Erlang / Elixir

Services et Acteurs

Telecom Bretagne 2015 @arnaudwetzel

But de la présentation

- Comprendre l'essence d'un service informatique
- et les solutions d'Erlang/Elixir pour le créer.
- Du concept à l'approche pratique par l'exemple.

Plan

Approche: Étudier Erlang/Elixir par l'étude d'un service abstrait puis concret

- La création ou la production de service
- Gestion du temps & de l'état en erlang/elixir
- Aperçu du langage
- Exemple de projet: la vente d'espace publicitaire en temps réel.

Création de services

Un peu de hauteur

- Un language de programmation
- Un code informatique dans un language
- Protocole
- Format de données
- Spécification
- Cahier des charges
- Test

- Documents/données

Pourriez vous les définir ?

Une représentation (des données immatérielles) d'un savoir (concept) en vu de le communiquer d'une entité à une autre; créant ainsi de la valeur.

Concept = langage

Un concept *est* sa ou ses **représentations**, notre capacité à représenter est notre capacité à penser.

- qu'est ce qu'un chiffre
- qu'est ce qu'un cercle
- qu'est ce qu'un Ami, l'amour, etc.

On peut donc dire qu'un concept est une forme de *langage*.

Création = composition

La création *est* la possibilité de **composer** des concepts pour en créer un autre.

- composer des nombres (opérations: l'addition, etc.)
- composer des mots (phrases, textes)
- composer des matériaux (industrie)

On compose donc des *langages* pour en former d'autres.

Existence = temps/espace

Si `A*B=C` alors `C` existe **après** `A` et `B`

La relation d'ordre de l'existence *est* le temps. Les concepts sont donc ordonnées.

Pour obtenir C au MOMENT de B alors il faut donner matière à A, le mémoriser, pour lui permettre de traverser le temps et le transmettre pour lui permettre de traverser l'espace.

On compose donc des *langages* pour en former d'autres.

En résumé

- Concept <=> Représentation / langage
- Création de Concept <=> Composition
- Existence du concept <=> Temps/Espace <=> Mémoire/Communication

  Mathématiciens, philosophes, informaticiens et même juristes ou artistes sont des linguistes qui cherchent à composer pour créer et faire exister leurs créations.

Quel rapport ?

Le Service dans un SI est un concept, un langage qui fourni un service à une société humaine. Sa création est un processus de traduction réparti dans le temps grâce à une mémoire et avec lequel on communique.

L'ordinateur **connecté** est l'outil élémentaire de la création de service. Ce n'est pas un calculateur.

- il manipule des 0/1 (représentation élémentaire)
- dans des registres (mémoire élémentaire)
- récursivement (composition élémentaire)
- et communique via IO (communication limité)
- en réseau (communication élémentaire)

Manipuler le temps par la maintenance d'un état

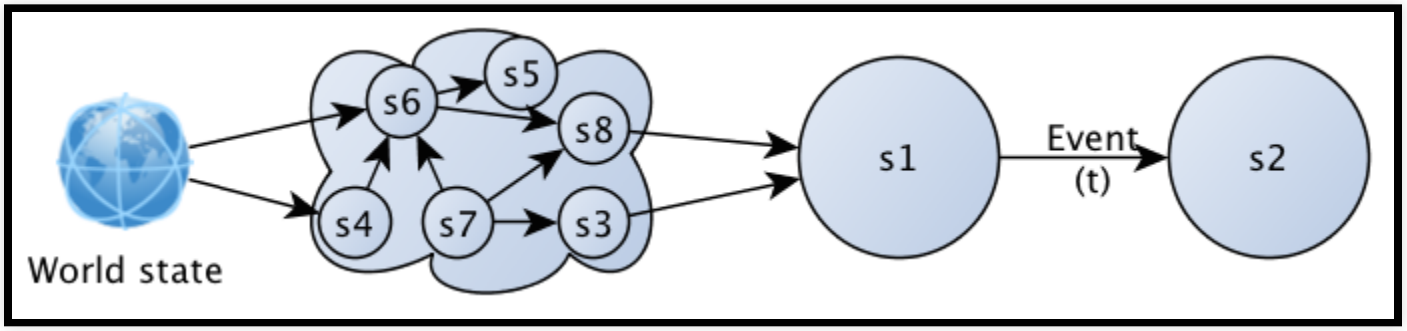
Un service: que connait le monde sur Telecom Bretagne ?

- porte à porte : l'homme
- stocker les connaissances, les copier les distribuer : le web
- parcourir ces connaissances pour extraire les concernées
- stocker des indexes

Service: `(évènement,état1)->état2`

Complexité

`(évènement,état1)->état2` : automate à état fini



- Votre main: 10^24 atomes
- Terre: 10^50 atomes
- Univers: 10^82 atomes
- Un programme de 34 octets: 2^(34*8) ~= 10^82 états

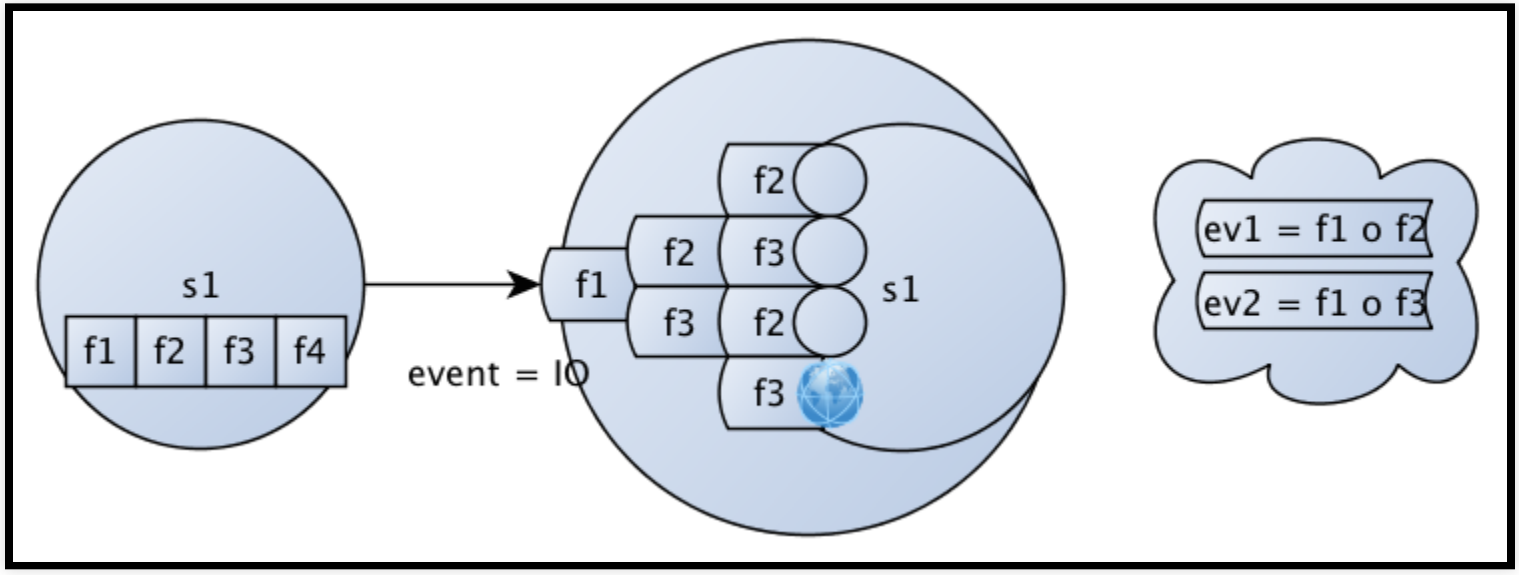
Détermination impossible

Prog fonctionnelle

`état2=f1*f2*f3(état1)` : immutabilité

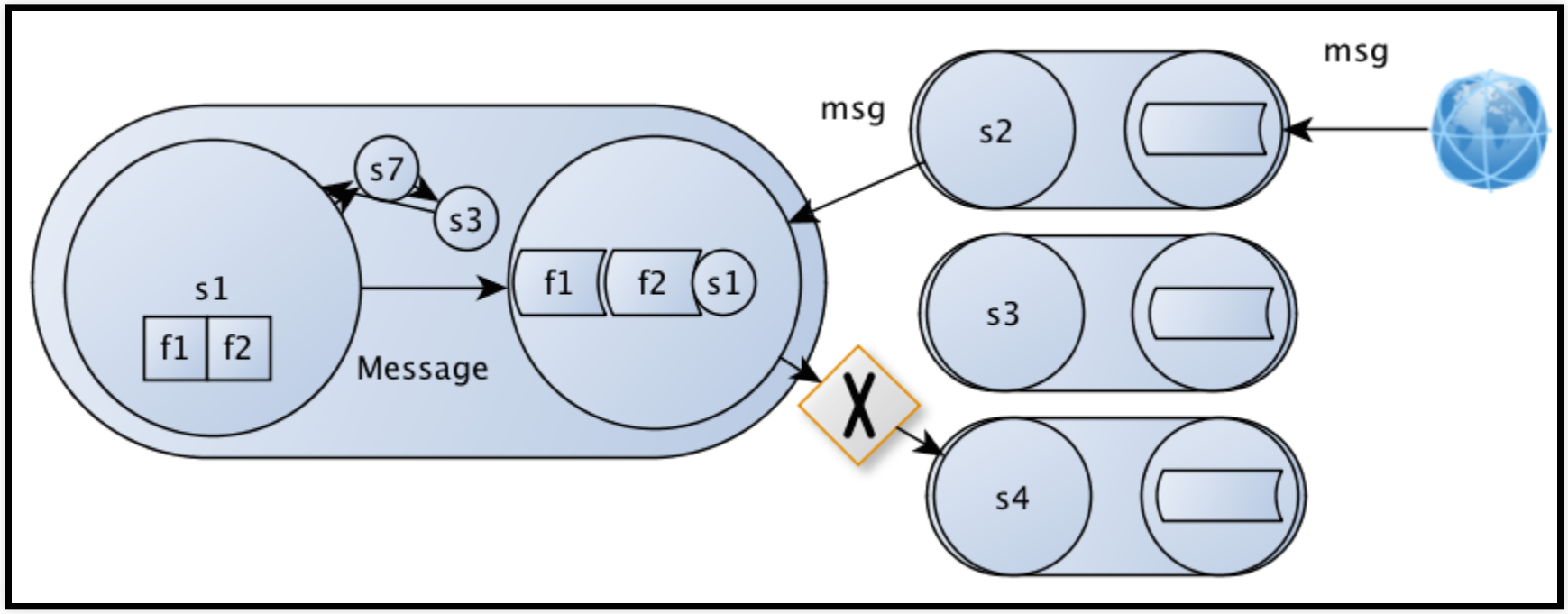
On associe un évènement à une composition de fonctions

Celui qui pense que le monde est mutable n'a jamais entendu parlé du temps



Acteurs / Microservices

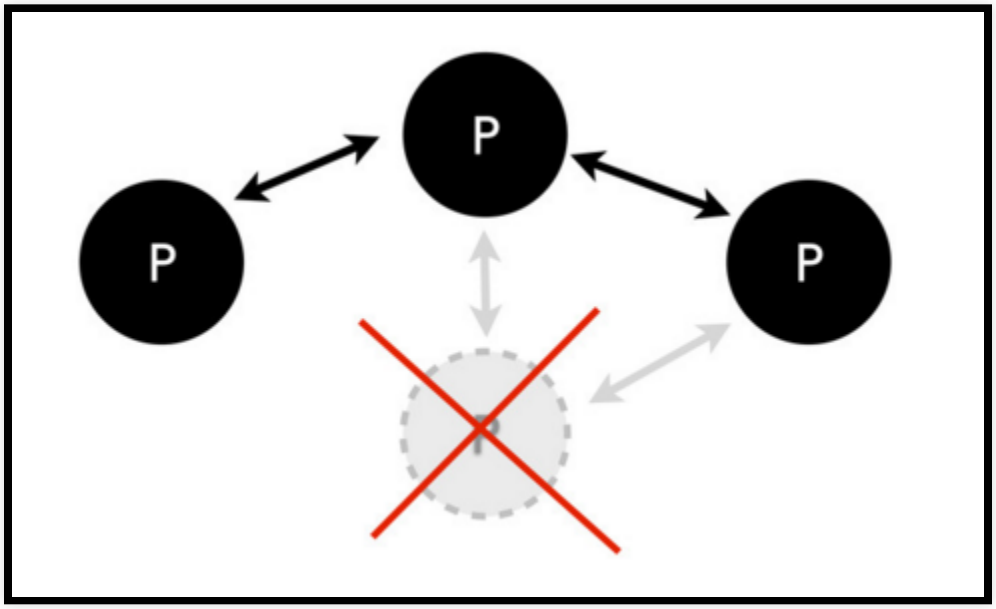
Divisé pour mieux règner, simple division des services.



Erlang=Disponibilité

L'échec est **inévitable**: conserver un état valide

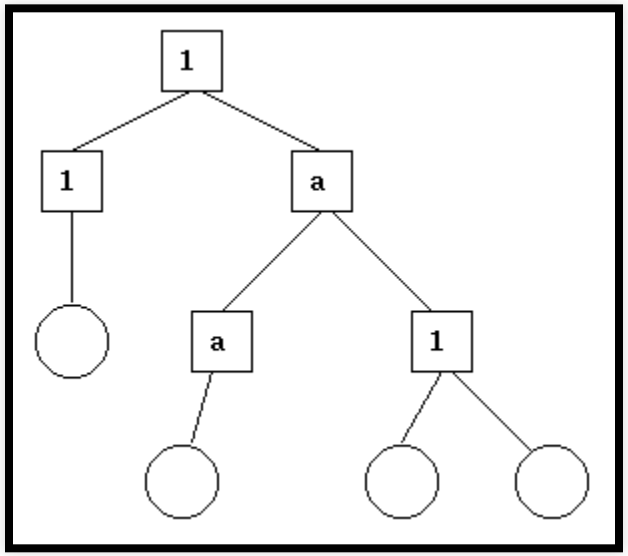
- tout acteur dans un état invalide doit mourir sans impact sur les autres: isolation et *let it crash*.
- un *link* signale la mort et permet aux autres acteurs de recréer un nouveau acteur dans un état acceptable.



Erlang OTP

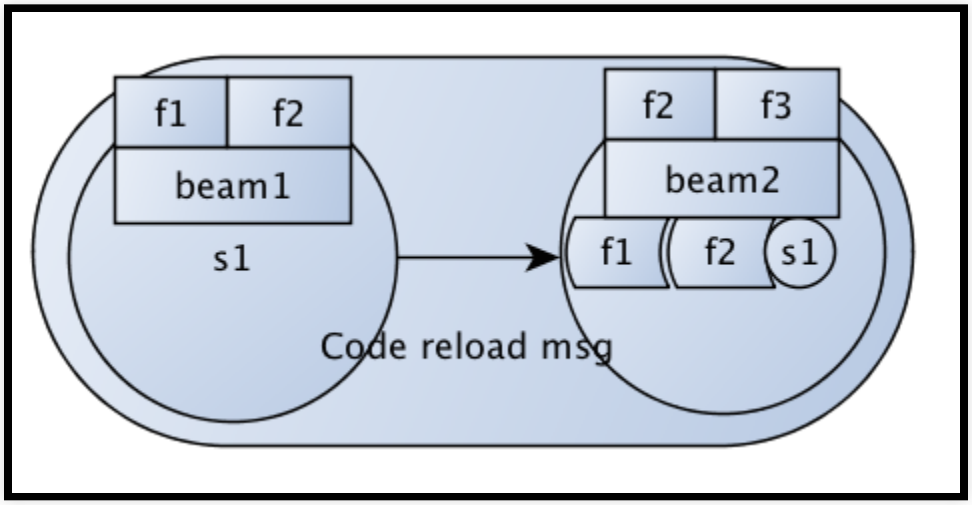
Un ensemble de normes: un framework.

Une architecture des *links* en arbre de dépendances d'états: l'arbre de supervision: reconstruction pyramidale d'un état fonctionelle.



code = état: hotload

Si on peut changer l'état d'un processus, on peut aussi changer son code.

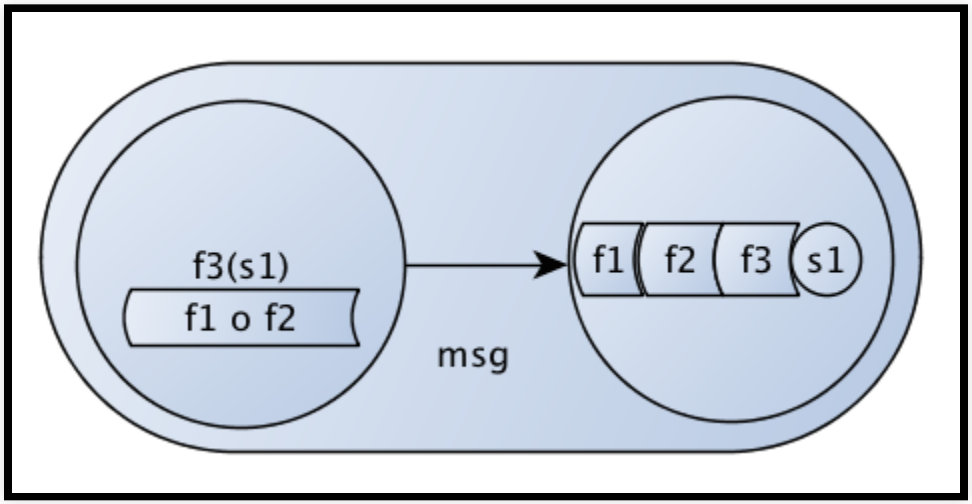


code = état: macro

La compilation est un précalcul comme un autre, la maitriser permet de créer des abstractions de langage sans impact à l'éxecution.

LA +-value d'Elixir: boîte à outils du langage

- représenter le code comme structure du langage
- manipuler cette structure à volonté pour générer le code



Aperçu d'Elixir

Les types: combinaison des types suivants:

- `%{}` dictionnaire ensemble de couple clé / valeur pour un ensemble de clé déterminé
- `[]` liste: Liste chaînée d'éléments
- `{}` tuple: taille fixe d'éléments
- `:something` atom
- `1.0 1` entiers floattant
- `<<1,10,20>>` binaire

Le Module

Ensemble de fonction c'est l'entité de manipulation du code de la VM.

```
defmodule MyModule do
  def myfun(arg) do
    :hello
  end
end
```

Le pattern matching

- Décomposer et composer les types de données
- Faire des assertions
- Contrôle des branches

```
coord = {1.0,10.0}
place = %{coord: coord, address: "9 rue du technopole", country: "fr"}
case place do
  %{coord: {lat,_}, country: "fr"} when lat < 0->
    # ouest de la France
    :france_west
  %{coord: {lat,_}} when lat < 0->
    # ouest ?
    :west
end
# la place en entrée n'est pas prévu pouvoir être à l'est
```

La Bit-Syntax

```
case Ip of
<<4::4, hlen::4, type::8, totlen::16,
      id::16, flags::3, frag::13,
      ttl::8, proto::8, hdrsum::16,
      src_ip::32,
      dst_ip::32, rest::binary>> when hlen>=5, 4*hlen=<dgramsize ->
        optlen = 4*(hlen - @ip_min_hdr_len),
        <<Opts:OptsLen/binary,Data/binary>> = RestDgram
end
```

Serveur brut

```
defmodule Server do
  def loop(account) do
    receive do
 {:credit,x}->loop(account+x)
 {:debit,x}->loop(account-x)
 {:get,sender}->
   send(sender,response(account))
   loop(account-x)
    end
  end
  def response(0), do: :void
  def response(amount) when amount > 0, do: {:ok,amount}
  def response(amount) when amount <= 0, do: {:error,amount}
end

pid = spawn_link(fn-> Server.loop(0) end)
send(pid,{:credit,2})
send(pid,{:credit,1})
```

Exemple pratique

Vente d'espace publicitaire en ligne

- Identifier les acteurs et trouver les langages communs ou pas et les traductions nécessaires:
- langage métier: SSP/DSP/Ad-Exchange/Éditeur
- langage visiteur cible: des jolies images
- langage éditeur: l'argent, les enchères, la rentabilité, les métriques financières
- langage annonceur: cible, CPM, retargeting,
- langage développeur: Elixir - java/clojure - HTML - SPecs Google Ad - :)
- langage chef de projet: des graphiques & dates
- langage exécuteur: 00011010101010111001

Contexte

Les éditeurs de site web veulent dynamiquement proposer aux enchères une impression de publicité. Les SSP (supply side plateform) sont les outils pour diffuser l'espace publicitaire sur plusieurs places de marchés. Ils vont interroger les acheteurs via leur DSP (demand site plateform) qui enchérissent. Le SSP doit trouver le meilleur enchérisseurs

Contraintes

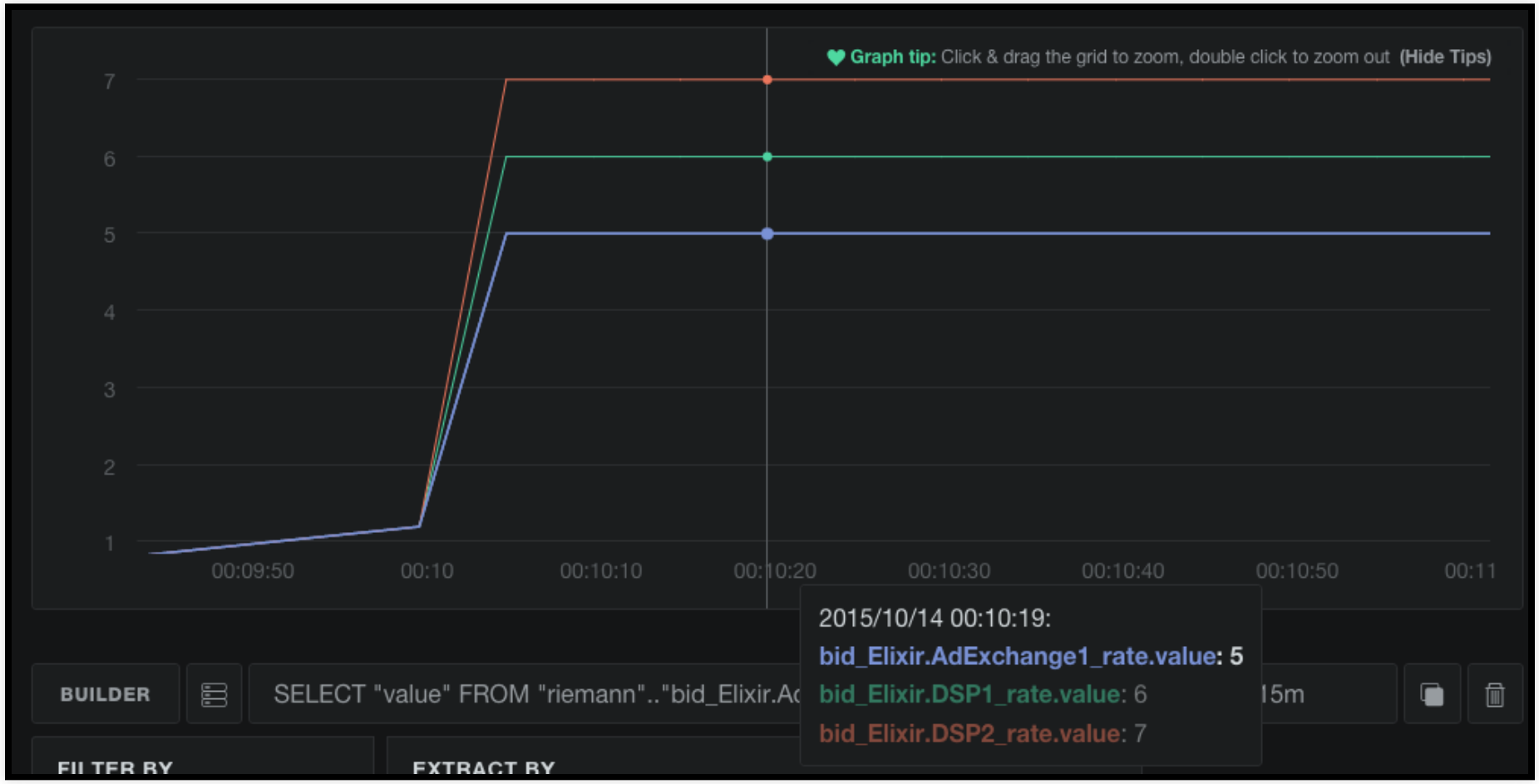
- l'interface vers le DSP peut être dans une techno propriétaire forcée
- le temps d'affichage de la publicité et donc le temps de réponse doit être borné
- les appels aux DSL doivent être parallélisés
- certains DSP n'autorise qu'un débit limité de requête
- Retenter un DSP échoué, c'est peut être le meilleur enchérisseur
- la publicité est de l'argent, donc chaque espace non rempli est une perte. Ainsi tout bug innatendu doit être gérer sans perdre des affichages. On aimerait corriger les bugs sans perdre d'impressions.

Implémentation et discussion

- Diagramme des Communications flux d'information
- Diagramme de supervision, dépendances d'états
- Code

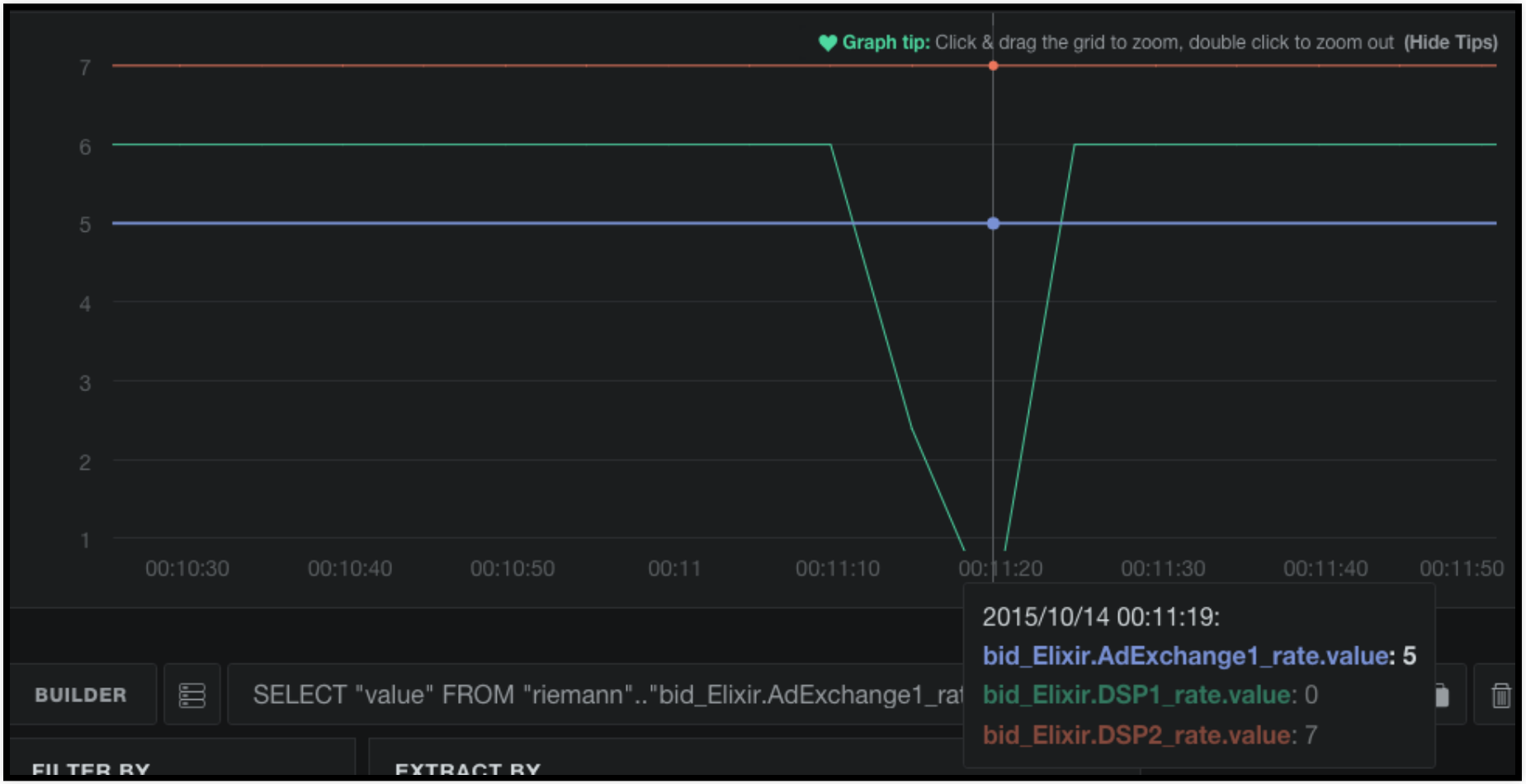
En pratique

Débit des requêtes contrôlé



En pratique

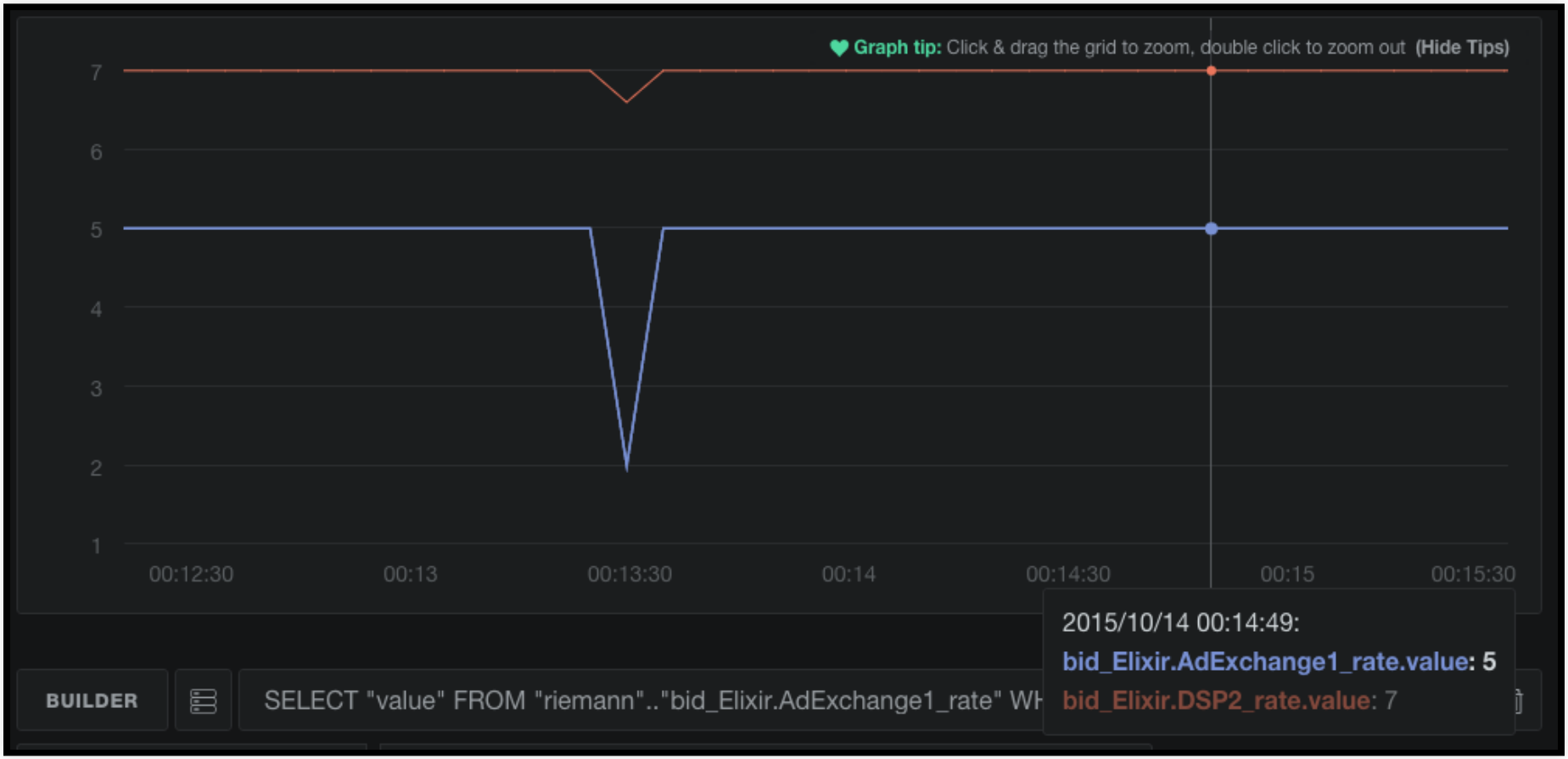
Mort de Java



En pratique

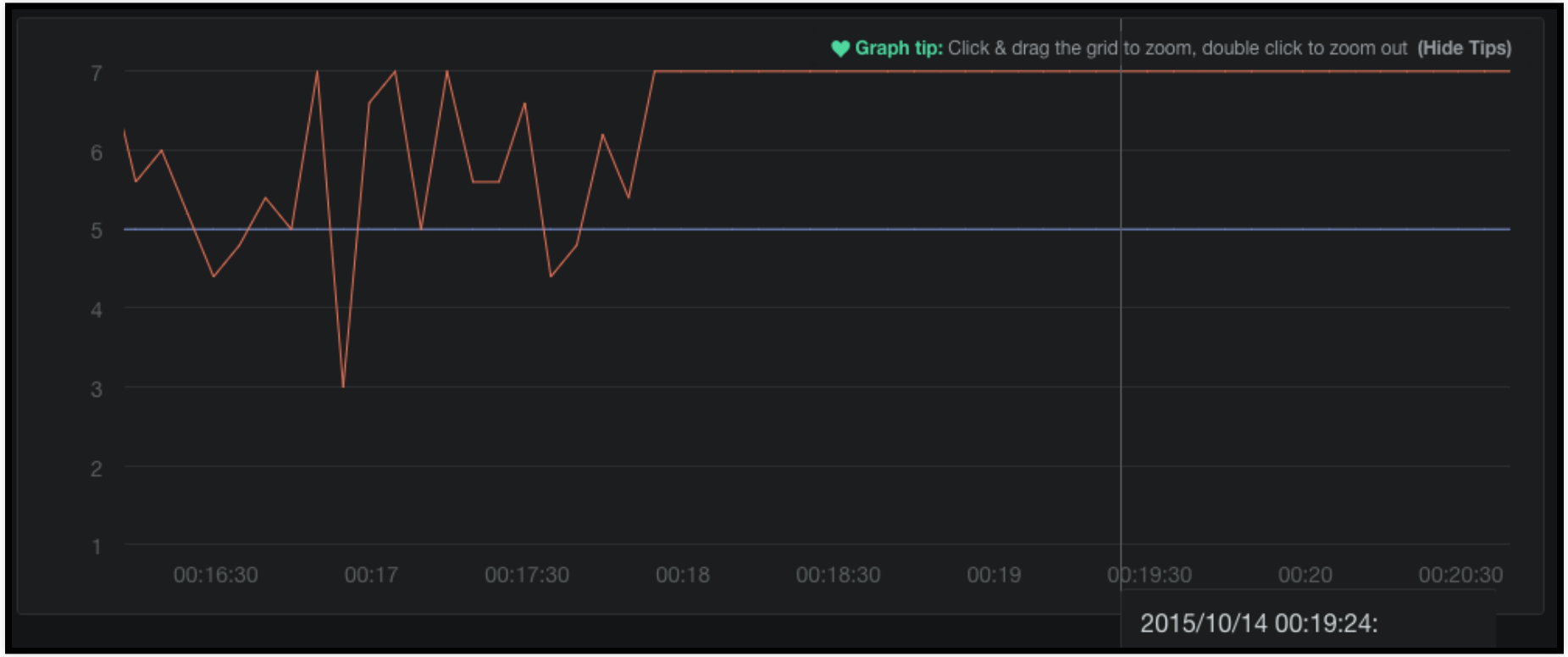
Configuration non gérée: impossible d'avoir un minimum d'enchère à 10€. L'utilisateur dynamiquement choisi `SSPDemo.Config.update min_cpm: 10`.

bidder.ex line 4


Graph tip: Click & drag the grid to zoom, double click to zoom out (Hide Tips)
7
6
5
4
3
2
1
00:12:30
00:13
00:13:30
00:14
00:14:30
00:15
00:15:30
2015/10/14 00:14:49:
bid_Elixir.AdExchange1_rate.value: 5
bid_Elixir.DSP2_rate.value: 7
BUILDER
SELECT "value" FROM "riemann".."bid_Elixir.AdExchange1_rate" WH

En pratique

On ajoute un bug innatendu aléatoire bidder.ex line 12



Des questions ?

Des réponses