifikatzailea exekutatuko du eta interpretatzen duena jatorrizko politikarekin bat datorrela bermatu beharko da.

### 9.1.1. Interfaze grafikoa eta politikaren biltegiratzea

Gauzatuko den lehenengo proba existitu daitekeen politikarik laburrenarekin egingo da, inolako *rule*-rik ez duen politika batekin. Ondorioz, politikak izango dituen eremu bakarrak ID (automatikoki sortzen dena) eta *effect* (interfaze grafikoan sortuko dena) izango dira. Behin interfaze grafikoan politika hori sortu denean, bere itxura grafikoa hurrengoa da:




*Irudia 23: politikaren lehen adibidea*

Ondoren, politika hori datu-basean gorde dela bermatu behar da. Kasu honetan, datu-baseak ID=1 esleitu dio politikari, sortu den lehen politika baita. Datu-basean ikus daiteke momentuz informazioa daukan taula bakarra *Policy* taularena eta *Info* taularena dela. Beraz, ikusten da biltegiratze-prozesua ondo egin dela, beharrezko eremuak bakarrik bete baititu. Gainera, *Info* taulak politika hori sortu duen erabiltzailearekin erlazionatzen du era egokian, eta politikaren sorrera-data ere ondo gordetzen du. Beste alde batetik, *policy1.xml* izeneko fitxategia ere era egokian sortzen da, politikaren informazioarekin.

Bigarren proban politika luzeago bat sortuko da interfaze grafikoan, *rule* bat gehituko zaiolako. *Rule* honek derrigorrezko *conditon*-a izango du, baina ez du inolako *obligation* egiturarik edukiko.

Horrez gain, *conditon* egiturak *input* bat izango du bere barruan. Beraz, eremu horiek interfaze grafikoan sartu ondoren, lortzen den politikaren adierazpena era grafikoan hurrengoa da:

**Policy**

| Effect |
|---|
| PERMIT |

**Created rules**

| Effect | Periodicity | Iteration | Resource | Action | Obligations | Conditions |
|---|---|---|---|---|---|---|
| DENY | | | | | | Function: isTrue; Inputs: Type: SYSTEM_REFERENCE; Value: onMaintenance |

*Irudia 24: politikaren bigarren adibidea*

Ondoren, politika hori datu-basean gorde dela bermatu behar da. Kasu honetan, datu-baseak ID=2 esleitu dio politikari, sortu den bigarren politika delako. Bertan ikus daiteke politikaren informazioa era egokian biltegiratu dela, beharrezko taulak betez: *Policy, Rule, Condition, Input* eta *Info*. Beste alde batetik, *policy2.xml* izeneko fitxategia ere era egokian sortzen da, politikak duen informazioarekin.

Hirugarren proban politika luzeena sortuko da interfaze grafikoan. Oraingoan ere *rule* bat gehituko zaio, baina kasu honetan *rule* honek derrigorrezko *conditon*-a izateaz gain, *obligation* egitura bat ere edukiko du. Gainera, *condition* egiturak *input* egitura bat ere izango du, baina *obligation*-ak ez. Beraz, eremu horiek interfaze grafikoan sartu ondoren, lortzen den politikaren adierazpena era grafikoan hurrengoa da:

**Policy**

| Effect |
|---|
| PERMIT |

**Created rules**

| Effect | Periodicity | Iteration | Resource | Action | Obligations | Conditions |
|---|---|---|---|---|---|---|
| PERMIT | | | | | FulfillOn: DENY; Task: Function: activate; Inputs: | Function: lowBattery; Inputs: Type: INTEGER; Value: 3 |

*Irudia 25: politikaren hirugarren adibidea*

Ondoren, politika hori datu-basean gorde dela bermatu behar da. Kasu honetan, datu-baseak ID=3 esleitu dio politikari, sortu den hirugarren politika delako. Bertan ikus daiteke politikaren informazioa era egokian biltegiratu dela, beharrezko taulak betez: *Policy, Rule, Condition, Input, Obligation, Task* eta *Info*. Beste alde batetik, *policy3.xml* izeneko fitxategia ere era egokian sortzen da, politikak duen informazioarekin.

Beraz, hiru proba hauekin interfaze grafikoaren eta politiken biltegiratzearen balidazio funtzionala gauzatu da, eta haien funtzionamendua egokia dela demostratu da. Hurrengo pausoa kodifikatzailearen eta dekodifikatzailearen balidazioa egitea da.

### 9.1.2. Kodifikatzailea eta dekodifikatzailea

Behin politika sortuta dagoenean, zerbitzariak kodifikatzailea exekutatu behar du, sortutako politika bakoitzaren XML fitxategia irakurri eta politika adierazpen bitar egokira itzultzeko. Proba hauek egiteko ere aurreko 3 adibideak hartuko dira, hau da, *policy1.xml, policy2.xml* eta *policy3.xml* fitxategietako politikak dira kodifikatu behar direnak.

Politiken luzerak desberdinak direnez, lortuko den adierazpen bitarraren luzerak ere desberdinak izango dira. Beraz, hurrengo taulan adibide bakoitzaren politika kodifikatuaren luzera zein den adierazten da:

| Politika | Bit kopurua | Byte kopurua |
|---|---|---|
| policy1.xml | 10 | 2 |
| policy1.xml | 53 | 7 |
| policy3.xml | 75 | 10 |

*Taula 15: kodifikatzailearen emaitzak*

15. taulan agertzen diren emaitzak teorikoki lortu beharko liratekeenak direla konprobatu da. Ondorioz, kodifikatzailearen funtzionamendua egokitzat hartu da, politiken adierazpen bitar egokiak lortzen baitira.

Bukatzeko, adierazpen bitar horiek bukaerako CDSra bidali dira, honek dekodifikatzailea exekuta dezan. Politika bakoitza dekodifikatu ondoren, proba desberdinak egin dira sentsoreak jatorrizko politikak ondo berreskuratzen dituela konprobatzeko, eta ikusi da kasu guztietan dekodifikatzaileak ere ondo funtzionatzen duela.

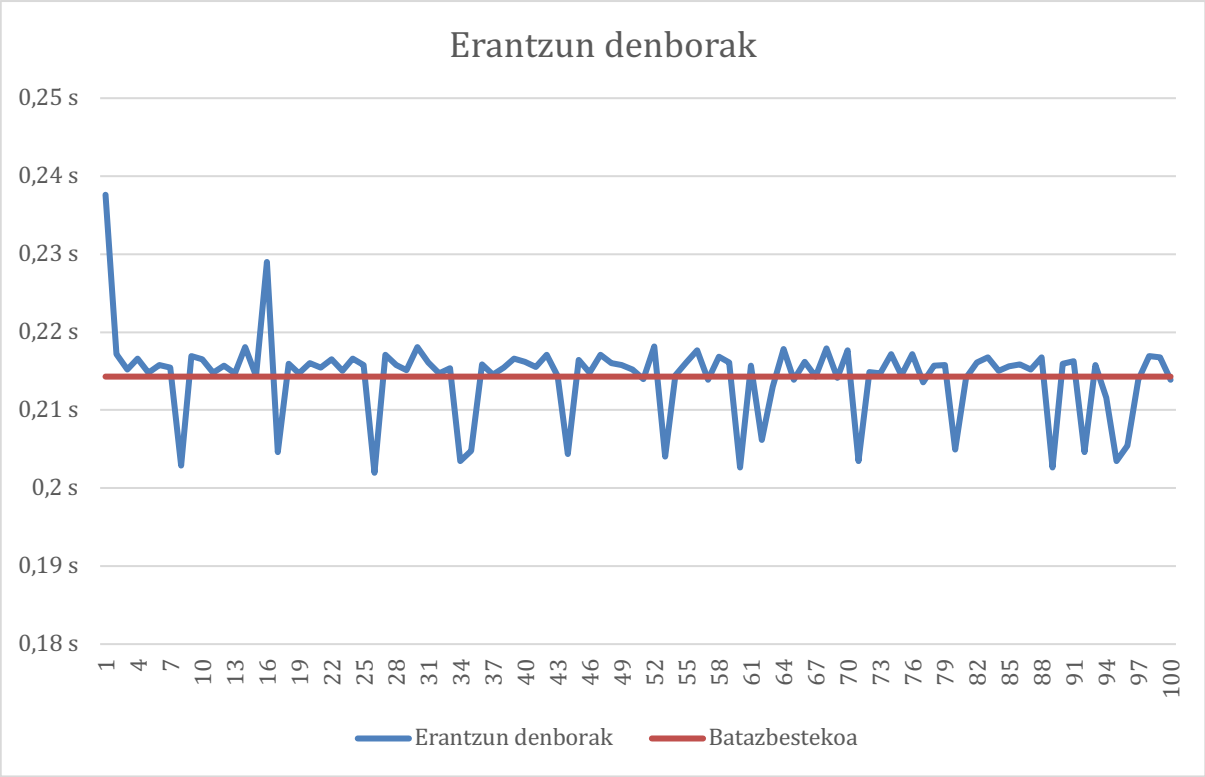
Ondorioz, proiektu honetan garatu diren 3 mekanismoen balidazio funtzionala gauzatu da, eta haien funtzionamendua egokia dela konprobatu da. Hala ere, atal honetan 3 probetako emaitzen laburpena baino ez da adierazi, baina VI. eranskinean emaitza zehatzen deskribapena ikus daiteke. Bertan, politikaren adibide bakoitzarekin lortzen den XML fitxategia, adierazpen bitarra… deskribatzen dira.

## 9.2. Errendimenduaren balidazioa

Sistemaren errendimenduaren balidazioa gauzatzeko, sistemaren erantzun-denbora neurtu behar da. Izan ere, bezero batek datu bat eskatzen duenetik CDSaren erantzun bat jasotzen duen arte pasatzen den denbora <1s izan beharko litzateke errendimendu egokia duela esateko. Horretarako, sistema osoa martxan jarriko da eta 100 proba desberdin egingo dira, bakoitzaren erantzun-denbora neurtuz. Hala ere, kontuan izan behar da sistemaren erantzun-denbora ez dela berdina izango jasotako politikaren luzera desberdina denean. Izan ere, politika luze bat kodifikatzeko eta dekodifikatzako behar den denbora luzeagoa izango da. Ondorioz, desberdintasun horiek aztertu ahal izateko, aurreko atalean erabili diren 3 politiken adibideak erabiliko dira, bakoitzarekin 100 proba eginik.

### 9.2.1. Lehenengo adibidea

Adibide honetan *policy1.xml* politika erabiliko da, hau da, 2 byte dituen politika. Bezeroaren autentifikazioa eta hasierako autorizazioa burutu direnean, zerbitzariak politika hori kodifikatuko du behar denean, bukaerako CDSra bidaliko du eta honek dekodifikazio-prozesua gauzatuko du. Ondoren, politika aplikatuko du sarbide-erabaki bat hartzeko eta bezeroari erantzun bat emango dio. Prozesu osoa 100 aldiz egin da, eta lortutako emaitzak hurrengo grafikan ikus daitezke:




*Irudia 26: Lehen adibidearen erantzun-denborak*

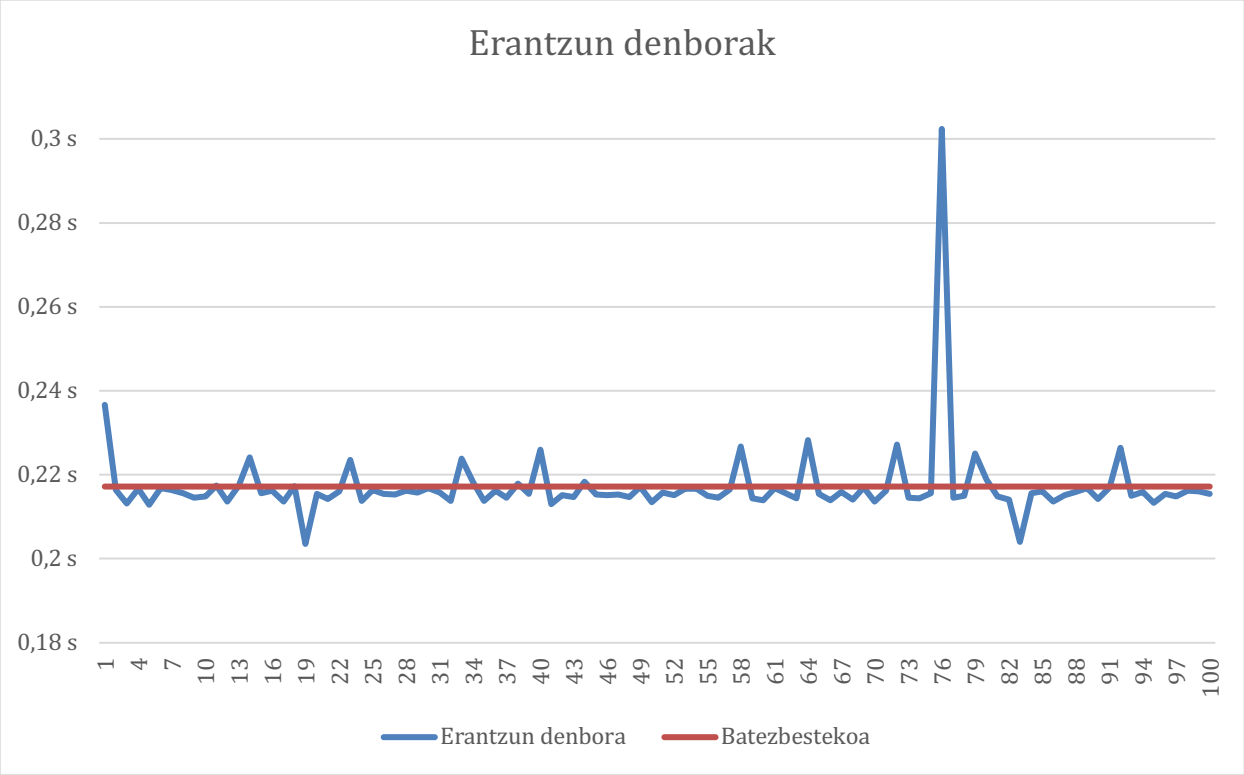
Lehenengo adibidearekin lortutako emaitzak 26. irudian daude adierazita. Bertan ikus daitekeen legez, erantzun-denbora guztian tarte onargarrian daude, segundo bat baino askoz ere txikiagoak baitira. Izan ere, batezbesteko erantzun-denbora 0,21429944 s-koa da. Gainera, proba hauetan eman den balio maximoa 0,237615 s-koa da eta minimoa 0,201967 s-koa da. Lortutako datu estatistikoen laburpena 16. taulan adierazten da.

| Maximoa | Minimoa | Batezbestekoa | Desbideratze tipikoa | Konfiantza tartea |
|---|---|---|---|---|
| 0,237615 s | 0,201967 s | 0,21429944 s | 0,005198113 s | 0, 212840312 s – 0, 215758568s |

*Taula 16: lehenengo adibidearen datu estatistikoak*

### 9.2.2. Bigarren adibidea

Adibide honetan *policy2.xml* politika erabiliko da, hau da, 7 byte dituen politika. Aurreko adibidean egin den moduan, prozesu osoa 100 aldiz egin da, eta lortutako emaitzak hurrengo grafikan ikus daitezke:




*Irudia 27: Bigarren adibidearen erantzun-denborak*

Bigarren adibidearekin lortutako emaitzak 27. irudian daude adierazita. Bertan ikus daitekeen legez, erantzun-denbora guztiak tarte onargarrian daude, segundo bat baino askoz ere txikiagoak baitira. Izan ere, batezbesteko erantzun-denbora 0,217204 s-koa da. Gainera, proba hauetan eman den balio maximoa 0,302363 s-koa da eta minimoa 0,203544 s-koa da.

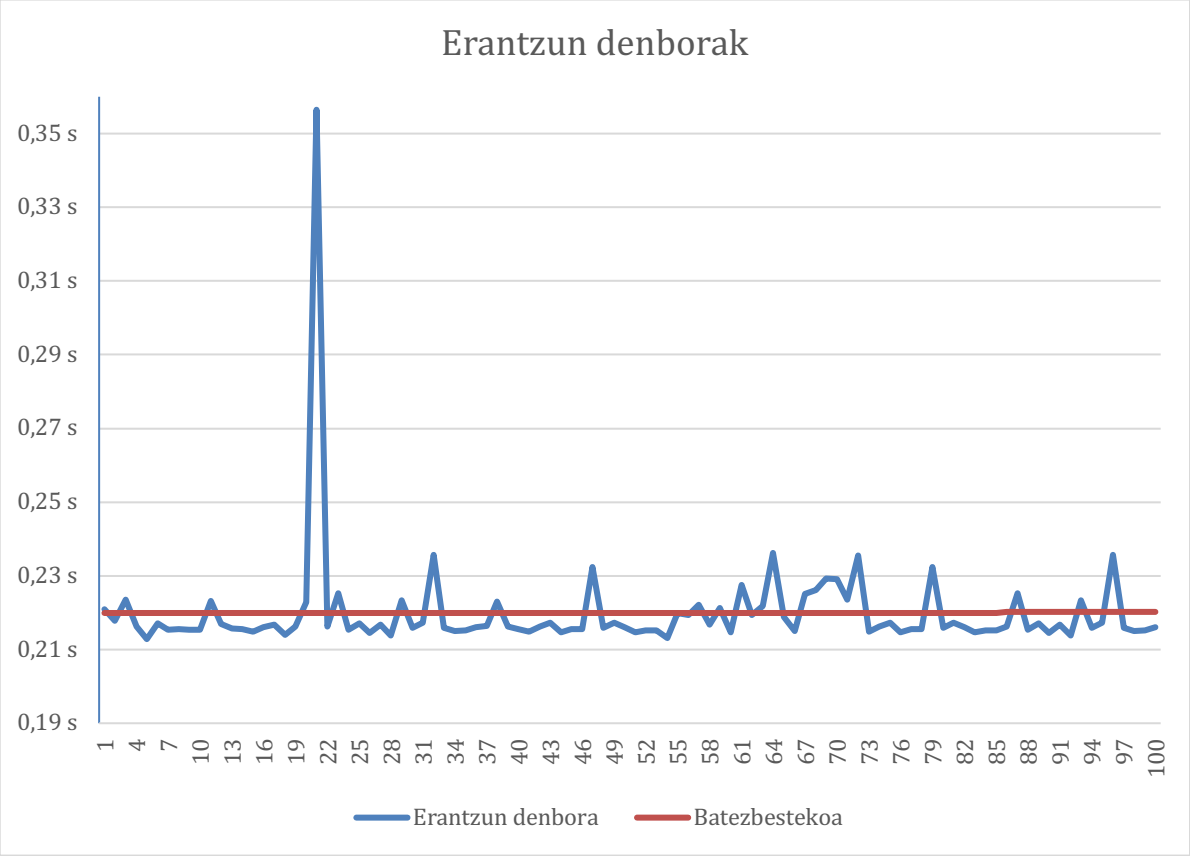
Balio maximoari dagokionez, aipatu behar da gainontzeko balioengandik oso urrun dagoela, denbora lar altua baita. Ondorioz, kasu konkretu hori kasu berezitzat har daiteke, momentuan eman daitekeen latentzia baten ondorioz, adibidez. Lortutako datu estatistikoen laburpena 17. taulan adierazten da.

| Maximoa | Minimoa | Batezbestekoa | Desbideratze tipikoa | Konfiantza tartea |
|---|---|---|---|---|
| 0,302363 s | 0,203544 s | 0,217204 s | 0,009574455 s | 0,215267 s – 0,219140 s |

*Taula 17: bigarren adibidearen datu estatistikoak*

### 9.2.3. Hirugarren adibidea

Adibide honetan *policy3.xml* politika erabiliko da, hau da, 10 byte-eko luzera duen politika. Aurreko adibidean egin den moduan, prozesu osoa 100 aldiz egin da, eta lortutako emaitzak hurrengo grafikan ikus daitezke:




*Irudia 28: hirugarren adibideko erantzun-denborak*

Hirugarren adibidearekin lortutako emaitzak 27. irudian daude adierazita. Bertan ikus daitekeen legez, erantzun-denbora guztiak tarte onargarrian daude, segundo bat baino askoz ere txikiagoak

baitira. Izan ere, batezbesteko erantzun-denbora 0,21991452 s-koa da. Gainera, proba hauetan eman den balio maximoa 0,356472 s-koa da eta minimoa 0,212828 s-koa.

Bigarren adibidearekin gertatzen zen moduan, kasu honetan ere maximo oso altu bat lortzen da gainontzeko balioekin konparatuz. Ondorioz, kasu berezitzat har daiteke balio hori, momentuko latentziaren ondorioz. Lortutako datu estatistikoen laburpena 18. taulan adierazten da.

| Maximoa | Minimoa | Batezbestekoa | Desbideratze tipikoa | Konfiantza tartea |
|---|---|---|---|---|
| 0,356472 s | 0,212828 s | 0,21991452 s | 0,014824711 s | 0,215753174 s – 0,224075866 s |

*Taula 18: hirugarren adibidearen datu estatistikoak*

# 10. Plangintza

Proiektu honen garapena hainbat fasetan banatu da. Beraz, atal honetan proiektuan zehar burutu diren fase desberdinak deskribatuko dira. Fase bakoitzari lan-pakete (LP) deritzo, eta fase bakoitzean hainbat ataza (A) burutu behar dira. Fase desberdinak azaldu baino lehen, proiektua osatzen duen lan-taldearen eta behar diren baliabide materialen deskribapena ere emango da. Bukatzeko, proiektuaren plangintza osoaren Gantt diagrama batean irudikatuko da.

## 10.1. Lan-taldea eta baliabide materialak

Proiektu hau osatzen duen lan-taldea hurrengo taulako pertsonek osatzen dute:

| Kodea | Izena | Erantzukizuna | Rola |
|---|---|---|---|
| I1 | Jasone Astorga | Senior ingeniaria | Proiektuaren gainbegiratu eta zuzendu |
| I2 | Maider Huarte | Senior ingeniaria | Proiektuaren gainbegiratu eta zuzendu |
| I3 | Ane Sanz | Junior ingeniaria | Proiektua burutu |

*Taula 19: proiektuaren lan-taldea*

Horrez gain, proiektu hau gauzatzeko behar diren baliabide materialak hurrengoak dira:

| Kodea | Materiala | Kopurua |
|---|---|---|
| PC | Ordenagailua | 1 |
| TB | TelosB CDSa | 1 |
| IR | Iris CDSa | 2 |

*Taula 20: proiekturako baliabide materialak*

## 10.2. Faseen deskribapena

### LP1 - Proiektuaren kudeaketa eta dokumentazioaren garapena

Fase hau proiektu osoan zehar ematen da, eta bertan proiektua kudeatzeko lanak eta beharrezko dokumentazioaren garapena egingo da. Lan-pakete honetan faseen kudeaketa eta plangintza ondo jarraitzen dela ere bermatuko da. Horretarako, batzar desberdinak egingo dira proiektuaren garapenaren jarraipen egokia egiteko. Fase honetan desberdintzen diren atazak hurrengoak dira:

- **A101 – Proiektuaren garapenaren jarraipena.** Ataza honetan proiektu osoaren jarraipena egingo da. Proiektu osoan zehar hainbat batzar egingo dira lanaren egoeraren berri emateko, sortzen doazen arazoak zuzentzeko eta egin beharreko lana bideratzeko.
  ***Iraupena:*** *100 ordu proiektu osoan zehar.*

- **A102 – Dokumentazioaren garapena.** Ataza honetan proiektuan zehar egiten diren zeregin guztiak dokumentatuko dira. Horrez gain, proiektuaren dokumentu finala ere idatziko da. Ataza hau proiektu osoan zehar gauzatuko da, gainontzeko faseekin batera.
  ***Iraupena:*** *100 ordu proiektu osoan zehar.*

Fase honetarako behar diren giza-baliabideak, baliabide materialak eta bakoitzaren beharrezko lan-orduak hurrengoak dira:

- **I1, I2:** 40 ordu bakoitzak.
- **I3:** 200 ordu.
- **Ordenagailua:** 200 ordu.

### LP2 - Proiektuaren definizioa

Fase hau proiektuaren lehen fasea da. Bertan proiektuari hasiera emango zaio, lana definituz, bere helburuak ezarriz eta proiektuan zehar egin beharrekoa adostuz. Fase hau oso garrantzitsua da, ondoren egin beharreko lan guztia hemendik abiatzen baita. Beraz, beharrezkoa da fase honetako atazak ondo burutzea. 4 ataza desberdin daude lan-pakete honetan:

- **A201 – Proiektuaren helburuen eta espezifikazioen definizioa.** Ataza honetan proiektuan zehar egin behar dena definituko da. Proiektua bukatzen denerako bete behar diren helburuak ezarri behar dira, eta proiektuak dituen baldintza eta espezifikazioak ere definitu behar dira. Ataza hau zehaztasunez gauzatzea garrantzitsua da ideia guztiak finkatzeko eta lana ondo burutu ahal izateko.
  ***Iraupena:*** *10 ordu.*

- **A202 – Proiektua garatzeko beharrezkoak diren kontzeptuen ikasketa.** Ataza honetan lanarekin hasi aurretik proiektuaren oinarria diren kontzeptuak ikasi behar dira. Izan ere, proiektu honetan Hidra protokoloaren eskuragarritasuna hobetzeko mekanismoak garatuko dira. Ondorioz, Hidraren eta beharrezkoak diren beste kontzeptu batzuen funtzionamendua ulertu behar da: RPL, IEEE 802.15.4, 6LoWPAN…
  ***Iraupena:*** *60 ordu.*

- **A203 – Alternatiben analisia.** Ataza honetan proiektua garatzeko dauden alternatiba desberdinak aztertuko dira, kasu bakoitzean aukerarik onena erabiltzeko. 5 analisi desberdin gauzatuko dira: ebaluaketa burutzeko mekanismoa, CDS plataforma, CDSetako sistema eragilea, interfaze grafikoa garatzeko teknologia eta politikak biltegiratzeko modua.
  ***Iraupena:*** *40 ordu.*

- **A204 – Arriskuen analisia.** Ataza honetan proiektuan eman daitezkeen arriskuen analisi bat egingo da, arrisku bakoitzari probabilitate bat eta inpaktu bat esleituz. Ondoren, arriskuak saihesteko edota arriskuak gertatzekotan arazoa konpontzeko moduak eman behar dira.
  ***Iraupena:*** *10 ordu.*

Fase honetarako behar diren giza-baliabideak, baliabide materialak eta bakoitzaren beharrezko lan-orduak hurrengoak dira:

- **I1, I2:** 5 ordu bakoitzak.
- **I3:** 120 ordu.
- **Ordenagailua:** 100 ordu.

### LP3 - Garatu beharreko moduluen diseinua

Behin proiektua definituta dagoenean eta egin beharrekoa ezarri denean, garatu behar diren moduluak diseinatu behar dira. Fase honetan 2 ataza desberdin daude:

- **A301 – Interfaze grafikoaren diseinua.** Ataza honetan politikak sortzeko interfaze grafikoa diseinatu behar da. Interfazearen diseinua egiterakoan, beharrezkoa izango da datu-basearen diseinua ere egitea, biek batera lan egiten baitute.
  ***Iraupena:*** *40 ordu*

- **A302 – Kodifikatzailearen eta dekodifikatzailearen diseinua.** Ataza honetan kodifikatzailea eta dekodifikatzailea diseinatu behar dira. Modulu bien diseinua ataza berdinean jarri da bien funtzionamendua antzekoa delako: dekodifikatzaileak kodifikatzailearen alderantzizko prozesua egiten du.
  ***Iraupena:*** *40 ordu.*

Fase honetarako behar diren giza-baliabideak, baliabide materialak eta bakoitzaren beharrezko lan-orduak hurrengoak dira:

- **I1, I2:** 5 ordu bakoitzak.
- **I3:** 80 ordu.
- **Ordenagailua:** 80 ordu.

### LP4 - Garatu beharreko moduluen garapena

Garatu behar diren 3 moduluak ondo diseinatuta daudenean, haien garapena egin behar da. Horretarako, alternatiben analisian erabakitako programazio lengoaiak erabili behar dira, eta oso garrantzitsua da diseinuan ezarritako pausoak jarraitzea garapenean. Fase honetan 2 ataza daude:

- **A401 – Interfaze grafikoaren garapena.** Ataza honetan politikak sortzeko interfaze grafikoa garatu behar da. Interfaze grafikoa garatzerakoan, beharrezkoa izango da datu-base egokia sortzea eta interfazearekin sinkronizatzea, bertan sortutako politikak datu-basean biltegiratu daitezen. Gainera, XML fitxategiak sortzeko funtzioak ere garatu beharko dira interfaze grafikoaren barruan.
  ***Iraupena:*** *80 ordu.*

- **A402 – Kodifikatzaile eta dekodifikatzailearen garapena.** Ataza honetan kodifikatzailea eta dekodifikatzailea garatu behar dira. Lehenengo eta behin kodifikatzailea garatuko da, XML fitxategietatik politikak irakurtzeko eta dagokion adierazpen bitarrera kodifikatzeko. Ondoren, dekodifikatzailea garatuko da, kodifikatzailearen alderantzizko prozesua egoteko.
  ***Iraupena:*** *80 ordu.*

Fase honetarako behar diren giza-baliabideak, baliabide materialak eta bakoitzaren beharrezko lan-orduak hurrengoak dira:

- **I1, I2:** 10 ordu bakoitzak.
- **I3:** 160 ordu.
- **Ordenagailua:** 160 ordu.

### LP5 - Maketaren diseinua eta inplementazioa

Behin lan honetako 3 moduluak garatuta daudenean, hurrengo fasea ebaluazio-maketa diseinatzea eta inplementatzea izango da. Izan ere, beharrezkoa da pauso hau egitea garatutako moduluen funtzionamendu egokia frogatu aurretik, balidazio hori maketan egiten delako. Beraz, 3 ataza daude fase honetan:

- **A501 – Ebaluazio-maketaren diseinua.** Ataza honetan ebaluazio-maketa diseinatuko da, erabiliko den CDS kopurua, CDSen kokapena eta bakoitzean kargatu beharreko programa ezarriz.
  ***Iraupena:*** *8 ordu.*

- **A502 – Ebaluazio-maketaren inplementazioa.** Ataza honetan A501 atazan diseinatutako maketa inplementatuko da. Horretarako, CDSetan beharrezko programak kargatuko dira, eta dagozkien lekuan kokatuko dira, maketa martxan jarriz.
  ***Iraupena:*** *8 ordu.*

- **A503 – Funtzionamenduaren eta konexioaren frogak.** Ataza honetan maketaren funtzionamendu egokia bermatzeko frogak egingo dira. Horretarako, *border-routerra* den CDSak sarean dauden CDSak detektatzen dituela frogatu behar da, baita horietara heltzeko konektibitatea duela ere.
  ***Iraupena:*** *16 ordu.*

Fase honetarako behar diren giza-baliabideak, baliabide materialak eta bakoitzaren beharrezko lan-orduak hurrengoak dira:

- **I1, I2:** 4 ordu bakoitzak.
- **I3:** 24 ordu.
- **Ordenagailua:** 24 ordu.
- **TelosB eta Iris CDSak:** 24 ordu bakoitzak.

### LP6 - Garatutako moduluen balidazio funtzionala

Behin ebaluazio-maketa martxan dagoenean, LP4 fasean garatutako moduluen balidazio funtzionala egin behar da maketaren gainean. 2 ataza desberdin daude fase honetan:

- **A601 – Interfaze grafikoaren balidazioa.** Ataza honetan interfaze grafikoaren funtzionamendua egokia dela bermatu behar da proba desberdinak eginez. Horretarako, zerbitzarian exekutatuko da interfazea eta baimena duen erabiltzaile batek politika desberdinak sortuko ditu, bakoitza era egokian sortu eta biltegiratzen dela bermatuz.
  ***Iraupena:*** *30 ordu.*

- **A602 – Kodifikatzaile eta dekodifikatzailearen balidazioa**. Ataza honetan kodifikatzaile eta dekodifikatzailearen funtzionamendu egokia bermatu behar da. Horretarako, maketa osoa martxan jarriko da eta bezero baten eskaera-prozesua gauzatuko da, proba bakoitzean politika desberdin bat erabiliz. Proba bakoitzean politikaren kodifikazioa/dekodifikazioa egokia izan dela egiaztatu behar da.
  ***Iraupena:*** *40 ordu.*

Fase honetarako behar diren giza-baliabideak, baliabide materialak eta bakoitzaren beharrezko lan-orduak hurrengoak dira:

- **I1, I2:** 4 ordu bakoitzak.
- **I3:** 40 ordu.
- **Ordenagailua:** 40 ordu.
- **TelosB eta Iris CDSak:** 40 ordu bakoitzak.

### LP7 - Sistemaren errendimendu frogak

Proiektu honen azken fasea sistemaren errendimendu frogak gauzatzea da. Izan ere, lan honetan garatutako moduluek sistema osoaren errendimenduan eragin negatiborik ez dutela egiten bermatu behar da. Fase honetan ataza bakarra dago:

- **A701 – Erantzun denboraren neurketa.** Ataza honetan sistemak duen erantzun-denbora neurtu behar da, denbora < 1s dela bermatzeko. Horretarako, bezero baten eskaera-prozesua martxan jarri behar da CDSaren erantzuna jasotzen duen arte pasatutako denbora neurtzeko. Probak politika desberdinekin egin behar dira, kodifikazio/dekodifikazio denbora desberdina izango delako politikaren luzeraren arabera. Gainera, emaitza fidagarriak lortzeko beharrezkoa da politika bakoitzeko 100 proba inguru egitea.
  ***Iraupena:*** *20 ordu.*

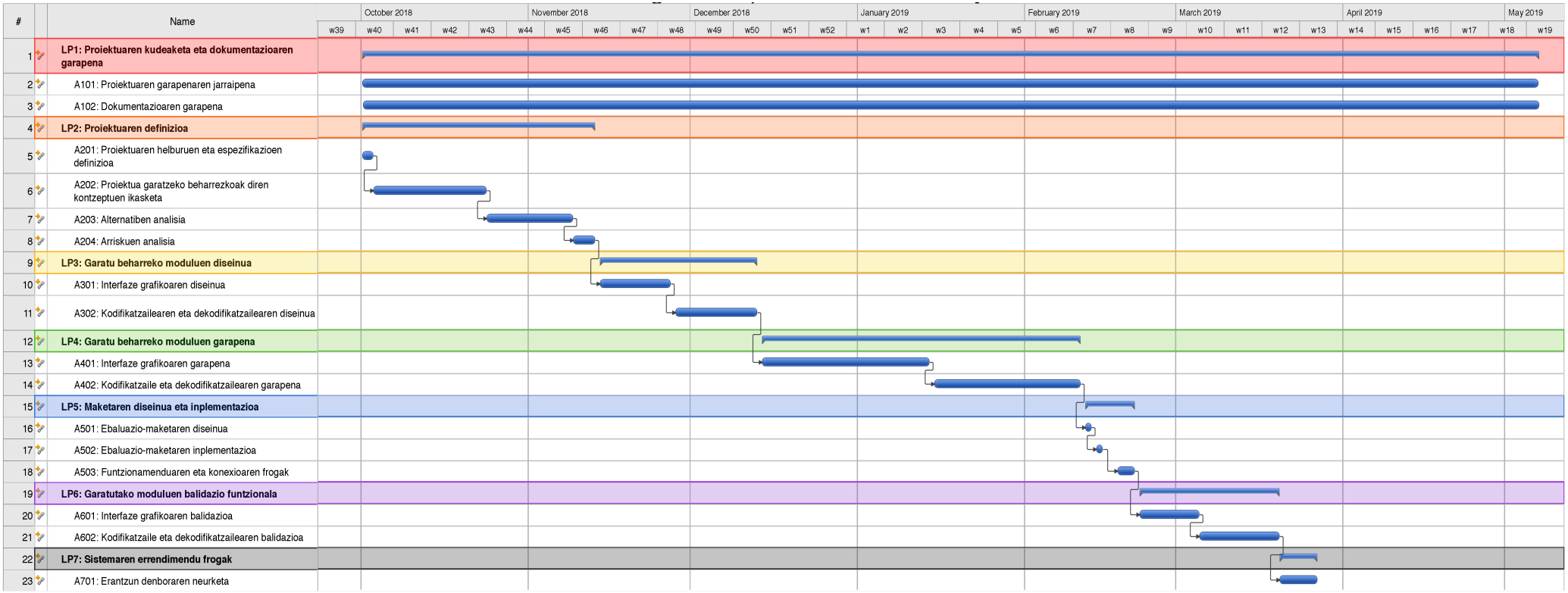
Fase honetarako behar diren giza-baliabideak, baliabide materialak eta bakoitzaren beharrezko lan-orduak hurrengoak dira:

- **I1, I2:** 4 ordu bakoitzak.
- **I3:** 20 ordu.
- **Ordenagailua:** 20 ordu.

- **TelosB eta Iris CDSak:** 20 ordu bakoitzak.

## 10.3. Gantt diagrama

Atal honetan aurretik deskribatutako plangintzaren Gantt diagrama adierazten da. Proiektuaren hasiera-data 2018/10/01-ean dago finkatuta, eta 29. irudian ikus daitekeen legez, proiektuaren bukaera 2019/05/07-rako izatea aurreikusi da.




*Irudia 29: Gantt diagrama*

# 11. Aurrekontua

Atal honetan proiektuaren alderdi ekonomikoak adieraziko dira, proiektuaren aurrekontu bat gauzatuz. Izan ere, proiektu hau gauzatzeko alde batetik giza baliabideak behar dira, bi ingeniari senior eta ingeniari junior batekin. Beste alde batetik, baliabide materialak ere behar dira, hau da, ordenagailuak eta CDS plataformak.

Ondorioz, baliabide horien aurrekontua gauzatzeko kalkulu desberdinak egin behar dira. Lehenengo eta behin, giza baliabideen kostua kalkulatzeko, haien barne-orduen kalkulua egin behar da, bakoitzaren orduko tasa eta lanordu kopurua erabiliz. 21. taulan barne-orduen kalkulua ikus daiteke.

| KODEA | IZENA | ORDUKO TASA | ORDU KOPURUA | GUZTIRA |
|---|---|---|---|---|
| I1 | Senior ingeniaria | 60 €/h | 72 h | 4.320 € |
| I2 | Senior ingeniaria | 60 €/h | 72 h | 4.320 € |
| I3 | Junior ingeniaria | 25 €/h | 624 h | 15.600 € |
| | | | **Guztira:** | **24.240 €** |

*Taula 21: barne-orduen kalkulua*

Ondoren, baliabide materialen kostua kalkulatzeko, haien amortizazioa kalkulatu behar da. Kalkulu hori egiteko, baliabide bakoitzaren kostuaz gain, bizitza erabilgarria eta erabiliko diren ordu kopurua behar dira. 22. taulan amortizazioen kalkulua adierazten da:

| KODEA | IZENA | KOSTUA | BIZITZA ERABILGARRIA | ORDU KOPURUA | GUZTIRA |
|---|---|---|---|---|---|
| PC | Ordenagailua | 900 € | 10000 h | 624 h | 56,16 € |
| IR1 | Iris sentsorea | 140 € | 8000 h | 64 h | 1,12 € |
| IR2 | Iris sentsorea | 140 € | 8000 h | 64 h | 1,12 € |
| TB | TelosB sentsorea | 90 € | 6000 h | 64 h | 0,96 € |
| | | | | **Guztira:** | **59,36 €** |

*Taula 22: amortizazioen kalkulua*

Proiektu honetarako ez da azpikontrataziorik ezta gastu finkorik aurreikusi. OndoriozP, proiektu osoaren aurrekontua kalkulatzeko, barne orduen eta amortizazioen guztizkoak gehitu behar dira. 23. taulan proiektu osoaren aurrekontua 44.459 €-koa dela ikus daiteke.

| KONTZEPTUA | GUZTIRA |
|---|---|
| Barne-orduak | 24.240 € |
| Amortizazioak | 59,36 € |
| **Guztira:** | **24.299 €** |

*Taula 23: aurrekontuaren laburpena*

# 12. Ondorioak

Proiektu honen helburu nagusia Hidra segurtasun protokoloaren erabilgarritasuna hobetzea izan da, horretarako hainbat mekanismo diseinatuz, garatuz eta inplementatuz. Horretarako, alde batetik erabiltzaile-interfaze grafiko bat diseinatu da segurtasun politika berriak era erraz batean sortu ahal izateko. Interfaze grafiko honi esker segurtasun-mekanismoen erabilera erraztu da segurtasunean ez adituak diren pertsonentzat. Ondorioz, mekanismo honen garapenak proiektuaren helburu nagusia betetzen laguntzen duela demostratu da.

Beste alde batetik, kodifikatzaile eta dekodifikatzaile moduluak ere garatu dira, politikak XML formatutik adierazpen bitarrera itzultzeko eta alderantziz. Honek ere Hidraren erabilgarritasunean hobekuntza nabarmena suposatzen du. Izan ere, zerbitzaritik CDSra bidaltzen den politika autorizazio lokala gauzatzeko konpresio maila altuko adierazpen bitarrean bidaltzen da. Era horretan, CDSen baliabideak (memoria, prozesatzeko gaitasuna…) ahalik eta era optimoenean aprobetxatzen dira, eta hau guztiz beharrezkoa da IoT inguruneetako aplikazioetan.

Gainera, garatutako moduluek sistemaren errendimenduan ez dute inpaktu negatiborik suposatzen, errendimendu frogetan erantzun-denborak tarte onargarrian mantentzen direla demostratu baita. Hori dela eta, sistema osoaren inplementazioa bideragarritzat har daiteke, kalitatezko zerbitzu bat lor daitekeelako segurtasun sendoa eta erabilgarria mantentzen den bitartean.

Ondorioz, proiektu honetan ezarritako helburu guztiak, bai helburu nagusia eta bai bigarren mailako helburuak, bete direla ikusi da.

Horrez gain, aipatu beharra dago proiektu honetan garatu diren mekanismoak Hidra protokoloarentzat sortu direla zehazki. Hala ere, IoT inguruneetako beste protokolo edota aplikazioetara hedatzeko aukera egongo litzateke sortutako mekanismoetan behar diren egokitzapenak gauzatuz. Ondorioz, proiektu honetan garatutako mekanismoak IoT aplikazioen inplementazioa eta hedapena emateko lagungarriak izan daitezke. Era horretan, teknikoki eta sozialki onuragarriak diren zerbitzu berri asko inplementatzea posible izango da.

Bukatzeko, proiektu honi buruzko artikulu zientifiko bat idatzi da [13] URSI kongresura aurkezteko, eta artikulua onartua izan da. Hori dela eta, proiektuari buruzko aurkezpen bat egingo da irailean ospatuko den URSI kongresuan, Sevillan.

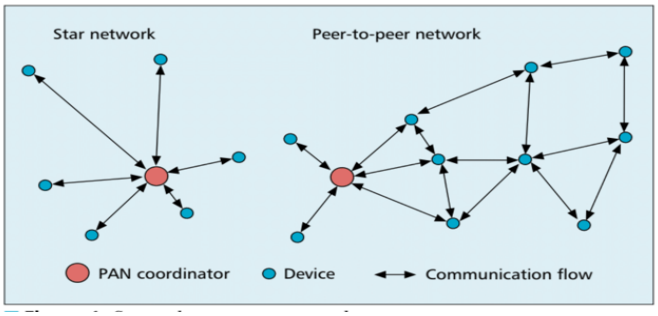
# Eranskinak

## I. eranskina: IEEE 802.15.4

IEEE 802.15.4 potentzia baxuko sareetako (LR-WPAN, *Low Rate Wireless Personal Area Network*) maila fisikoa eta lotura mailako MAC (*Media Access Control*) azpimaila zehazten dituen estandarra da. Izan ere, kostu baxuko nodo asko behar dituzten aplikazioetarako diseinatuta dago. Protokolo honek beharrizan bereziak ditu, adibidez, gailuetan bateria kontsumo txikia egotea, trama txikiak transmititzea, banda-zabalera txikia erabiltzea, potentzia txikia kontsumitzea, gailuetako memoria gutxi erabiltzea… Beraz, estandar honen helburua abiadura eta potentzia gutxiko sareetan zerbitzua ematea da, beharrizan berezi horietara egokituz.

Estandar hau erabiltzen duten sareetan bi nodo mota bereizten dira. Lehen mota FFD (*Full Function Device*) da, eta hauek PAN (*Personal Area Network*) koordinatzaile moduan edo nodo normal moduan lan egiteko gaitasuna dute. Gainera, edozein gailurekin komunikatu daitezke, eta mezuak bideratzeko gaitasuna dute. Bigarren gailu mota RFD (*Reduced Function Device*) da, eta hauek nodo normalak dira, ez dituzte inoiz koordinatzaile funtzioak betetzen eta FFDekin bakarrik komunikatzen dira.

Estandar honek sareko konexioa egiteko hiru topologia mota onartzen ditu: puntu-puntuko topologia, izar topologia, eta *mesh* topologia 30. irudian ikus daitekeen moduan. Aplikazioaren arabera, sarearen topologia era batekoa edo bestekoa izango da. Hala ere, edozein topologia izanda ere, beharrezkoa da sare bakoitzean gutxienez FFD batek koordinatzaile moduan lan egitea.




*Irudia 30: IEEE 802.15.4 protokoloaren topologiak*

Alde batetik, MAC azpimailari dagokionez, hainbat funtzio betetzen dira: asoziazioa, disoziazioa, tramen entregatze-antzematea, tramen onarpena, denbora tarteen kudeaketa egokia… Horrez gain, MAC tramaren egitura hurrengo irudian ikus daiteke:

| *Bytes* | *2* | *1* | *0-20* | *Variable* | *2* |
|---|---|---|---|---|---|
| Frame control | Sequence number | Address info | | Payload | Frame check sequence |
| MAC header (MHR) | | | | MAC service data unit (MSDU) | MAC footer (MFR) |
| MAC protocol data unit (MPDU) | | | | | |

*Irudia 31: MAC tramaren egitura*

Trama osoaren tamaina ezin daiteke 127 byte baino handiagoa izan. Hori kontuan izanik, 4 MAC trama mota bereizten dira: gida trama, datu trama, hartu-agiri trama eta MAC komando trama. Gida tramek eta datu tramek goiko geruzetako informazioa garraiatzen dute, baina beste biak MAC geruzan sortzen dira, protokoloaren funtzionamendu egokia ahalbidetzeko.

Beste alde batetik, maila fisikoari dagokionez, estandar honek aukera desberdin bi eskaintzen ditu. Lehen aukerarekin transmisioa 868/915 MHz-etako bandan gauzatzen da. Gainera, aukera honek jasaten dituen abiadurak 20 kb/s-koak dira 868-ko bandarako, eta 40 kb/s-koak 915-ko bandarako.

Bigarren aukerak, ordea, 2,4 GHz-eko ISM (*Industrial, Scientific and Medical*) bandan transmititzen du, eta 250 kb/s-ko abiadurak onartzen ditu.

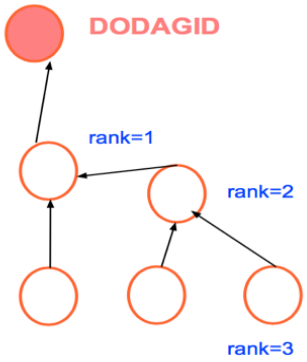
Horrez gain, hiru banden artean 27 kanal definitzen dira. Hasteko, 868 MHz-ko bandan kanal bakarra existitzen da. Ondoren, 915 MHz-eko bandan 10 kanal desberdin daude, 902-928 MHz-eko tartean. Bukatzeko, 2,4 GHz-eko bandan 16 kanal definituta daude, 2,4-2,4835 GHz-eko tartean. Kanal batetik bestera 5MHz-eko tarte librea uzten da.

Helbideratze moduei dagokienez, IEEE 802.15.4 protokoloak helbideratze mota bi onartzen ditu. Alde batetik, IEEE-ko 64 biteko helbide hedatuak erabiltzea onartzen du. Beste alde batetik, PAN barruan bakarrak diren 16 biteko helbideak ere onartzen ditu.

## II. eranskina: RPL

RPL (*Routing Protocol for Low Power and Lossy Networs*) protokoloak potentzia, memoria eta baliabide mugatuak dituzten gailuez osatutako sareetan bideratzea nola egin behar den definitzen du. RPL sareetako topologia DODAGetan (*Destination Oriented Directed Acyclic Graph*) oinarritzen da, hau da, helburu konkretu baterantz, *root* izenekoa, kokatuta dagoen grafo aziklikoa sortzen da bideraketa gauzatzeko.

Sare batean hainbat DODAG daudenean, eta haien identifikatzailea berdina denean (*RPLInstanceID*), DODAG multzo horrek RPL instantzia bat sortzen du. Horrez gain, DODAG bateko nodo bakoitzak maila (*rank)* bat dauka, nodo horren posizio indibiduala definitzen duena *root* nodoarekiko. Hurrengo irudian DODAG bat ikus daiteke, nodo bakoitzaren rank-a adieraziz.




*Irudia 32: DODAG*

DODAG mota desberdin bi bereizten dira RPL protokoloan. Lehenengo mota *grounded DODAG* da, eta mota honetako grafoek konektibitatea eskaintzen dute behar duten *host*-entzat. Bigarren mota *floating DODAG* da, eta mota honetako grafoek, ordea, barruko nodoen arteko bideak bakarrik eskaintzen dituzte.

DODAG desberdin bi existitzeaz gain, operazio modu desberdin bi ere bereizten dira. Lehenengo operazio modua *storing* modua da, non nodo bakoitzak bideratze taularen kopia bat gordetzen duen. Bigarren moduan, *non-storing* moduan, nodo guztiek trafikoa *root* nodora bidaltzen dute, eta hau da bideratze funtzioak betetzearen arduraduna.

Horrez gain, nodo bat aldi berean instantzia desberdinetakoa izan daiteke. Gainera, posible da instantzia batzuetan *root* moduan lan egitea eta beste batzuetan ez. RPL instantzia baten DODAG bakarra dagoenean, instantzia lokala deitzen zaio. DODAG bat baino gehiago daudenean, ordea, instantzia globala deitzen zaio. Kasu hauetan, DODAGak koordinatuta daude, eta haien instantzia IDa berdina izan behar da LLN osoan.

RPL protokoloaren funtzionamendua nolakoa den ulertu ahal izateko, beharrezkoa da existitzen eta transmititzen diren kontrol mezuak deskribatzea.

## Kontrol mezuak

RPL protokoloan 4 kontrol-mezu desberdin existitzen dira: DIS, DIO, DAO eta DAO-ACK. Mezu bakoitzaren egitura eta edukia desberdina da, eta bakoitzak funtzio bat betetzen du bideratze prozesuaren barruan.

**DODAG Information Solicitation (DIS)**

RPL nodo baten objektu-informazio eskatzeko erabiltzen da DIS mezua. Mezuaren egitura hurrengo irudian ikus daiteke:

| Flags | Reserved | Option(s) |
|---|---|---|
| *Unused* | *Unused* | |

*Irudia 33: DIS mezuaren egitura*

**DODAG Information Object (DIO)**

Mezu honetan garraiatzen den informazioari esker nodoak hainbat gauza egin ditzake: RPL instantzia bat aurkitu, bere konfigurazio parametroak jakin, DODAG-a mantendu... Nodo batek berari buruzko informazioa bidaltzen du. DIO mezu baten egitura hurrengo irudian ikus daiteke:

| RPLInstanceID | | | | Version Number | Rank | |
|---|---|---|---|---|---|---|
| G | 0 | MOP | Prf | DTSN | Flags | Reserved |
| DODAGID | | | | | | |
| Options | | | | | | |

*Irudia 34: DIO mezuaren egitura*

DIO-ak transmititzeko, RPLko nodoek *Trickle Timer*-a erabiltzen dute. Nodo batek bere informazioa transmititzen du, baina konturatzen bada beste transmisio batzuk daudela edota bere transmisioa erredundantea izan daitekeela, ez du transmititzen.

**Destination Advertisement Object (DAO)**

DAO mezua helburu baten informazioa DODAGetik gora hedatzeko erabiltzen da, DODAG bateko parte izateko eskaera bidaltzeko. *Storing* moduan lan egiten bada, mezua *unicast* motakoa da, aukeratutako gurasoari bakarrik bidaliz. *Non-storing* moduan lan egiten bada, ordea, mezua *unicast* motakoa ere da, baina *root* nodoari bidaltzen zaio. DAO mezuaren egitura hurrengo irudian ikus daiteke:

| RPLInstanceID | K | D | Flags | Reserved | DAOSequence |
|---|---|---|---|---|---|
| DODAGID | | | | | |
| Options | | | | | |

*Irudia 35: DAO mezuaren egitura*

Mezuan agertzen den K eremuaren bidez mezuak DAO-ACK erantzun mezua behar duen ala ez adierazten du. D eremuaren bidez, ordea, mezuak DODAGID eremua duen ala ez adierazten du.

**Destination Advertisement Object Acknowledgement (DAO-ACK)**

Mezu hau DAO mezu bati dagokion erantzuna da, *root* nodoak edo DAO guraso batek bidalita.

| RPLInstanceID | D | Reserved | DAOSequence | Status |
|---|---|---|---|---|
| DODAGID | | | | |
| Options | | | | |

*Irudia 36: DAO-ACK mezuaren egitura*

## DODAG bat sortzeko prozesua

DODAG bat nola sortzen den deskribatzeko, beharrezkoa da *Objective Function* (OF) zer den azaltzea. OF metrika eta limitazio batzuen konbinazioa da, egoera bakoitzean helburu batera heltzeko biderik onena erabaki ahal izateko. Hau da, OFak DODAGak sortzeko arauak zeintzuk diren espezifikatzen du. OFak izan ditzakeen arauak hurrengoak izan daitezke: guraso kopuru maximoa finkatzea, biderik onena aukeratzea latentziaren arabera, zifratuta ez dauden loturak ekiditea... Beraz, hau kontuan harturik, DODAG bat sortzeko hurrengo prozedura jarraitzen da:

1. *Root* nodoak DIO bat bidaltzen du grafoari buruzko informazioa hedatuz.
2. Auzokideak diren nodoek mezu hori jasoko eta prozesatuko dute. Horrez gain, posible da beste *root* nodo batzuen DIO mezuak ere jasotzea.
3. Nodoek, jasotako mezuen arabera, erabaki bat hartzen dute grafo horrekin elkartzeko ala ez, hainbat faktoretan oinarrituz: *objective function*, grafoaren ezaugarriak...
4. Nodo bat grafora lotu denean, grafoko *root* nodora joateko bide bat dauka, gainera, *root* nodoa beste nodoaren gurasoa izango da. Horrez gain, nodoak bere *rank*-a zein den kalkulatzen du.
5. Nodo hori konfiguratuta badago router moduan lan egiteko, DIO mezu bat sakabanatzen du bere auzoko nodoetara, grafoaren momentuko informazioarekin.
6. DIO mezua jasotzen duten auzoko nodoek prozesu berdina jarraitzen dute: gurasoa aukeratu, bidea sortu, eta DIO mezu berria bidali auzokoei beharrezkoa izanez gero.
7. Prozesu honen bitartez grafoa eratzen doa *root* nodotik gainontzeko nodoetara, eta prozesua *leaf* motako nodoetan bukatzen da. Prozesua amaitzerakoan, nodo bakoitzak bere gurasora (edo gurasoetara) joateko bideraketa sarrera dauka. Beraz, nodo guztiek daukate *root* nodora joateko aukera. Goranzko ibilbide honi *upward routing* deitzen zaio.

Horrez gain, posible da nodoek era aktibo batean grafo bati buruzko informazioa eskatzea, DIS mezuaren bitartez. DIS mezuaren erantzun moduan nodoek DIO mezu bat jasotzen dute.

Behin grafoa sortuta dagoenean, *upward routing* egiteaz gain, beheranzko ibilbidea ere egiteko aukera dago: *downdard routing.* Kasu honetan, era desberdinean gertatzen da bideraketa, lan-moduaren arabera, hau da, grafoak *storing* moduan edo *non-storing* moduan lan egiten badu, beheranzko ibilbidearen bideraketa era desberdinean egiten da.

***Downdard routing storing* moduan**

Trafikoa grafotik behera bideratu ahal izateko DAO mezuak erabiltzen dira. DAO mezuen bitartez *leaf* nodoetara heltzeko hurbiltasunaren informazioa bidaltzen da. Beraz, nodo bat grafo batera gehitzen denean jarraitu beharreko prozesua hurrengoa da:

1. Nodoak DAO mezua bidaltzen dio bere gurasoari.
2. Gurasoa den nodoak DAO mezua jasotzen duenean, informazioa prozesatzen du eta sarrera bat gehitzen du bideratze-taulara, nodo berri horretara heldu ahal izateko. Gainera, DAO mezua bidaltzen dio bere gurasoari.
3. Prozesu hau errepikatzen da *root* nodora heldu arte eta nodoetara joateko bide osoak existitu arte. Beraz, bidea pausoz pauso eraikitzen da.

*Storing* moduan lan egiten denean, bideratze taulak behar dira nodo guztietan, eta honek nodoetan memoria gehiago erabiltzea suposatzen du.

***Downdard routing non-storing* moduan**

Kasu honetan grafotik behera bideratu ahal izateko jarraitzen den prozesua hurrengoa da:

1. Nodoak DAO mezu bat bidaltzen dio bere gurasoari, eta mezu honek bide osoa jarraitzen du *root* nodora heldu arte.
2. *Root* nodoak nodo horretara joateko bidea eraikitzen du DAO *parent set*-ak erabiliz. Hau da, nodoek gurasoaren informazioa gehitzen dute DAO mezuko "*transit-info*" eremuan eta *root* nodoak informazio hori erabiltzen du bidea sortzeko.

Kasu honetan, nahiz eta nodo bakoitzak ez izan bideratze taularen kopia bat, transmititu behar diren mezuen tamaina handiagoa da, informazio gehiago garraiatu behar delako.

## Begizten kontrola

LLN sareetan transmisio tasak sare tradizionaletan baino baxuagoak dira. Horren ondorioz, sortu daitezkeen begizten eragina ez da hain handia izango, baina hala ere, gomendagarria da begiztak ekiditea.

Hori dela eta, nahiz eta RPLk ez duen bermatzen begiztarik ez dela egongo, begiztak ekiditeko ahaleginak egiten ditu, eta begizten detekziorako mekanismoak erabiltzen ditu. Alde batetik, begiztak ekiditeko arau bi erabiltzen ditu:

1. ***Max_depth rule*** arauaren eraginez, nodo batek ezin du guraso moduan arau hori baino *rank* handiagoa duen nodo bat aukeratu.
2. Nodo bat ezin da "greedy" izan, hau da, ezin da grafotik beherantz joan guraso gehiago eduki ahal izateko.

Beste alde batetik, begiztak detektatzeko erabiltzen den mekanismoetako baten bideraketa goiburuan bit berezi batzuk aktibatzen dira, eta bit horiei esker begiztak detektatzea ahalbidetzen da. Adibidez, nodo batek beherantz bidaltzen badu pakete bat, *down* bita aktibatuko du. Ondoren, nodo batek pakete hori jasotzen duenean, bere bideratze-taulan kontsultatu ondoren gorantza bidali behar duela ikusten badu, begizta bat dagoela ondorioztatzen da, eta paketea baztertzen da.

### Timer-aren kudeaketa

Bideratze protokolo tradizionaletan *keepalive* periodikoak bidaltzen dira bideratze taulak eguneratuta mantendu ahal izateko. Hala ere, LLN sareetan metodo hau ez da baliagarria, baliabide mugatuetan kostua lar handia izango litzatekeelako.

Hori dela eta, RPLk *adaptive timer* bat erabiltzen du, *Trickle Timer* izenekoa. Mekanismo honen bitartez DIO mezuen *multicast* bidaltze-tasa perodikoa kontrolatzen da. Sarea egonkorra bihurtzen doan heinean, denbora-tartea ere handitzen da, eta DIO mezu gutxiago bidaltzen dira. Sarean inkonsistentziaren bat ematen denean, ordea, denbora-tarte txikitzen da DIO-ak maiztasun handiagoarekin bidaltzeko. Beraz, DIO mezuen bidaltze-tasa sarearen egonkortasunaren araberakoa da.

## III. eranskina: 6LoWPAN

IPv6 protokoloa sare mailako IP protokoloaren bertsio bat da, eta IPv4 ordezkatzeko sortu zen, bertsio honek dituen mugak gainditu ahal izateko. Izan ere, IoT aplikazioei esker Internetera konektatu behar diren gailu kopurua oso handia da, eta IPv4 protokoloko helbideratzea 32 bitekoa denez, ez dago helbide nahikorik gailu guztiei bat esleitu ahal izateko. Beraz, IPv4 protokoloa erabiliz ezinezkoa izango litzateke IoT aplikazioetako gailuak era egokian desplegatzea, beraz, IPv6 protokoloa erabiltzen da

Beraz, IPv6 protokoloak 128 biteko helbideak erabiltzen ditu, era honetan sarera konekta daitezkeen gailu kopurua handituz. Horrez gain, IP protokoloaren bertsio honek aldaketa gehiago eskaintzen ditu aurreko bertsioarekin konparatuz, adibidez, MTU (*Maximun Transmission Unit*) minimoa handitzen du 576 byte-etik 1280 byte-etara, fragmentazioa bukaerako nodoetan gauzatzen da, *scoped multicast* gehitzen da…

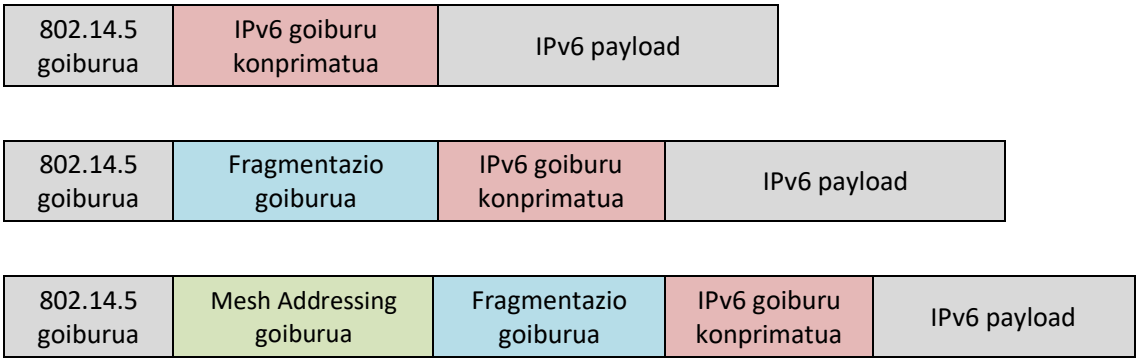
Hala ere, IPv6 eta IEEE 802.15.4 protokoloak ez dira zuzenean bateragarriak. Hori dela eta, komunikazioan bi protokolo horiek erabili ahal izateko, beharrezkoa da egokitzapen protokolo bat erabiltzea, IPv6 IEEE 802.15.4-ren gainean erabiltzea ahalbidetzen duena. Egokitzapen protokolo hau 6LoWPAN (IPv6 over LoWPAN) da.

6LoWPAN protokoloak hainbat funtzio betetzen ditu IPv6 IEEE 802.15.4 protokolora egokitzeko. Adibidez, goiko geruzetako goiburuak konprimatzen ditu, fragmentazioa gauzatzen du, *layer-two forwarding* egiten du…

### Goiburuen konpresioa/enkaptsulazioa [14]

Esan bezala, 6LoWPAN-ek IPv6-ko goiburuak konprimatzen ditu. Horretarako, balio komunen erabileran oinarrituz, goiburuetako eremu batzuk ezabatze ahalbidetzen du.

Horrez gain, protokolo honetan 3 goiburu mota desberdintzen dira*: mesh addressing* goiburua, fragmentazio goiburua eta IPv6-ko goiburu konprimatua. Mota bakoitza identifikatzeko, goiburu bakoitzak *header type* eremu bat dauka. Hiru goiburu mota hauek elkarren artean pilatzen dira, eta horri esker, posible da beharrezkoak ez diren goiburuak kentzea. Adibidez, datagrama txikietan fragmentazioaren goiburua ez da agertzen, edota datagramak irrati-lotura bakarrean garraiatzen direnean *mesh adressing* goiburua ez da agertzen. Beraz, hau kontuan harturik, 6LoWPAN-eko datagramen egiturak hurrengo irudian agertzen direnak izan daitezke:




*Irudia 37: 6LoWPAN datagrama motak*

**Fragmentazio goiburua**

Garraiatu behar den payload-a lar handia denean, IEEE 802.15.4 datagrama bakarrean sartzeko erabiltzen da goiburu hau. Goiburuaren barruan hurrengo eremuak ager daitezke:

- ***Header Type (2 bit):*** Goiburua identifikatzen du fragmentazio goiburu moduan. Fragmentazio goiburuari dagokion identifikatzailea 11 da.
- ***Datagram Size (11 bit):*** Fragmentatu gabeko payload-aren tamaina totala adierazten du. Informazio hau fragmentatutako zati bakoitzean agertzen da.
- ***Datagram Tag (16 bit):*** Payload bati dagozkion zati guztiak identifikatzen ditu.
- ***Datagramm Offset (8 bit):*** Fragmentuaren offset-a identifikatzen du payload-aren barruan.


*Irudia 38: fragmentazio goiburuaren egitura*

Goiburu horiek kontuan harturik, ikus daiteke garraiatzen den lehen fragmentuan Datagram Offset eremua ez dela agertzen, beraz, fragmentu horren datagramaren goiburuaren luzera 4 byte-etakoa izango da. Ondoren, gainontzeko fragmentuetan eremu hori agertu behar denez, goiburuaren luzera 5 byte-etakoa izango da.

| | | | | |
|---|---|---|---|---|
| *1.fragmentua:* | 11000 | Datagram size | Datagram Tag | Payload |
| | *5b* | *11b* | *16b* | |

| | | | | | |
|---|---|---|---|---|---|
| *Beste fragmentuak:* | 11100 | Datagram size | Datagram Tag | Datagram Offset | Payload |
| | *5b* | *11b* | *16b* | *8b* | |

*Irudia 39: fragmentazio goiburuaren edukia*

***Mesh Addressing* goiburua**

Goiburu hau paketeak hainbat saltotan zehar transmititzeko eta *layer-two forwarding* jasateko erabiltzen da. Existitzen diren eremuak hurrengoak dira:

- ***Header Type (2 bit):*** Goiburua identifikatzen du *mesh addressing* goiburu moduan. Goiburu honi dagokion identifikatzailea 10 da.
- ***S eta D bitak (2 bit):*** Jatorriko eta helmugako helbideratze motak identifikatzen ditu.
- ***Hop limit (4 bit):*** Transmisiorako egon daitekeen jauzi kopurua adierazten du. Balioa dekrementatzen da jauzi bakoitzean, eta 0-ra helduz gero datagrama baztertzen da.

- ***Source Address (16-64 bit):*** Jauzi bateko hasierako muturra identifikatzen du, IEEE 802.15.4-ko helbideekin.
- ***Destination Address (16-64 bit):*** Jauzi bateko amaierako muturra identifikatzen du, IEEE 802.15.4-ko helbideekin.


*Irudia 40: Mesh Addressing goiburuaren egitura*

Beraz, *mesh addressing* motako goiburuaren luzera totala 5-17 byte artean egongo da, kasu bakoitzean erabiltzen den helbideratze motaren arabera.

**IPv6 goiburu konprimatua**

IPv6-ko goiburua konprimatzeko, HC1 (*Header Compression 1*) konpresio mota erabiltzen da. 6LoWPAN-ek IPv6-ko goiburuaren eremu batzuk ezabatzen ditu, normalean erabiltzen diren balioetan oinarrituz. Adibidez, ezabatzen diren eremu batzuk hurrengoak dira:

- 64 biteko sarearen aurrizkia kentzen du jatorri eta helmugako helbideetan. Horren ordez, bit bakarra jartzen da bakoitzarentzat, bietan ezaguna den *link-local* aurrizkia jartzeko.
- *Next Header* eremua 2 bitetara konprimatzen du, UDP, TCP edo ICMP den identifikatzeko.
- *Traffic Class* eta *Flow Label* eremuak bit bakarrean adierazten ditu, biak 0 direnean.
- *Payload Length* informazio erredundantea da, beste leku batzuetatik lor daitekeelako (IEEE 802.15.4 datagramatik edo fragmentazio goiburutik), beraz, ezabatu egiten da.
- *Interface Identifier*-ak (IID) IEEE 802.15.4 datagramatik lortu ahal badira, IPv6 goiburutik ken daitezke.
- Bertsioa ezabatzen da, beti delako IPv6.

Beraz, hori kontuan izanik, IPv6-ko goiburuak izan dezakeen egitura hurrengo irudian ikus daiteke.

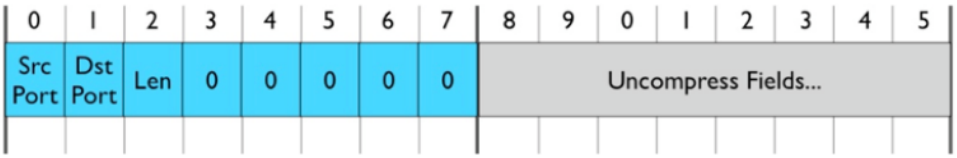

*Irudia 41: IPv6 goiburuaren egitura*

- 0-7 bitetan HC1 konpresioa erabili dela adierazten da
- 8-15 bitetan aurretik azaldutako IPv6 eremuen konpresioa adierazten da.
  - Helbideetan bit bat erabiltzen da IPv6 aurrizkia *link-local* den ala ez adierazteko, eta beste bita IIDa IEEE 802.15.4 datagramatik lor daitekeen ala ez adierazteko.
  - TF eremuak *Traffic Class* eta *Flow Label* eremuak 0 diren eta ezabatu diren ala ez adierazten du.
  - *Next Header* eremuak hurrengo goiburua UDP (01), TCP (11) edo ICMP (10) den identifikatzen du.
  - HC2 eremuak hurrengo goiburua HC2 mekanismoarekin konprimatuta dagoen ala ez adierazten du.
- 16-23 bitetan *Hop Limit* eremua adierazten da, ezin baita konprimatu.
- 24. bitetik aurrera, konprimatu ez diren eremuak adierazten dira.

**UDP eremuen konpresioa**

Aurretik deskribatutako HC1 kodifikazioan HC2 bita aktibatuta dagoenean eta *Next Header* eremuak hurrengo goiburua UDP dela adierazten duenean, HC1 kodifikazioaren ondoren 8 bit gehitzen dira UDP goiburuaren konpresioa nolakoa den adierazteko. Konprimatu daitezkeen UDP-ren eremuak hurrengoak dira:

- ***Source/Destination Port*:** 6LoWPAN-ek portu ezagun batzuk ditu, 61616-61631 tartean. UDP portuak tarte horretan daudenean, lehen 12 bitak ken daitezke, eta jatorri/helmugako portuak 4 bitetan agertzen direla adierazteko bit bat aktibatzen da bakoitzarentzat.
- ***Length:*** IPv6-ko *Payload Length* eremutik lortu ahal bada, eremu hau ezabatzen da eta 0 bita jartzen da hau adierazteko.

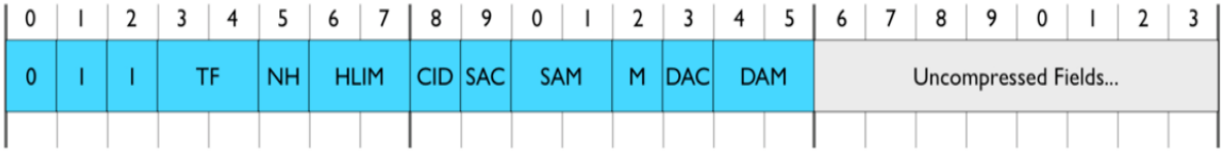



*Irudia 42: UDP eremuen egitura*

Hala ere, deskribatutako konpresio metodoa eraginkorra da komunikazioa *link-local* eremuan gertatzen denean eta ez denean *multicast.* Hori horrela ez den kasuetan ez da konpresio eraginkorra. Hori dela eta, beste konpresio mekanismo desberdin bat erabiltzen da: *improved UDP/IPv6 Header Compression*

***Improved UDP/IPv6 Header Compression***

Konpresio mota honetan HC konpresio mekanismoa erabiltzen da, IPv6 goibrurako IPHC erabiliz. Mekanismo honekin lortzen den IPv6 goiburuaren egitura hurrengo irudian ikus daiteke:




*Irudia 43: Improved UDP mekanismoarekin lortutako goiburua*

- Lehen 3 bitek goiburu mota eta IPHC konpresio mota erabiliko dela adierazten dute.
- TF eremuak *Traffic Class* eta *Flow Label* konprimatu diren ala ez adierazten du.
- NH eremuak hurrengo goiburua UDP (01), TCP (11) edo ICMP (10) den adierazten du.
- HLIM eremuak *Hop Limit* eremuaren balioa 1-255 tartean dagoen edo balio osoa garraiatzen den adierazten du.
- 8-15 bitetan IPv6-ko jatorri/helburu helbideetan erabilitako konpresio metodoak adierazten dira.
    - CID (*Context Identifier*) 0 bada, lehenetsitako testuingurua erabiltzen da helbideak konprimatzeko. Modu hau normalean helbide biak LoWPAN berdinekoak direnean erabiltzen da. CID 1 bada, ordea, 4 biteko eremu bi gehitzen dira, 16 testuinguru posibleetatik helbide bakoitzean zein erabiltzen den adierazteko.
    - SAC (*Source Address Compression*) eremua egoerarik gabeko konpresioa (normalean *link-local* komunikazioetan) edo testuinguruan oinarritutako konpresioa (komunikazio globaletan) erabiltzen den adierazteko erabiltzen da.
    - SAM (*Source Address Mode*) eremuak jatorriko helbidea osorik garraiatzen den, lehen 16 edo 64 bitak kentzen diren edo helbide osoa kentzen den adierazteko erabiltzen da.
    - M (*Multicast*) eremuak helburuko helbidea *unicast* edo *multicast* motakoa den adierazteko balio du. *Unicast* denean, DAC eta DAM eremuak SAC eta SAM eremuen baliokideak dira. *Multicast* denean, DAM eremuak *multicast* konpresio desberdinak adierazten ditu.

Behin IPv6-ko goiburuak metodo honekin nola konprimatzen diren azalduta, UDP goiburuak nola konprimatzen diren azaldu behar da. Hala ere, kasu honetan ez dago aurreko metodoarekin konparatuz aldaketa handirik. Ematen den aldaketarik handiena *Payload Length* eremua ezin dela inoiz garraiatu da, beti lortu ahal delako IPv6-ko goiburutik. Horrez gain, UDPko *checksum* eremua kentzeko aukera ere ematen du, goiko geruzetan funtzio berdina betetzen bada.

Deskribatutako konpresio mota bien arteko desberdintasuna hobeto ulertzeko, bien arteko konparaketa bat ikus daiteke hurrengo taulan, kasu bakoitzean lortzen diren goiburu-luzerak adieraziz. Taula horretan ikus daitekeen legez, *link-local unicast* helbideak erabiltzen ez direnean, askoz ere eraginkorragoa da konpresio modu hobetua.

| | 1.modua | Modu hobetua |
|---|---|---|
| ***Link-local unicast* helbideak** | 7 byte | 6 byte |
| ***Multicast* helbide ezagunak** | 23 byte | 7 byte |
| **Helbide globalak** | 31 byte | 9-10 byte |

*Taula 24: lehen moduaren eta modu hobetuaren arteko konparaketa*

## IV. eranskina: HIDRA

Dokumentuan zehar azaldu den bezala, Hidra sarbide kontrolerako mekanismo bat da, bezeroaren eta CDSaren artean autentifikazioa eta autorizazio dinamikoa bermatzen duena. Protokolo honen eginkizun nagusia sarbide kontrol sendoa eta aldi berean dinamikoa eskaintzea da, CDS oso mugatuetan ere erabilgarria izan daitekeena. Hori dela eta, mutur bien artean E2E (*End to End*) komunikazio segurua bermatuz.

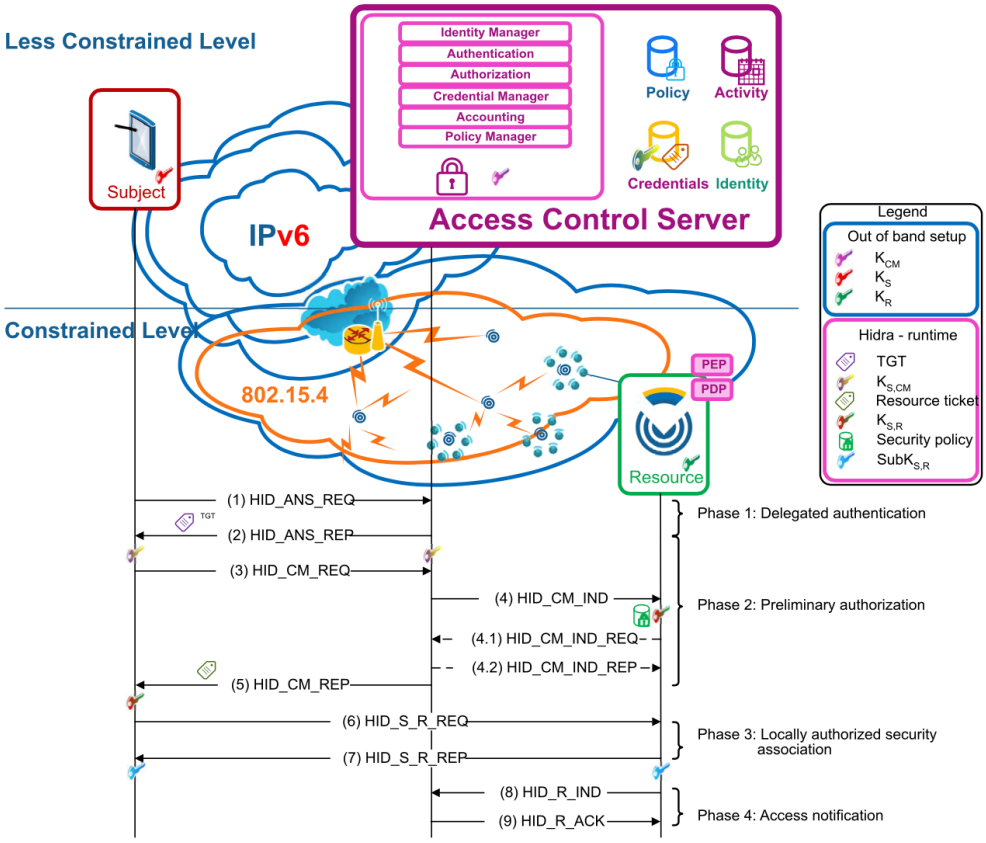
Sarbide kontroleko prozedurari adierazkortasuna eman ahal izateko, protokolo honek arkitektura zentralizatua eta banatua konbinatzen ditu, autorizazioa pauso bitan gauzatuz, hau da, *multi-step* arkitektura jarraituz. Alde batetik, zentralizatutako zerbitzari batek (*Access Control Server*, ACS), hasierako sarbide kontrola gauzatzen du, iragazki baten moduko funtzioak gauzatuz, hau da, baimenik gabeko eskaera gehienak baztertuz. Beste alde batetik, CDS bakoitza banatutako kontrol gune bat da, eta sarbide kontrol dinamikoa gauzatzen da bertan, momentu bakoitzeko testuinguru lokalean oinarrituz.

Beraz, Hidra protokoloaren funtzionamendu zehatza ulertu ahal izateko, beharrezkoa da prozeduran parte hartzen duten elementuak eta haien funtzioak deskribatzea. Hasteko, 44. irudian ikus daitekeen legez, subjektu bat dago, sentsoreko baliabide batera atzipena lortu nahi duena. Ondoren, ACS zerbitzaria ere kontuan hartu behar da eta hasierako sarbide kontrola gauzatzeaz gain, beste hainbat funtzio ere bete behar ditu: kredentzialen banaketa, bezeroen autentifikazioa, *accounting*-a… Bukatzeko, esan bezala, eskaera jasoko duen CDSa dago, eta honek sarbide-kontrol lokala gauzatuko du.

Aurrerago Hidra protokoloan ematen diren pauso guztiak eta elkar trukatzen diren mezuak era zehatzean azalduko dira, baina lehenengo eta behin, Hidraren funtzionamendu orokorra pauso gutxitan azalduko da:

1. ACSak bezeroaren eskaera bat jasotzen du, eta bere autentifikazioa gauzatzen du.
2. Autentifikazioa positiboa bada, subjektuak CDSra atzitzeko ticket bat behar duenez, ACSan hasierako autorizazioa gauzatzen da.
3. Autorizazioa positiboa bada, ACSko CMak (*Credential Manager*) ticketa eta beharrezko gakoak sortzen ditu. Gainera, dagokion politika sentsorera bidaltzen da.
4. CDS-ra eskaera bat heltzen denean, honek ticketa eta aurretik jasotako politikaren baldintza lokalak konprobatzen ditu. Horren oinarrituz, sarbidea onartzeko edo ezeztatzeko erabakia hartzen du.
5. Sarbidea onartu bada, sarbide erabakiaren berri ematen zaio ACSari.

Pauso horietan Hidra protokoloaren funtzionamendu orokorra dago azalduta. Hala ere, funtzionamendu erreala askoz ere zehatzagoa da, eta ulertu ahal izateko, mezu bakoitzaren elkar trukea deskribatzea beharrezkoa da. Hurrengo irudian protokoloaren prozesua zein den adierazten da.




*Irudia 44: Hidra protokoloaren funtizonamendua*

44. irudian ikus daitekeen legez, protokoloak 4 fasetan burutzen ditu segurtasun funtzioak. Lehen fasean autentifikazioa gauzatzen da. Horretarako, bezeroak ACSari 1. mezua bidaltzen dio (*HID_ANS_REQ*), TGT (*Ticket Granting Ticket*) bat eskatuz eta zerbitzariarekiko autentifikatuz. Ondoren, autentifikazioa positiboa izan bada, ACSak TGTa bidaltzen dio 2. mezuaren barruan (*HID_ANS_REP*).

Bigarren fasean hasierako autorizazioa gauzatzen da. Fase hau ACSak egiten du, hau da, arkitektura zentralizatuak, eta honi esker, baimenik gabeko eskaera gehienak CDSra heldu baino lehen baztertzen dira. Hau lortzeko, lehenengo eta behin bezeroak CDSra atzitzeko ticketa eskatzen dio 3. mezuan (*HID_CM_REQ*) ACSko CMari, horretarako aurreko fasean lortutako TGTa bidaliz. CMak beharrezko baimena duela konprobatzen du, eta horrela izatekotan, atzitu nahi den CDSra 4.mezua (*HID_CM_IND*) bidaltzen du, aurrerago jasoko duen eskaera kudeatzeko beharrezkoa den informazioarekin. Informazio honen artean, dagokion segurtasun politika bidaltzen da, ondoren sentsoreak aplikatu dezan. Horrez gain, 4. mezuaren freskotasuna bermatzea beharrezkoa da. Horretarako, norabide bakarreko gako-katea erabiltzen da, beharrezkoak direnean 4.1 (*HID_DM_REQ*) eta 4.2 (*HID_CM_REP*) mezuak bidaliz.

Sentsoreak 4. Mezuan norabide bakarreko gako bat jasotzen du, $K^i$, eta gako hau zuzena dela konprobatu behar du. Lehenengo mezua bada, ez dauka zuzena den konprobatzeko modurik, beraz, CMari hurrengo gakoa bidaltzeko eskatzen dio 4.1 mezuaren bitartez. Ondoren, CMak katearen hurrengo gakoa bidaltzen dio sentsoreari 4.2 mezuarekin, eta sentsoreak gako horri *hash* funtzioa aplikatzerakoan, 4. mezuan jaso duen gakoa lortu beharko luke. Horrela, 4. mezua era positiboan autentifikatuko du.

Hurrengo mezuetan, sentsoreari kateko gako bat heltzen zaionean, sentsoreak aurreko mezuei dagozkien gakoak ditu gordeta, beraz, jarraian *hash* funtzioa aplikatzen du inolako eskaerarik egin gabe, autentifikazioa gauzatzeko. *Hash* funtzioa aplikatzerakoan ez bada aurretik zeukan gakoaren balioa lortzen, tartean mezuren bat galdu delako izan daiteke. Beraz, *hash* funtzioa behin eta berriro aplikatzen du, 5 aldiz gehienez, berak duen gakoa lortzeko. 5 aldiz egin ondoren ez badu emaitza positiborik lortu, 4.1 mezua bidali behar dio CM-ari, katearen hurrengo gakoa eskatuz, eta aurretik deskribatutako prozesua gauzatzen da, gako berria onartu arte.

Bigarren fase honekin bukatzeko, behin gakoa baieztatu denean, CMak bezeroari sentsorera atzitzeko ticketa bidaltzen dio 5. mezuaren bitartez (*HID_CM_REP*).

Hirugarren fasean autorizazio lokala gauzatzen da sentsorean. Fase honetan, bezeroak aurreko fasean lortutako ticketa bidaltzen dio sentsoreari 6. Mezuan (*HID_S_R_REQ*). Ondoren, autorizazio erabakia hartzeko, sentsoreak 4. mezuan jasotako segurtasun politika aplikatzen du. Hartutako erabakia positiboa bada, 7. Mezuarekin (*HID_S_R_REP*) erantzuten dio sentsoreak bezeroari, Security Association-a ezartzen da, eta momentu horretatik aurrera sentsoreko baliabideak erabiltzeko gai izango da bezeroa.

Laugarren eta azken fasean atzipen jakinarazpena gauzatzen da. Fase honetan, sentsoreak ACSra bidaltzen du gertatu berri den sarbideari buruzko informazioa 8. mezuaren barruan (*HID_R_IND*). Ondoren, ACSak baieztapen mezu batekin erantzuten dio, 9. Mezuarekin (*HID_R_ACK*), sentsoreak informazio hori ezabatu dezakeela jakinarazteko. Fase honi esker, zerbitzariak gertatzen denaren informazioa gorde dezake.

Prozesu hori kontuan izanik, hurrengo taulan transmititzen den mezu bakoitzaren edukia eta deskribapena azaltzen dira:

| Zenbakia | Izena | Edukia |
|---|---|---|
| 1 | HID_ANS_REQ | **$ID_S$ , $ID_{CM}$ , $Lifetime_{TGT}$ , $Nonce_1$**<br>*Nonce1: eskaera erantzunarekin erlazionatzeko* |
| 2 | HID_ANS_REP | **$ID_S$ , $Ticket_{CM}$ , {$K_{S,CM}$ , $Nonce_{S,CM}$ , $Nonce_1$ , $ID_{CM}$}$K_S$**<br>$Ticket_{CM}$ = {$K_{S,CM}$ , $ID_S$ , $Nonce_{S,CM}$}$K_{CM}$<br>$Nonce_{S,CM}$: freskotasuna bermatzeko |
| 3 | HID_CM_REQ | **$ID_R$ , $Lifetime_{TR}$ , $Nonce_2$ , $Ticket_{CM}$ , $AuthN_{CM}$** |

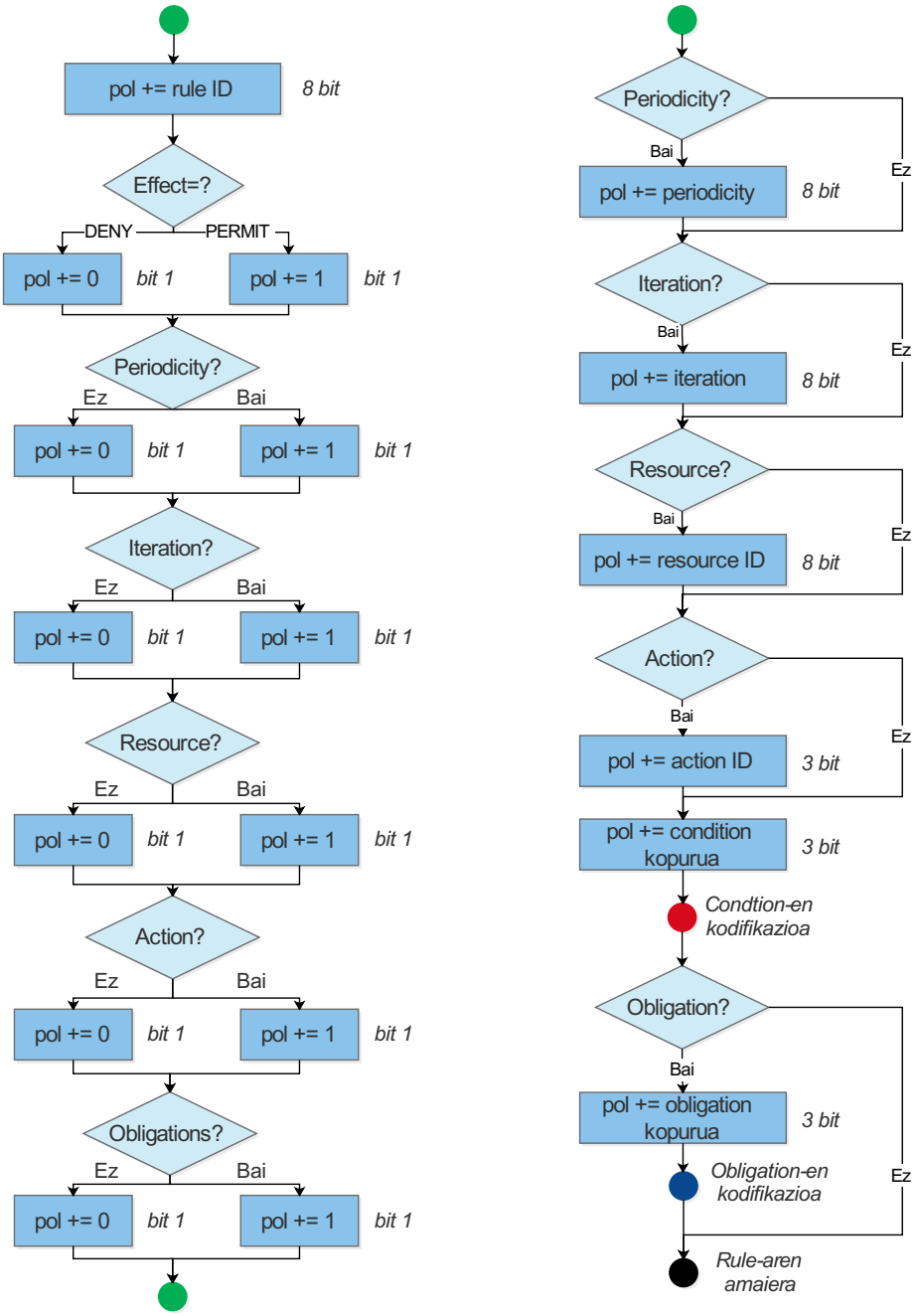
| | | |
|---|---|---|
| | | $AuthN_{CM}$ = {$ID_S$ , $Nonce_{S,CM}$+1}$K_{S,CM}$ |
| 4 | HID_CM_IND | **$ID_R$ , $ID_S$ , $Nonce_{S,R}$ , $Lifetime_{TR}$ , $K^i_{R,CM}$ , $Auth_Z$ , $MAC(K_R: ID_S , Nonce_{S,R} , Lifetime_{TR} , K^i_{R,CM} , Auth_Z)$**<br>$Auth_Z$ = {$Policy_R$}$K_R$ |
| 4.1 | HID_CM_IND_REQ | **$ID_R$ , $Nonce_3$ , $MAC(K_R: ID_R , Nonce_3)$** |
| 4.2 | HID_CM_IND_REP | **$ID_R$ , $K^{i+1}_{R,CM}$ , $MAC(K_R: ID_R , Nonce_3 , K^{i+1}_{R,CM})$** |
| 5 | HID_CM_REP | **$ID_S$ , $Ticket_R$ , {$K_{S,R}$ , $Nonce_{S,R}$ , $Nonce_2$ , $ID_R$}$K_{S,CM}$**<br>$Ticket_R$ = {$K_{S,R}$ , $ID_S$ , $Nonce_{S,R}$ , $Attr_S$ , $Attr_C$}$K_R$ |
| 6 | HID_S_R_REQ | **$Ticket_R$ , $AuthN_R$ , $Nonce_4$**<br>$Ticket_R$ = {$K_{S,R}$ , $ID_S$ , $Nonce_{S,R}$ , $Attr_S$ , $Attr_C$}$K_R$<br>$AuthN_R$ = {$ID_S$ , $Nonce_{S,R}$ , Subkey}$K_{S,R}$ |
| 7 | HID_S_R_REP | **{$Nonce_{S,R}$ , Subkey , $Nonce_4$}$K_{S,R}$** |
| 8 | HID_R_IND | **$ID_R$ , {$Nonce_5$ , $ID_{pol}$ , $Log_{S,R}$}$K_R$**<br>$Log_{S,R}$ = {$ID_S$ , $ID_{RR}$ , $ID_{RA}$ , $ID_{GE}$ , $ID_{RI}$ , $ID_{Ob}$ , $Nonce_{6+i}$} |
| 9 | HID_R_ACK | **$ID_R$ , $Nonce_5$ , $MAC(K_R: ID_R , Nonce_5)$** |

*Irudia 45: Hidra protokoloaren mezuak*

## V. eranskina: kodifikatzailearen fluxu-diagramak

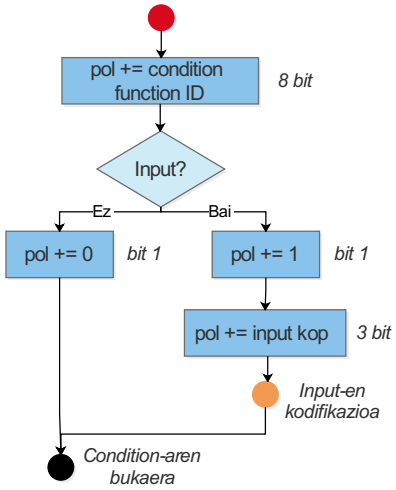
Eranskin honetan kodifikatzaile moduluaren diseinu zehatza deskribatzen da, egitura bakoitza kodifikatzeko jarraitu behar diren pausoak fluxu-diagramen bitartez.

**Rule egituraren kodifikazioa**



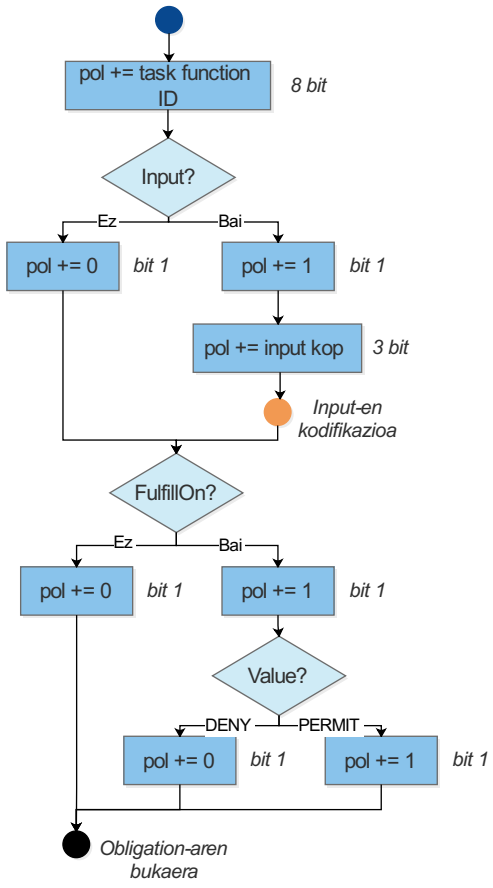

*Irudia 46: rule egituraren kodifikazioaren fluxu-diagrama*

**Condition egituraren kodifikazioa**



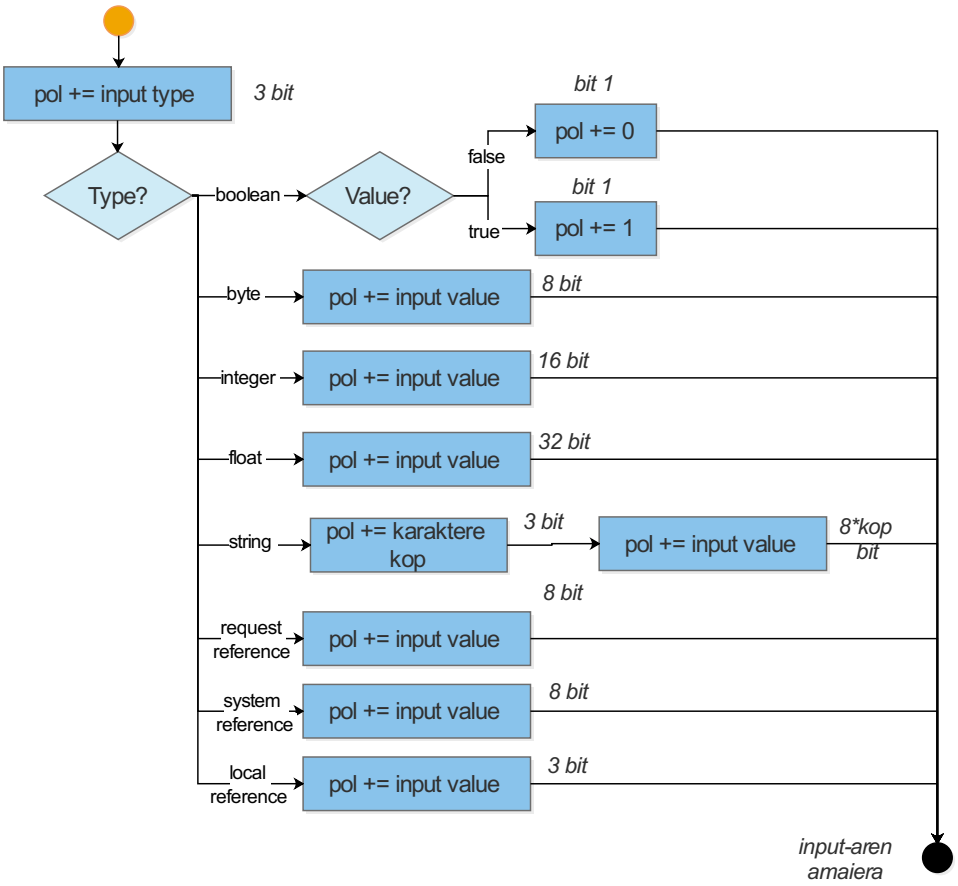

*Irudia 47: condition egituraren kodifikazioaren fluxu-diagrama*

**Obligation eta task egituren kodifikazioa**




*Irudia 48: obligation eta task egituren kodifikazioaren fluxu-diagrama*

**Input egituraren kodifikazioa**




*Irudia 49: input egituraren kodifikazioaren fluxu-diagrama*

## VI. eranskina: balidazio funtzionalaren emaitza zehatzak

Eranskin honetan balidazio funtzionalaren emaitza zehatzak adieraziko dira. Izan ere, memorian emaitza orokorrak bakarrik azaldu dira, laburpen moduan. Hori dela ete, hemen balidaziorako erabili den adibide bakoitzaren emaitzak adieraziko dira. Lehenengo eta behin, interfaze grafikoan politika bakoitza definitu ondoren sortzen diren XML fitxategien edukiak erakutsiko dira. Ondoren, politika bakoitzaren adierazpen bitar zehatza jarriko da.

### *Policy1* adibidea

**Policy1.xml fitxategia**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<policy>
    <id>1</id>
    <effect>PERMIT</effect>
</policy>
```

**Adierazpen bitarra**

```
00000001 10
```

Adierazpen bitarra alde batetik byte-etan banatu da, eta beste alde batetik koloreka adierazi dira, eremu bakoitzaren kodifikazioa adierazteko. Adibidez, kasu honetan lehen 8 bitak (gorriak) politikaren ID-arenak dira, hurrengo bita (beltza) eragina adierazteko eta azken bita (gorria) *rule*-ik ez dagoela adierazteko. Hurrengo adibideetako sekuentziak ere era berean adieraziko dira.

### *Policy2* adibidea

**Policy2.xml fitxategia**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<policy>
   <id>2</id>
   <effect>PERMIT</effect>
   <rule>
      <id>1</id>
      <effect>DENY</effect>
      <condition>
         <function>isTrue</function>
         <input>
            <type>SYSTEM_REFERENCE</type>
            <value>onMaintenance</value>
         </input>
      </condition>
   </rule>
</policy>
```

**Adierazpen bitarra**

00000010 11001000 00001000 00000100 00001110 01110000 00000

### *Policy3* adibidea

**Policy3.xml fitxategia**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<policy>
  <id>3</id>
  <effect>PERMIT</effect>
  <rule>
    <id>2</id>
    <effect>PERMIT</effect>
    <condition>
      <function>lowBattery</function>
      <input>
        <type>INTEGER</type>
        <value>3</value>
      </input>
    </condition>
    <obligation>
      <task>
        <function>activate</function>
      </task>
      <fulfillOn>DENY</fulfillOn>
    </obligation>
  </rule>
</policy>
```

**Adierazpen bitarra**

00000011 11001000 00010100 00100100 00000010 01010000 00000000 00011001 00000000 010

# Bibliografia

[1] K. Lasse Lueth, "State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating," *Iot Analytics,* 2018. Available: https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/.

[2] K. Su, J. Li and H. Fu, "Smart city and the applications," *2011 International Conference on Electronics, Communications and Control (ICECC),* 2011. . DOI: 10.1109/ICECC.2011.6066743.

[3] C. Bormann, M. Ersue and A. Keranen, "Terminology for Constrained-Node Networks-RFC 7228," 2014. Available: http://www.rfc-editor.org/rfc/rfc7228.txt.

[4] L. Seitz, G. Selander and C. Gehrmann, "Authorization framework for the Internet-of-Things," pp. 1-6, 2013. . DOI: 10.1109/WoWMoM.2013.6583465.

[5] B. Parducci and Lockhart Hal, "eXtensible Access Control Markup Language (XACML) Version 3.0," *Oasis,* 2013. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html.

[6] G. Zhang and W. Gong, "The Research of Access Control Based on UCON in the Internet of Things," vol. 6, pp. 724-731, 2011. . DOI: 10.4304/jsw.6.4.724-731.

[7] S. Gusmeroli, S. Piccione and D. Rotondi, "IoT Access Control Issues: A Capability Based Approach," *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing,* pp. 787-792, 2012. . DOI: 10.1109/IMIS.2012.38.

[8] J. Hernández-Ramos *et al*, "DCapBAC: Embedding Authorization logic into Smart Things through ECC optimizations," vol. 93, 2014. . DOI: 10.1080/00207160.2014.915316.

[9] V. Beltran and A. F. Skarmeta, "An overview on delegated authorization for CoAP: Authentication and authorization for Constrained Environments (ACE)," *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT),* pp. 706-710, 2016. . DOI: 10.1109/WF-IoT.2016.7845482.

[10] C. Bormann and P. Hoffman, "Concise Binary Object Representation (CBOR), RFC 7049," 2013. Available: https://www.rfc-editor.org/info/rfc7049. DOI: 10.17487/RFC7049.

[11] J. Astorga *et al*, "Ladon1 : end-to-end authorisation support for resource-deprived environments," *IET Information Security,* vol. 6, *(2),* pp. 93-101, 2012. . DOI: 10.1049/iet-ifs.2010.0259.

[12] M. Uriarte *et al*, "Expressive Policy-Based Access Control for Resource-Constrained Devices," *IEEE Access,* vol. 6, pp. 15-46, 2018. . DOI: 10.1109/ACCESS.2017.2730958.

[13] A. Sanz *et al*, "Advanced IoT cybersecurity for human beings," accepted in *URSI (Simposium Nacional De La Unión Científica Internacional De Radio),* Sept 2019, .

[14] G. Montenegro *et al*, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks, RFC 4944," 2007. Available: https://www.rfc-editor.org/info/rfc4944. DOI: 10.17487/RFC4944.