UNIVERSITEIT GENT

Academiejaar 2013–2014

Faculteit Ingenieurswetenschappen en Architectuur
Valentin Vaerwyckweg 1 – 9000 Gent

# Automatische testbed monitoring voor toekomstig internetonderzoek

**Masterproef voorgedragen tot het behalen van het diploma van**
**Master in de industriële wetenschappen: informatica**

## Andreas DE LILLE

*Promotoren:* *Geert VAN HOOGENBEMT*
*Piet DEMEESTER*

*Begeleiders:* *Brecht VERMEULEN*
*Wim VAN DE MEERSSCHE*
*Thijs WALCARIUS*
*Wim VANDENBERGHE*

# Abstract

Door de huidige verschuiving naar cloudgebaseerde technologiën zal het belang van netwerkprotocollen en van de beschikbaarheid van netwerken alleen maar toenemen. Om deze verschuiving vlot te laten verlopen is er meer en meer onderzoek nodig naar netwerktechnologiën. Om onderzoekers en onderzoekscentra beter te laten samenwerken is FIRE (Future Internet Research and Experimentation) opgestart. Dit project zal trachten om de uitwisseling van ideeën tussen onderzoekers te verhogen. Daarnaast wordt, door de ontwikkeling van een gemeenschappelijke architectuur, het delen van testfaciliteiten makkelijker gemaakt. Deze architectuur draagt de naam SFA (Slice Federation Architecture). Om het leven van onderzoekers makkelijker te maken is jFed ontworpen. jFed wordt gebruikt om testfaciliteiten aan te sturen via SFA. Deze manier van werken zorgt ervoor dat men snel proefopstellingen kan maken op verschillende testfaciliteiten. Toch zijn er enkele nadelen verbonden aan deze manier van werken. Het is voor een onderzoeker soms erg moeilijk om te bepalen of een bepaald gedrag in zijn experiment te wijten is aan eigen ontwikkelingen, of aan het falen van een testbed.

Deze masterproef verhelpt dit probleem door de invoer van een automatisch monitoringsproces. Een monitoringsservice zal instaan voor de monitoring van de testfaciliteiten. De informatie die hierbij verzameld wordt, wordt via een monitoringsAPI ter beschikking gesteld. Deze API (application programming interface) vormt een stevige basis waarop andere toepassingen kunnen gemaakt worden.

# Abstract

Due to the new cloud-based technologies, network protocols and network reachability are now more important than ever. To make this change quick and clean, we need more and more network research. FIRE (Future Internet Research and Experimentation) is an European project created to improve the network and internet experimentation. FIRE is the opportunity to jointly develop potentially disruptive innovations for the future of the internet. It is about collaborative research and sharing test facilities. Researchers working for FIRE will now work closer together, sharing ideas. FIRE is also used to share test facilities from all over the world, so researchers have access to many different test facilities.

To make handling of these different testbeds easier, jFed was created. jFed is a java tool with the purpose of controlling testbeds. jFed uses the SFA (Slice Federation Architecture) to control and configure testbeds. Unfortunately, the current situation has a major downside in that it is very hard for researchers to determine if a certain behavior is caused by the testconfiguration or by the test facility.

This thesis will try to solve the aforementioned problem by creating an automated testbed monitoring system. A monitoringAPI will then share this information. Doing so, will provide future tools easy access to this information.

# Lijst van afkortingen

AM Aggregate Manager

API application programming interface

FED4FIRE Federation 4 FIRE

FFA First Federation Architecture

FIRE Future Internet Research and Experimentation

FLS First Level Support

GENI Global Environment for Network Innovations

MA Management Authority

RSpec Resource Specification

SA Slice Authority

SFA Slice-based Federation Architecture

SFA slice-based federation architecture

SLA Service Level Agreements

# Lijst van figuren

# Inhoudsopgave

# Inleiding

Het gebruik van netwerken en het Internet om computers en allehande randapperatuur te verbinden zal in de toekomst alleen maar stijgen. Het is dan ook van groot belang dat onderzoek op dit gebied vlot en correct verloopt. Daarom is het aangewezen dat onderzoekers samenwerken om zo ideeën en nieuwe technologiën te delen. Daarnaast moeten er ook testfaciliteiten zijn om deze nieuwe technologiën te testen. FIRE (Future Internet Research and Experimentation) is een Europees onderzoeksproject dat zich op deze doelen richt.

Om de configuratie en werking van de verschillende testbeds gelijk te trekken, is de federation architectuur ontworpen. De invoering hiervan zit in het FED4FIRE (Federation for FIRE) project. De federation architectuur die in deze masterproef behandeld wordt, is de SFA 2.0 (Slice-federation-architecture). SFA deelt een testbed op in meerdere niveau's, het onderste niveau zijn de componenten. Een component is een computer, router, ... van een testbed. Een term die hier ook vaak komt bij kijken is een node. Een node is een computer binnen een testbed. Een node is dus een component van een bepaald type.

Omdat testbeds vaak door meerdere onderzoekers tegelijk gebruikt worden, worden componenten vaak gemultiplexed. Zo krijgt elke onderzoeker 'een stuk' van een computer, een sliver. Meerdere slivers vormen samen een slice. Een slice is dus een verzameling componenten waarop een experiment gedraaid wordt. Daarnaast kan men de componenten ook groeperen in aggregaten. Een aggregaat is een verzameling componenten die valt onder eenzelfde beheerder. Een goed voorbeeld van een aggregate is de virtual wall. De virtual wall is een testfaciliteit van iMinds, gebruikt om netwerken te simuleren. De virtual wall is een aggregate, in die zin dat alle componenten beheerd worden door iMinds. SFA wordt later in de scriptie uitgebreid besproken.

Het beheer van al deze verschillende aggregates is geen sinecure. Om dit beheer te vereenvoudigen heeft iMinds binnen het Fed4FIRE project een monitoringsservice gemaakt. Deze service is echter snel ontwikkeld en is niet geschikt voor uitbereidingen. Deze masterproef bestaat uit 2 grote delen. Het eerste deel is een nieuwe monitoringsservice maken die de testbeds controleert. De informatie die deze service verzameld zal beschikbaar gemaakt worden door een monitoringsAPI.

FIRE werkt nog samen met een gelijkaardig project, GENI. GENI (Global Environment for Network Innovations) is een Amerikaans project met gelijkaardige doelstellingen als FIRE. Hierdoor is de samenwerking tussen beide projecten zeer hoog. Het tweede deel van de masterproef is de integratie van monitoringsAPI in het GENI project.

Hoofdstuk 1 Situeert de masterproef. Hier wordt ook kort de opdracht uitgelegd.

Hoofdstuk 2 gaat dieper in op de SFA-architectuur. De SFA-architectuur wordt gebruikt om de configuratie en besturing van testbeds over heel de wereld gelijk te maken.

Hoofdstuk 3 maakt een analyse van de bestaande FIRE en GENI monitor en geeft hierbij ook de probleemstelling aan.

# Hoofdstuk 1

# Situering

In dit hoofdstuk wordt het achterliggende kader van de masterproef geschetst. Daarnaast wordt ook de opdracht uitgewerkt. De opdracht bestaat uit 2 grote delen enerzijds een monitoringsservice maken om testbeds te controleren. De informatie van deze monitoringsservice wordt via de monitoringsAPI beschikbaar gesteld aan de buitenwereld. Anderzijds wordt de monitoringsAPI geïntegreerd in een groter Amerikaans framework.

## 1.1 Situering

Deze masterproef is een onderdeel van een groter Europees onderzoeksproject genaamd FIRE (Future Internet Research and Experimentation). FIRE is gericht is op onderzoek naar toekomstige internet- en netwerktechnologiën. Door onderzoekscentra te laten samenwerken(FIRE, 2014), tracht FIRE het onderzoek vlotter te laten verlopen. FIRE heeft twee grote doelen. Enerzijds de samenwerking tussen verschillende onderzoekscentra te verbeteren, anderzijds het delen van testfacilities makkelijker te maken.

Het eerste doel is de samenwerking tussen verschillende onderzoekscentra te verbeteren. Onderzoekers binnen eenzelfde vakgebied komen vaak gelijkaardige problemen tegen. FIRE vermijdt dat men telkens het wiel opnieuw uitvindt, door deze onderzoekers makkelijker en meer te laten samenwerken. Hierdoor worden oplossingen en ideeën meer gedeeld, zodat de ontwikkeling sneller kan verlopen.

Het tweede doel is het delen van testfaciliteiten makkelijker te maken. Door FIRE krijgt een onderzoeker van een onderzoekscentrum toegang tot testfaciliteiten van andere onderzoekscentra binnen FIRE. Testfaciliteit is een algemene term die duidt op zowel hardware als software dat gebruikt wordt om testen te verrichten. Een concreet voorbeeld van een testfaciliteit is een testbed. Een testbed is een server of een verzameling servers waarop men experimenten laat lopen. Zo kan er op een testbed bijvoorbeeld een server en een aantal cliënts gesimuleerd worden. Deze worden verbonden met een aantal tussenliggende routers. Vervolgens wordt een videostream opgestart. Op deze videostream kan men storing introduceren door pakketten te droppen. Deze storing zal ervoor zorgen dat het beeld aan de client-side hapert. Er kunnen technieken ingebouwd worden aan client-side om deze storing op te vangen. Zo kan er overgeschakeld worden naar een lagere kwaliteit indien blijkt dat de beschikbare bandbreedte onvoldoende is. Testen van degelijke technieken verloopt dan ook aan de hand van testbeds.

Het probleem dat zich hier stelt is dat elk testbed op zijn eigen manier werkt. Onderzoekers hebben nu wel toegang tot andere testbeds, maar moeten voor elk testbed eerst de nieuwe configuratie leren. Verschillende testbeds laten samenwerken op deze manier is geen sinecure. Om deze configuratie gelijk te maken heeft men de federation architectuur ingevoerd. De invoering van deze architectuur, binnen FIRE is een onderdeel van het FED4FIRE-project (Federation 4 FIRE). De federation architectuur die hier gebruikt wordt is SFA 2.0. Deze architectuur heeft als doel om de configuratie en werking van testbeds verdeeld over de wereld gelijk te maken.

SFA 2.0 werkt met 3 niveau voor verantwoordelijkheid. De bovenste is de MA (Management Authority), deze is verantwoordelijk voor de stabiliteit van een heel testfaciliteit. Een SA (slice authority) is verantwoordelijk voor een of meerdere slices. De laatste is een gebruiker, bijvoorbeeld een onderzoeker die een experiment wil uitvoeren op een testbed.

Daarnaast maakt SFA ook gebruik van een specifieke naamgeving. Een component is een primaire block van de architectuur, bijvoorbeeld een computer of een router. Meerdere componenten worden vervolgens gegroepeerd in aggregaten. Alle componenten van een aggregaat vallen onder dezelfde MA (Management Authority). Elke aggregaat wordt gecontrolleerd door een AM (aggregate manager). De AM beheert de allocatie van de verschillende experimenten op de aggregaat.

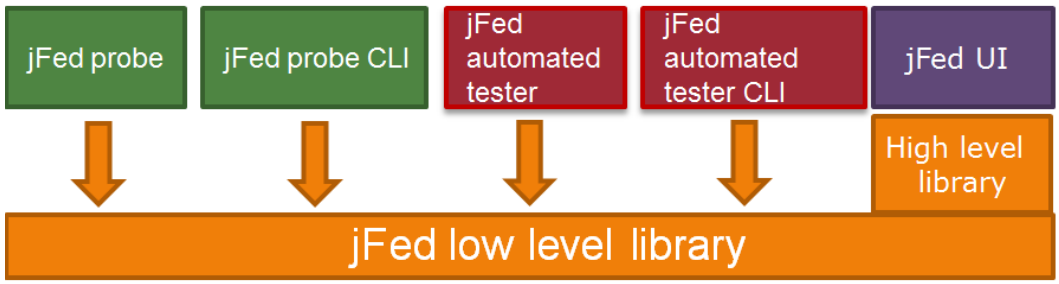
De MA (Management Authority) bepaald vervolgens hoe de resources verdeeld worden. Indien een component gemultiplexed wordt spreekt van men slivers. De gebruiker heeft dan een sliver van de component tot zijn beschikking. Meerdere slivers worden gegroepeerd tot aan slice. Een experiment wordt ook uitgevoerd binnen een slice. Meer uitleg over SFA architectuur volgt later in de scriptie.

Om het leven van onderzoekers makkelijker te maken is er de tool jFed. jFed werd door iMinds ontwikkeld(iMinds, 2014b) en is een javatool die de SFA architectuur gebruikt om testbeds aan te sturen. iMinds is een onafhankelijk onderzoekscentrum dat opgericht werd door de Vlaamse overheid(iMinds, 2014c). iMinds is leider van het FED4FIRE project(iMinds, 2014a).

Met behulp van jFed kunnen onderzoekers snel en eenvoudig netwerken simuleren en testen uitvoeren. Toch is er nog ruimte voor verbetering in jFed. Een van de voornaamste problemen is dat een onderzoeker niet weet of het testbed dat hij gebruikt betrouwbaar is. Bepalen of een vreemd gedrag in een experiment te wijten is aan eigen ontwikkelingen of aan het falen van een testbed, kan op deze manier zeer tijdrovend zijn.

Om dit probleem op te lossen heeft iMinds een monitoringssysteem uitgebouwd(, iMinds). Dit monitoringssysteem werkt, maar is door de snelle ontwikkeling niet voorzien op uitbereidingen. Deze masterproef zal enerzijds een monitoringsservice maken die deze testbeds in de gaten houdt. Anderzijds zal deze informatie via een monitoringsAPI beschikbaar gemaakt worden voor onderzoekers. Het is de bedoeling dat deze API een stevige basis vormt waarop andere applicaties kunnen gebouwd worden. Merk op dat monitoring op 3 niveau's mogelijk is: component,slice en aggregate. De monitoringsservice die hier besproken wordt, richt zich op de bovenste laag. Deze laag kijkt of een testbed online is en hoeveel resources er beschikbaar zijn. Deze informatie is momenteel beschikbaar via een aantal websites, maar zit nog niet in de primaire gebruikersinterface.

Deze monitoringsservice zal gebruik maken van een module van jFed. jFed bestaat immers uit verschillende modules (Figuur 1.1). De jFed low level library zorgt samen met de high level library voor o.a. de beveiliging van verbindingen en het omzetten van argumenten naar de juiste codering. De jFed UI is de userinterface. De probe wordt hier buiten beschouwing gelaten. De jFed automated tester en automated tester CLI zijn, in het kader van deze masterproef, wel belangrijk. Deze zorgen immers voor het uitvoeren van monitoringstesten.




**Figuur 1.1:** Opbouw van jFed.

FIRE reikt echter verder dan Europa alleen, zo zijn er ook overeenkomsten met onderzoeksprojecten buiten Europa. Een voorbeeld daarvan is GENI (Global Environment for Network Innovations). Geni is een Amerikaans onderzoeksproject gericht om aggregaten te bundelen en beschikbaar te stellen aan onderzoekers(GENI, 2014a). GENI maakt, net als FIRE, gebruik van de SFA architectuur(GENI, 2014b). Hierdoor is het mogelijk om onderzoekers van FIRE te laten werken op testbeds van GENI en omgekeerd.

GENI heeft zelf ook een gedistribueerde monitoringservice uitgebouwd(GENI, 2014c). Deze service maakt gebruik van datastores(GENI, 2014d). Een datastore houdt de monitoring informatie van een testbed of aggregate bij. Deze informatie wordt dan opgehaald door een collector. De webservice van FED4FIRE zou ook als een datastore bekeken worden. Op deze manier kan de monitoringsAPI geïntegreerd worden in een groter monitoringsframework. Dit zal als tweede deel van de masterproef behandeld worden. De werking van de GENI monitor wordt later in de scriptie uitgebreid besproken.

# Hoofdstuk 2

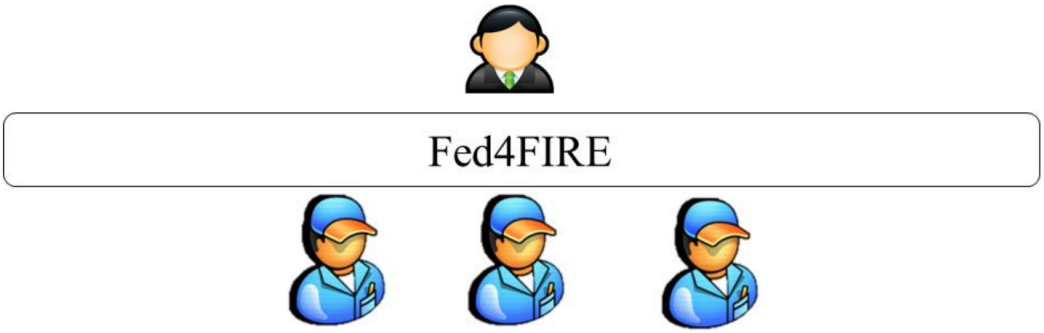
# SFA-Architectuur

Deze masterproef zal als onderdeel van FIRE, een Europees onderzoeksproject naar een innovatief internet, een monitoringsservice maken met een bijhorende API. FIRE gebruikt voor zijn testbedden de SFA architectuur. Deze architectuur is ontwikkeld om de configuratie en aansturing van testbeds over de hele wereld gelijk te maken. Hierdoor moeten onderzoekers maar een configuratie leren, waarmee ze vervolgens op elk testbed kunnen werken. Dit hoofdstuk gaat dieper in op de SFA-architectuur. Hierbij worden de relevante kern begrippen besproken.

## 2.1 Doel

SFA (Slice-based Federation architecture) is een framework dat gebruikt wordt om testbeds aan te sturen(Peterson *et al.*, 2010). SFA is gebaseerd op FFA (First Federation Architecture) en wordt gebruikt om een van de FIRE doelstellingen, het delen van testbeds makkelijker maken, waar te maken. Doordat alle testbeds op een andere manier werkten, was het voor een onderzoeker erg moeilijk om verschillende testbeds te gebruiken. Een onderzoeker moest eerst kennis maken met de specifieke configuratie van een testbed, alvorens hij ermee kon werken. Hierdoor kwam de noodzaak om de configuratie van testbeds gelijk te trekken.

SFA is een controle framework om testbeds mee aan te sturen. Het idee is dat SFA een standaard is die door testbeds geïmplementeerd wordt. Eenmaal dat is gebeurd kan een onderzoeker die kennis heeft van SFA kan werken, direct ook werken met alle testbeds die er compatibel mee zijn. Zoals te zien is in Figuur 2.1 komt de onderzoeker de klant en het testbed is de service provider.




**Figuur 2.1:** De onderzoeker (boven) is de klant, de testbeds (onderhouden door onderste personen) is de service provice. Fed4FIRE voorziet de link tussen beide partijen.

SFA voorziet een aantal functionaliteiten. De eerste is voorzien dat het beleid van de owner nageleefd wordt. SFA voorziet in mechanismen om dit te controleren, waarbij verondersteld wordt dat er meerder eigenaars zijn. Deze eigenaars hebben elk een testfaciliteit en vormen samen een federatie. Het beleid is vastgelegd binnen een federatie of verzameling testbeds.

SFA moet ook voorzien dat operators onderhoud kunnen uitvoeren, hiervoor moet het mogelijk zijn om machines te verwijderen of te vervangen. Ook toevoegen van nieuwe machines moet mogelijk zijn. Daarnaast moeten onderzoekers de mogelijkheid krijgen om slices aan te maken. Een slice is een container waarin een experiment draait. Verder moet het mogelijk zijn voor eigenaars om de autorisatie te controleren. Hierbij is het bijvoorbeeld mogelijk om maar een beperkt aantal mensen toegang te verlenen.

## 2.2 Entiteiten

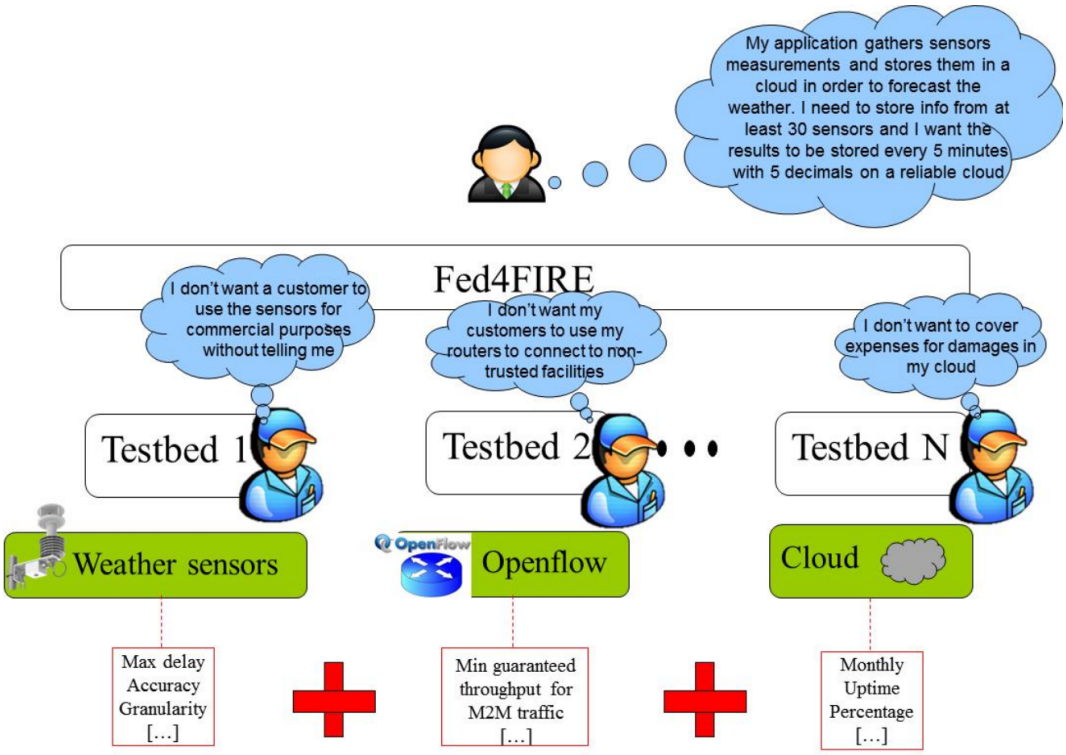
SFA herkent 4 entiteiten.

1. Owners
2. Operators
3. Researchers
4. Identity anchors / identity providers

Wat hierop volgt is een bespreking van elke entiteit met zijn verantwoordelijkheden.

### 2.2.1 Owners

De eigenaars of verantwoordelijke voor het testbed. De eigenaar is verantwoordelijk voor de werking van zijn (deel van) het testbed. De Owners bepalen welke beleidsregels er van toepassing zijn. Deze beleidsregels worden aangeduid met SLA (Service Level Agreements). Zo kan het zijn dat de eigenaar van een testbed niet wil dat er commerciële testen gebeuren zonder dat hij daarvan op de hoogte is. Figuur 2.2 geeft een voorbeeld van een aantal mogelijke beleidsregels.




**Figuur 2.2:** Eigenaars bepalen het beleid van hun testbed.

### 2.2.2 Operators

Operators voorzien het onderhoud van het testbed. Dit onderhoud omvat o.a. herstellingswerken, beveiliging, voorkomen van schadelijke activiteiten.

### 2.2.3 Researchers

De Onderzoekers is de klant, hij gebruikt een testbed om experimenten uit te voeren. Deze experimenten verlopen in kader van zijn onderzoek.

### 2.2.4 Identity providers

Een identity provider of identity anchor is iemand die entiteiten rechten kan geven. Zo kan een identity provider een hoofdonderzoeker rechten geven om onderzoekers binnen zijn project te laten werken.

## 2.3 Opbouw

Een testbed is opgebouwd uit meerdere onderdelen die opgedeeld kunnen worden in meerdere lagen. Volgende tekst bespreekt deze onderdelen.

### 2.3.1 Component

Een testbed bestaat uit vele onderdelen. Een primaire bouwblok van een testbed is een component. Een component is bijvoorbeeld een computer, router of programmeerbaar access point. Indien de component een computer is, wordt deze ook een node genoemd. Een node is dus een computer , meestal binnen een testbed, verbonden met het netwerk.

### 2.3.2 Aggregate

Al deze componenten worden gegroepeerd in aggregates of aggregaten. Een aggregaat is een verzameling componenten die onder eenzelfde beheerde valt. Zo is de virtual wall2 van iMinds een aggregaat omdat het beheer van dit volledige testbed onder iMinds valt.

### 2.3.3 Aggregate manager

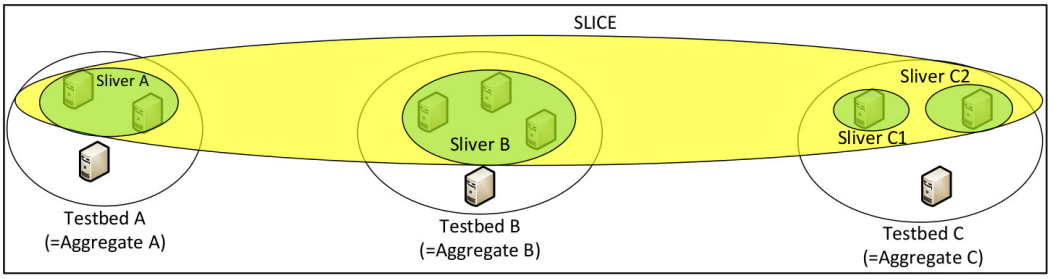
Elke aggregate wordt beheerd door een AM (aggregate manager). De aggregate manager is een stuk software dat een interface aanbied aan onderzoekers. Via deze interface kan bijvoorbeeld een slot gereserveerd worden om een experiment op te zetten. Een aggregate manager vervult taken zoals 'stukken van het testbed', slices genaamd, toe te wijzen aan onderzoekers of een experiment.

### 2.3.4 Sliver

Een component kan echter ook gemultiplexed worden zodat er meerdere experimenten tegelijk op kunnen draaien. Dit kan door bijvoorbeeld virtualisatie toe te passen. Het 'stuk' van een component wordt een sliver genoemd.

### 2.3.5 Slice

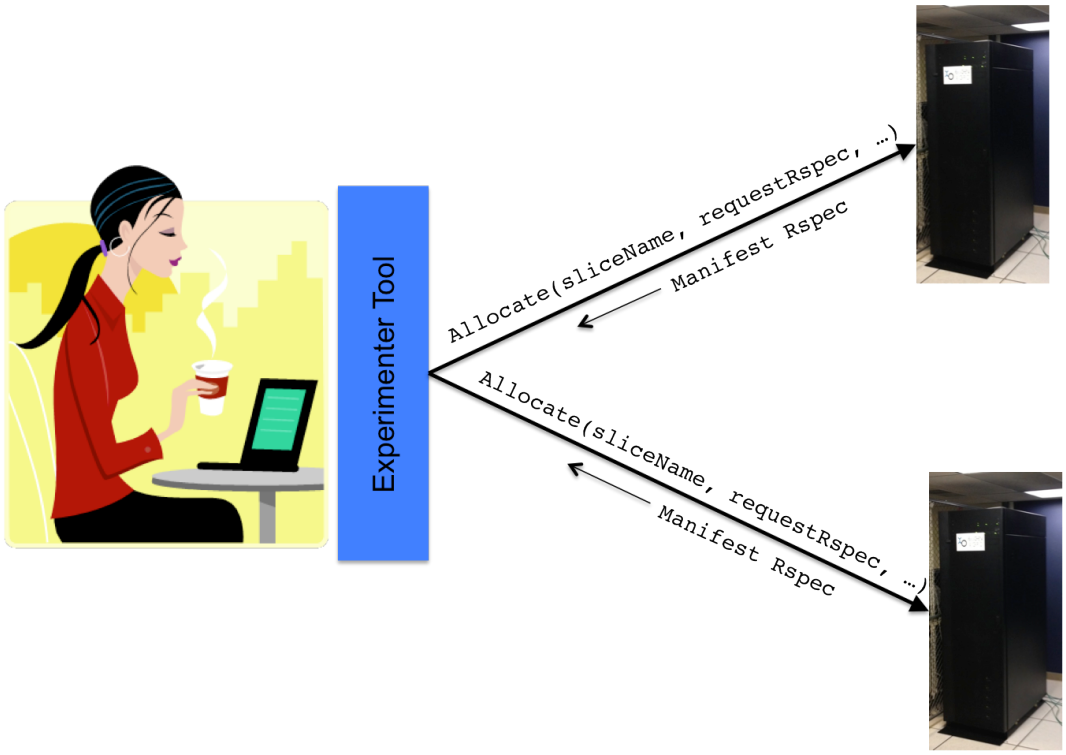
Een slice is een verzameling van slivers. Een slice is een abstract begrip dat omschreven kan worden als een container waarin een experiment draait. Vanuit het perspectief van de onderzoeker komt dit overeen met de testopstelling die hij ter beschikking heeft. Vanuit het perspectief van de owner of eigenaar van het testbed is dit een administratieve opdeling om bij te houden welke testen er waar gebeuren. Figuur 2.3 geeft een voorbeeld van een slice. Merk op dat een slice over meerdere aggregates heen kan lopen.




**Figuur 2.3:** Een slice (geel) bestaat uit een verzamling slivers (groen).

## 2.4 RSpec

Een RSpec (Resource Specification) , is een xml file die een proefopstelling beschrijft(GENI, 2014e). RSpecs kunnen opgedeeld worden in 3 groepen. De eerste soort is een request RSpec, deze beschrijft welke resources een onderzoeker wil gebruiken in zijn experiment. De AM (aggregate manager) antwoordt hierop met een manifest RSpec, zoals beschreven in Figuur 2.4 . Deze RSpec beschrijft de resources die gealloceerd zijn voor het experiment. Merk op dat dit proces transparant gebeurd, de meeste jFed tools zullen automatisch een RSpec genereren. Toch is het mogelijk om RSpecs zelf aan te maken, wat vooral voor complexere testopstellingen gebeurd.




**Figuur 2.4:** Een onderzoeker, of de tool die hij gebruikt stuurt eerst een request RSpec en krijgt vervolgens een manifest RSpec terug.

De derde soort RSpec, de advertisement RSpec, wordt gebruikt bij de listResourcetest. De listResourcetest komt later in de scriptie aan bod. Een advertisement RSpec lijst alle resources op die beschikbaar zijn op een testbed.

Voor een onderzoeker is een RSpec een beschrijving van een experiment. Het voordeel van dit formaat in plain tekst is dat onderzoekers zeer eenvoudig experimenten kunnen herhalen. Doordat een RSpec een volledige beschrijving is van een proefopstelling, is dit een meerwaarde in wetenschappelijke verslagen.

# Hoofdstuk 3

# Analyse en probleemstelling

Zoals eerder besproken, zal deze masterproef een monitoringsAPI maken in opdracht van iMinds, een onderzoekscentrum. De monitoringsAPI heeft als doel de resultaten van de achterliggende monitoringsservice aan te bieden. De monitorService zal op zijn beurt alle aggregaten (testfaciliteiten) binnen FIRE (Future Internet Research and Experimentation), een project voor de verbetering van onderzoek naar internet en netwerken, in de gaten houden. Dit hoofdstuk zal de werking van de FIRE monitor beschrijven. De FIRE monitor is de monitoringsservice die bij aanvang van de masterproef de monitoring verzorgde. Daarna zal er dieper ingegaan worden op de probleemstelling. Vervolgens wordt ook de GENI (Global Environment for Network Innovations) monitor kort besproken. GENI is een Amerikaans project met gelijkaardige doelstellingen als FIRE.

## 3.1 FIRE Monitor

iMinds (onderzoekscentrum waar de masterproef uitgewerkt wordt en tevens leidinggevend in het FIRE project), heeft een monitoringsservice gemaakt die al enige tijd draait (, iMinds), de FIRE monitor. Deze monitoringsservice is ruimer dan de monitoring van testbeds, ook het monitoren van experimenten wordt door deze service afgehandeld(, iMinds).

### 3.1.1 Componenten

Deze monitoringsservice bestaat uit verschillende componenten. De eerste component is de Facility monitoring, deze monitoring wordt gebruikt bij de FLS (first level support). De first level support heeft als doel om de basis zaken van monitoring af te handelen. De voornaamste test is de pingtest die kijkt of een testbed nog online is. De aggregate bepaald zelf welke testen er uitgevoerd worden.

De tweede component is de infrastucture monitoring. Deze component is gericht om componenten binnen een experiment. De verzamelde data bevat o.a. aantal verstuurde pakketten, aantal verloren pakketten, cpu-load, ... .

Een derde component is de OML measurement library, deze bibliotheek laat het toe dat een onderzoeker zijn eigen monitoring framework gebruikt om de metingen van zijn experiment te doen.

Deze masterproef richt zich op de Facility monitoring. De tweede en de derde component zijn hier minder relevant en worden verder buiten beschouwing gelaten. De monitoringsservice waarnaar verwezen wordt, is bijgevolg de Facility Monitoring.

Deze monitoringsservice (Facility Monitoring) is opgedeeld in een aantal stukken. Het eerste stuk is de FLS-monitor (First Level Support). Deze is beschikbaar op https://flsmonitor.fed4fire.eu/,zie ook Figuur 3.1 . Deze service heeft het doel actuele informatie weer te geven over de status van het testbed.

Fed4FIRE First Level Support Monitoring

| Testbed Name | Ping latency (ms) | GetVersion Status | Free Resources | Internal testbed monitoring status | Last check internal status |
|---|---|---|---|---|---|
| BonFIRE | 31.17 | N/A | N/A | ok | 2014-03-25 11:16:09+01 |
| Fuseco | 16.83 | ok | 1 | ok | 2014-03-25 11:16:03+01 |
| Koren | 284.47 | N/A | N/A | N/A | N/A |
| NETMODE | 67.33 | ok | 20 | ok | 2014-03-25 11:13:22+01 |
| NITOS Broker | 73.03 | ok | 38 | ok | 2014-03-25 11:15:01+01 |
| NITOS SFAWrap | 31.29 | ok | 65 | N/A | N/A |
| Norbit | N/A | N/A | N/A | ok | 2014-03-25 11:10:29+01 |
| Ofelia (Bristol island) | 11.89 | N/A | N/A | ok | 2014-03-25 11:10:02+01 |
| Ofelia (i2CAT island) | N/A | N/A | N/A | ok | 2014-03-25 11:10:02+01 |
| Planetlab Europe | 32.11 | ok | 285 | ok | 2014-03-25 11:15:02+01 |
| SmartSantander | 53.82 | ok | 0 | ok | 2014-03-25 11:10:01+01 |
| Virtual Wall | 0.21 | ok | 23 | ok | 2014-03-25 11:14:39+01 |
| w-iLab.t 2 | 20.33 | ok | 68 | ok | 2014-03-25 11:14:58+01 |

**Figuur 3.1:** FLS testbed monitoring

Figuur 3.1 geeft een beeld van de monitoringssite. De laatste 2 kolommen zijn van minder belang. De eerste kolom geeft de naam van het testbed weer. Daarnaast wordt het resultaat van de laatste ping test getoond. De volgend 2 kolommen bevatten het resultaat van respectievelijk de getVersiontest en de free resources test.GetVersion geeft aan of de AM (aggregate manager) nog werkt terwijl de kolom free resources aangeeft hoeveel resources er nog beschikbaar zijn. De vorm van deze testen is relatief eenvoudig, een enkelvoudig resultaat wordt teruggegeven.

Het tweede deel van de monitoringsservice, nightly login testing, bevat complexere testen. Deze testen worden typisch 1 tot 2 keer per dag uitgevoerd. Deze testen zijn diepgaander dan de FLS-monitor. Een belangrijke test die hier gebeurd is de logintest. Hierbij wordt een getest of het aanmelden op een testbed mogelijk is. Een andere test die uitgevoerd wordt, is de stitchingtest. Deze zal kijken of het mogelijk is om een netwerk op te zetten tussen verschillende testbeds. Zie Figuur 3.2

jFed Automated Scenario Tests

Options: Show code tag

| Category | Test Name | Last Test Start Time (CET) | Last Test Duration | Last Partial Success | Last Full Success | Time since last Failure | Last Log | History |
|---|---|---|---|---|---|---|---|---|
| login | Fuseco | 2014-03-25 09:17:47 | 14 seconds | FAILURE | FAILURE | | log | history |
| login | InstaGeni BBN | 2014-03-25 09:18:01 | 3 minutes and 6 seconds | SUCCESS | SUCCESS | 9 days and 13 hours | log | history |
| login | InstaGeni Clemson | 2014-03-25 09:21:08 | 45 seconds | WARN | WARN | | log | history |
| login | InstaGeni GATech | 2014-03-25 09:21:54 | 2 minutes and 24 seconds | SUCCESS | SUCCESS | 27 days and 14 hours | log | history |
| login | InstaGeni Illinois | 2014-03-25 09:24:20 | 3 minutes and 27 seconds | SUCCESS | SUCCESS | 5 months and 11 days | log | history |
| login | InstaGeni Kettering | 2014-03-25 09:27:50 | 10 seconds | FAILURE | FAILURE | | log | history |
| login | InstaGeni MAX | 2014-03-25 09:28:00 | 2 minutes and 55 seconds | SUCCESS | SUCCESS | 2 months and 9 days | log | history |

**Figuur 3.2:** resultaten van de stitching test

Zoals zichtbaar in Figuur 3.3, is het ook mogelijk om de geschiedenis van deze testen op te vragen.

jFed Automated Scenario Tests

Options: Show code tag

| Category | Test Name | Test Start Time | Test Duration | Partial Success | Full Success | Log |
|---|---|---|---|---|---|---|
| stitching | Stitching vwall1 - vwall2 | 2014-04-21 03:50:33 | 6 seconds | FAILURE | FAILURE | log |
| stitching | Stitching vwall1 - vwall2 | 2014-04-21 04:07:34 | 2 minutes and 27 seconds | FAILURE | FAILURE | log |
| stitching | Stitching vwall1 - vwall2 | 2014-04-21 04:18:35 | 9 minutes and 46 seconds | SUCCESS | FAILURE | log |
| stitching | Stitching vwall1 - vwall2 | 2014-04-21 10:14:35 | 9 minutes and 26 seconds | SUCCESS | FAILURE | log |
| stitching | Stitching vwall1 - vwall2 | 2014-04-21 20:13:10 | 20 minutes | FAILURE | FAILURE | log |
| stitching | Stitching vwall1 - vwall2 | 2014-04-21 20:33:29 | 3 minutes and 13 seconds | SUCCESS | FAILURE | log |
| stitching | Stitching vwall1 - vwall2 | 2014-04-21 21:55:49 | 3 minutes and 19 seconds | SUCCESS | FAILURE | log |
| stitching | Stitching vwall1 - vwall2 | 2014-04-21 23:12:05 | 3 minutes and 17 seconds | SUCCESS | SUCCESS | log |
| stitching | Stitching vwall1 - vwall2 | 2014-04-22 10:13:13 | 3 minutes and 17 seconds | SUCCESS | SUCCESS | log |
| stitching | Stitching vwall1 - vwall2 | 2014-04-22 20:22:31 | 3 minutes and 14 seconds | SUCCESS | SUCCESS | log |
| stitching | Stitching vwall1 - vwall2 | 2014-04-23 10:20:53 | 3 minutes and 16 seconds | SUCCESS | SUCCESS | log |

**Figuur 3.3:** geschiedenis van resultaten

### 3.1.2 testen

De FLS monitor voert 3 soorten testen uit.

1. Een ping test die kijkt of een testbed nog online is. Dit is een eenvoudig ping commando dat regelmatig uitgevoerd wordt.
2. Een getVersion call naar de AM (aggregate manager) API. Doordat de getVersion call geen authenticatie vereist, wordt deze test gebruikt om te testen of een AM nog werkt.
3. De listResources test, hiermee wordt gekeken hoeveel resources er nog beschikbaar zijn. Indien dit nul is, is het niet mogelijk om nieuwe testen te starten.

Daarnaast zijn er nog een aantal test gedefinieerd. De belangrijkste zijn hier vermeld.

1. De login test, deze test zal proberen om aan te melden op een testbed.
2. Stitching test, deze test is redelijk complex. Vereenvoudigd zal een stitching test 2 verschillende testbeds met elkaar verbinden. Hiervoor wordt er ingelogd op beide testbeds en vervolgens wordt er op elk testbed een node gereserveerd. Dan worden deze nodes met elkaar verbonden en probeert men te pingen.

### 3.1.3 Werking FLS monitor

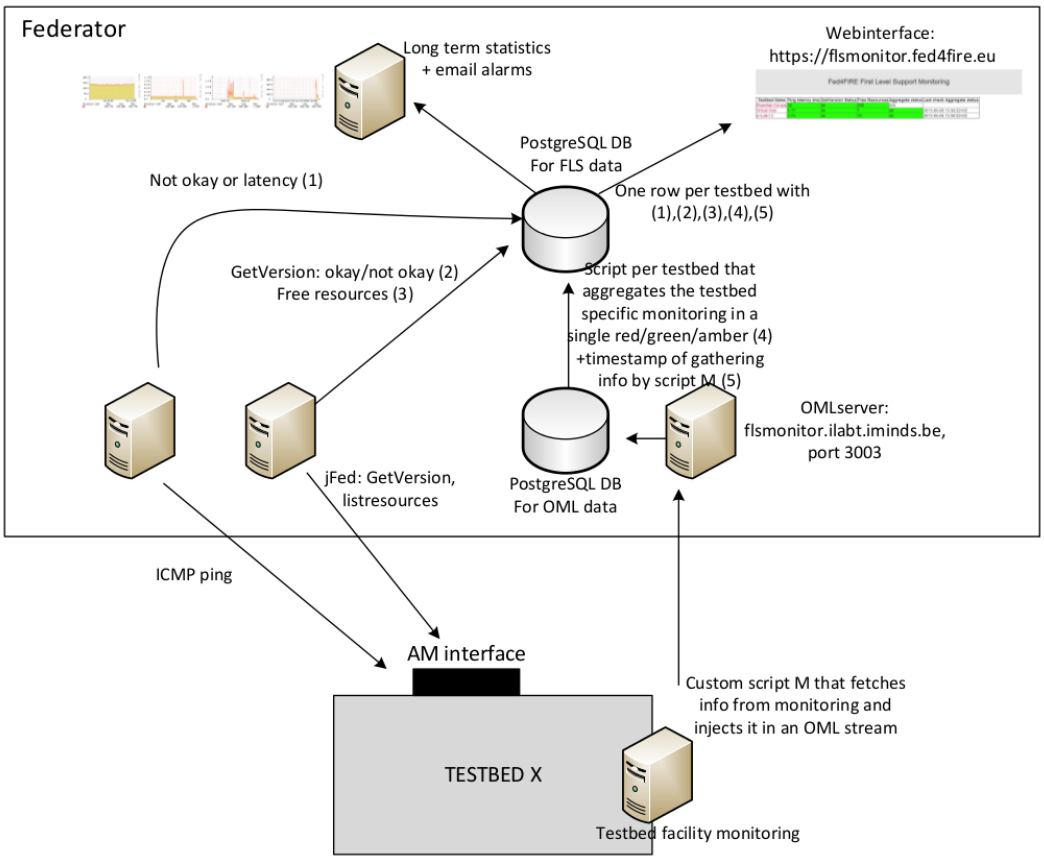
Figuur 3.4 geeft weer welke data er opgevraagd wordt. De eerste kolom is geeft de naam van het testbed weer. De tweede kolom geeft de latency weer. De derde en vierde kolom komen overeen met de getVersion call en de listResources call. De 5e kolom houdt de interne status van het testbed bij, dit wordt door het testbed zelf ingevuld en doorgegeven. Ten slotte is er nog een timestamp die aangeeft wanneer de laatste update verlopen is. Indien een timestamp te oud wordt, duidt dat op problemen met de monitor die draait op het testbed.

Fed4FIRE First Level Support Monitoring

| Testbed Name | Ping latency (ms) | GetVersion Status | Free Resources | Internal testbed monitoring status | Last check internal status |
|---|---|---|---|---|---|
| BonFIRE | 31.17 | N/A | N/A | ok | 2014-03-25 11:16:09+01 |
| Fuseco | 16.83 | ok | 1 | ok | 2014-03-25 11:16:03+01 |
| Koren | 284.47 | N/A | N/A | N/A | N/A |
| NETMODE | 67.33 | ok | 20 | ok | 2014-03-25 11:13:22+01 |
| NITOS Broker | 73.03 | ok | 38 | ok | 2014-03-25 11:15:01+01 |
| NITOS SFAWrap | 31.29 | ok | 65 | N/A | N/A |
| Norbit | N/A | N/A | N/A | ok | 2014-03-25 11:10:29+01 |
| Ofelia (Bristol island) | 11.89 | N/A | N/A | ok | 2014-03-25 11:10:02+01 |
| Ofelia (i2CAT island) | N/A | N/A | N/A | ok | 2014-03-25 11:10:02+01 |
| Planetlab Europe | 32.11 | ok | 285 | ok | 2014-03-25 11:15:02+01 |
| SmartSantander | 53.82 | ok | 0 | ok | 2014-03-25 11:10:01+01 |
| Virtual Wall | 0.21 | ok | 23 | ok | 2014-03-25 11:14:39+01 |
| w-iLab.t 2 | 20.33 | ok | 68 | ok | 2014-03-25 11:14:58+01 |

**Figuur 3.4:** FLS testbed monitoring

Figuur 3.5 beschrijft de werking van de FLS monitor. Punt 1 stelt de pingtest voor. Hierbij wordt gekeken of de latency beneden een waarschuwingswaarde valt. De ping test verloopt rechtstreeks naar het testbed en niet via de AM. Punt 2 en 3 zijn de getVersion en listResources call naar de AM op het testbed. Als deze data wordt bijgehouden in een postgreSQL databank. Punt vier gaat over de interne status van het testbed. Dit wordt echter niet geïmplementeerd in de masterproef. Rechtsboven is de webview weergeven, deze haalt de resultaten rechtstreeks uit de databank. Linksboven is de federator weergegeven. De federator is een component binnen een federatie. Deze component mag vrij ingevuld worden en wordt gebruikt binnen een federatie om het beheer vlotter te laten verlopen. Zo wordt in dit voorbeeld de statistieken op lange termijn bijgehouden.




**Figuur 3.5:** Werking van de FLS monitor

De FIRE monitor heeft dus een centrale database waarin de monitor informatie zit. Deze database is de verbinding tussen de website en de resultaten van de test. Langs de ene kant wordt de uitvoer van de testen in de databank geüpdatet. Langs de andere kant wordt deze data opgevraagd door de website. Het is echter zo dat de FLS monitor zelf geen lange termijn statistieken bijhoudt. Dit komt doordat er bij het uitvoeren van testen geen nieuwe lijnen toegevoegd worden aan de databank. In plaats daarvan wordt de lijn geüpdatet.

## 3.2 Probleemstelling

### 3.2.1 Bereikbaarheid

Het voornaamste probleem is niet de structurering van de data, maar de bereikbaarheid. In de vorige versie is de data enkel bereikbaar via de webinterface, of rechtstreeks via de databank. Dit maakt het moeilijk voor nieuwe ontwikkelingen om deze data te gebruiken. Deze masterproef lost dit probleem op door het gebruik van een montoringsAPI. Deze zorgt ervoor dat de data vlot beschikbaar is via aanvragen aan de API.

Het ontwerpen van deze monitoringsAPI is het eerste deel van de masterproef. Deze API moet een vlotte toegang tot de resultaten garanderen. Ook moet het mogelijk zijn om deze resultaten de filteren. Een filter die er zeker noodzakelijk is, is het opvragen van de laatste resultaten van een testbed. Eenmaal deze api af is, kan alle communicatie met de achterliggende databank verlopen via de API. De website's die momenteel bestaat (zie Figuur **??**), zal nu niet meer rechtstreeks contact maken met de databank. In plaats daarvan zal de API gebruikt worden als datalaag.

Ook toevoegen van nieuwe resultaten, door de monitoringsservice verloopt via de API. Dit heeft als voordeel dat de API complexere zaken zoals foutafhandeling kan afhandelen. Een bijkomend voordeel is dat de databank niet extern bereikbaar moet zijn. Authenticatie kan in een hogere laag afgehandeld worden. Merk op dat authenticatie geen onderdeel is van deze masterproef.

Daarnaast moet ook gekeken worden naar bestaande monitoringssystemen. Zo heeft GENI (Global Environment for Network Innovations) een dergelijk framework. Integratie met dit framework zal de samenwerking tussen beide partijen bevorderen. De GENI monitor wordt later in dit hoofdstuk besproken.

### 3.2.2 Structuur

Naast de bereikbaarheid is er ook een structureel probleem. De testen van de FLS monitor hebben een eenvoudige structuur. De testen zijn eenvoudig omdat ze slechts een beperkt aantal parameters nodig hebben en een enkelvoudig resultaat teruggeven. Zo geeft een pingtest de ping waarde terug. Een listResources test geeft het aantal beschikbare resource terug. Beide waarden zijn gewoon getallen, die opgeslagen worden in een kolom van de databank.

De nightly login testen zijn echter complexer. Deze bestaan uit meerdere opeenvolgende stappen, elke stap heeft een tussenresultaat. Hierdoor volstaat een kolom niet meer. De oplossing die toen aangewend is, is het gebruiken van een tweede databank. In die databank is er niet voor elk tussenresultaat een kolom voorzien, maar heeft men de resultaten samen genomen in 2 tussen resultaten. Deze manier lost het probleem van het variabele aantal tussenresultaten op, maar geeft geen proper overzicht van alle tussen statussen. Voor deze informatie moet men zich wenden tot de log.

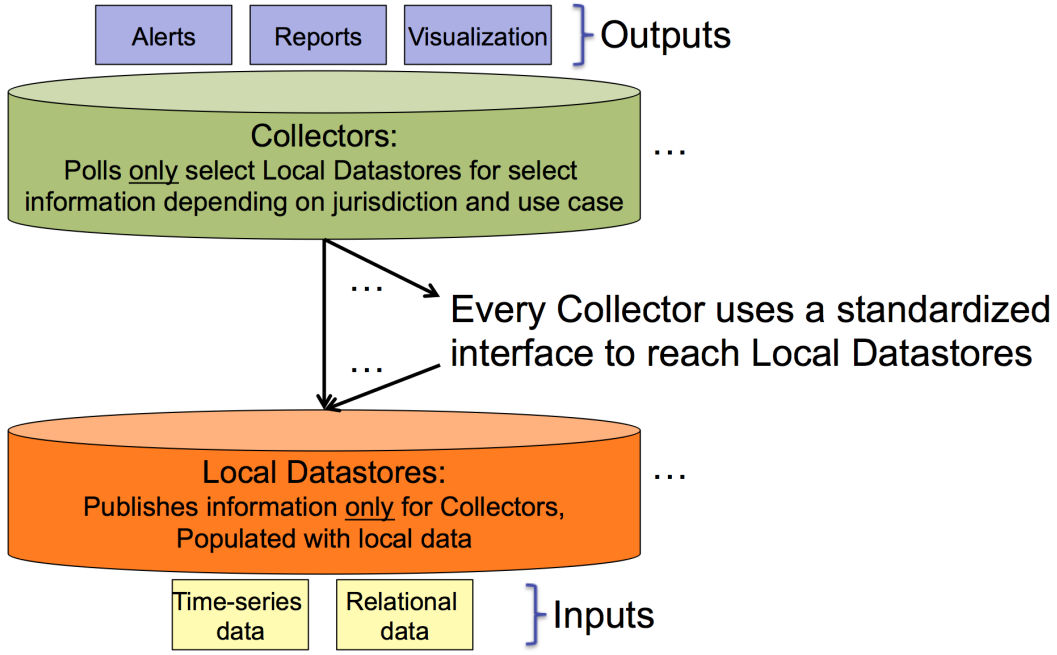
## 3.3 GENI monitor

Als er binnen FIRE een nieuw monitoringAPI gemaakt moet worden, is het nuttig om onderzoek te doen naar bestaande oplossingen hiervoor. Zo beschikt GENI over een monitoringsframework (GENI, 2014c), dat hieronder uitgelegd wordt.

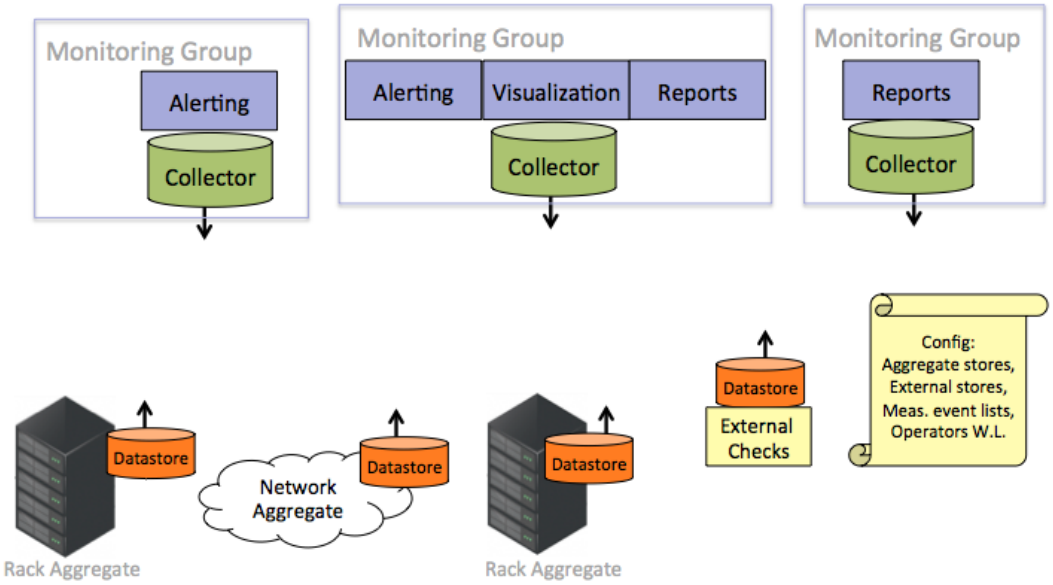
GENI (Global Environment for Network Innovations) is een amerikaans project dat , net zoals FIRE, een virtueel testlaboratorium heeft. Dit lab is gedeeld over meerdere onderzoekscentra. Aangezien GENI en FIRE beide bezig zijn met onderzoek naar innovatieve netwerk en internet ontwikkelingen, is de samenwerking tussen beide partijen vanzelfsprekend.

Testbeds binnen GENI maken ook gebruik van een Slice federation architectuur. GENI heeft ook een monitoring framework ontwikkeld. Dit framework werkt met meerdere databronnen, datastores. Deze datastores hebben een REST polling API om informatie op te halen. Deze API wordt aangesproken door een collector. Een collector verzameld de data. Dit principe is geschetst in Figuur 3.6

Een collector kan meerdere testbeds pollen. Hiermoet moet niet alle data opgehaald worden, het is mogelijk dat een collector resultaten van een test op verschillende testbeds ophaald. Een collector zal welke data dan ook nodig is ophalen om de aangesloten monitoring applicaties te laten werken. Normaal gezien zal een monitorings applicatie een collector gebruiken, al is dat geen vereiste. Dit is beschreven in Figuur 3.7. De boxen zijn de monitoringapplicaties. Deze hebben elk verschillende data nodig, weergegeven in blauw. Elke monitoringapplicatie heeft een collector (groen). De collector zal de nodige data pollen van een datastore (oranje).




**Figuur 3.6:** Principe geni datastore




**Figuur 3.7:** Een collector kan verschillende delen data ophalen.

Door de intense samenwerking tussen FIRE en GENI waarbij testbeds beschikbaar gemaakt worden voor onderzoekers, is het zeker handig om de monitorAPI compatibel te maken hiermee. Daarvoor moet de monitoringAPI een datastore vormen. Doordat de huidige FIRE monitor veel meer informatie ter beschikking heeft dan de datatore, zal deze masterproef kijken wat er geïntegreerd kan worden en of er uitbereidingen nodig zijn aan de datastoreAPI. Merk op dat het GENI datastore API nog in ontwikkeling is.

## 3.4 Besluit

De vorige service werkte wel, maar was niet voorzien op de komst van complexere testen. Er zijn 2 grote problemen, enerzijds de bereikbaarheid van de data, anderzijds de structuur. Het eerste probleem is opgelost door het maken van een API. Het tweede probleem is opgelost door het uitbouwen van een complexe databank. Deze databank houdt zowel de resultaten als de configuratie van de testen bij. Ook is databank voorzien voor het opslaan van tussenresultaten. Als laatste onderdeel is er de integratie met GENI. Hierbij zal onderzocht worden wat er mogelijk is en waarvoor er nog uitbereidingen nodig zijn.

# Bibliografie


FIRE (2014). What is fire. URL http://www.ict-fire.eu/getting-started/what-is-fire.html.

GENI (2014a). About geni. URL https://www.geni.net/?page_id=2.

GENI (2014b). The geni api. URL http://groups.geni.net/geni/wiki/GeniApi.

GENI (2014c). Geni operational monitoring project. URL http://groups.geni.net/geni/wiki/OperationalMonitoring.

GENI (2014d). Overview of operational monitoring. URL http://groups.geni.net/geni/wiki/OperationalMonitoring/Overview.

GENI (2014e). Resource specification (rspec). URL http://groups.geni.net/geni/wiki/GENIConcepts#TheGENIAMAPIandGENIRSpecs.

iMinds (2014a). Federation for fire. URL http://www.iminds.be/nl/projecten/2014/03/07/fed4fire.

iMinds (2014b). jfed : Java based framework to support sfa testbed federation client tools. URL http://jfed.iminds.be/.

iMinds (2014c). Over iminds. URL http://www.iminds.be/nl/over-ons.

B. V. (iMinds) (2014). D2.4 - second federation architecture. URL http://www.fed4fire.eu/fileadmin/documents/public_deliverables/D2-4_Second_federation_architecture_Fed4FIRE_318389.pdf.

L. Peterson, R. Ricci, A. Falk & J. Chase (2010). Slice-based federation architecture. URL http://groups.geni.net/geni/attachment/wiki/SliceFedArch/SFA2.0.pdf.