IceStorm

- Service de messagerie *ICE*
- Comme tous les services, décrits par des interfaces *Slice*
- Objectifs :
  - Découpler client/serveurs
  - Communication → diffusion/captation de messages
  - Faciliter la parallélisation, la redondance

IceStorm

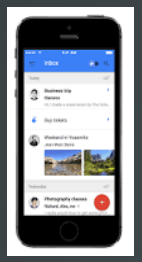
- Principes :
  - Intermédiaire entre clients/serveurs
  - La communication est gérée via un tiers, le service de messagerie
  - Relations en Publish/Subscribe

IceStorm

La messagerie

Subscribers (serveurs)

Publisher (client)



Event

Events

Sub0

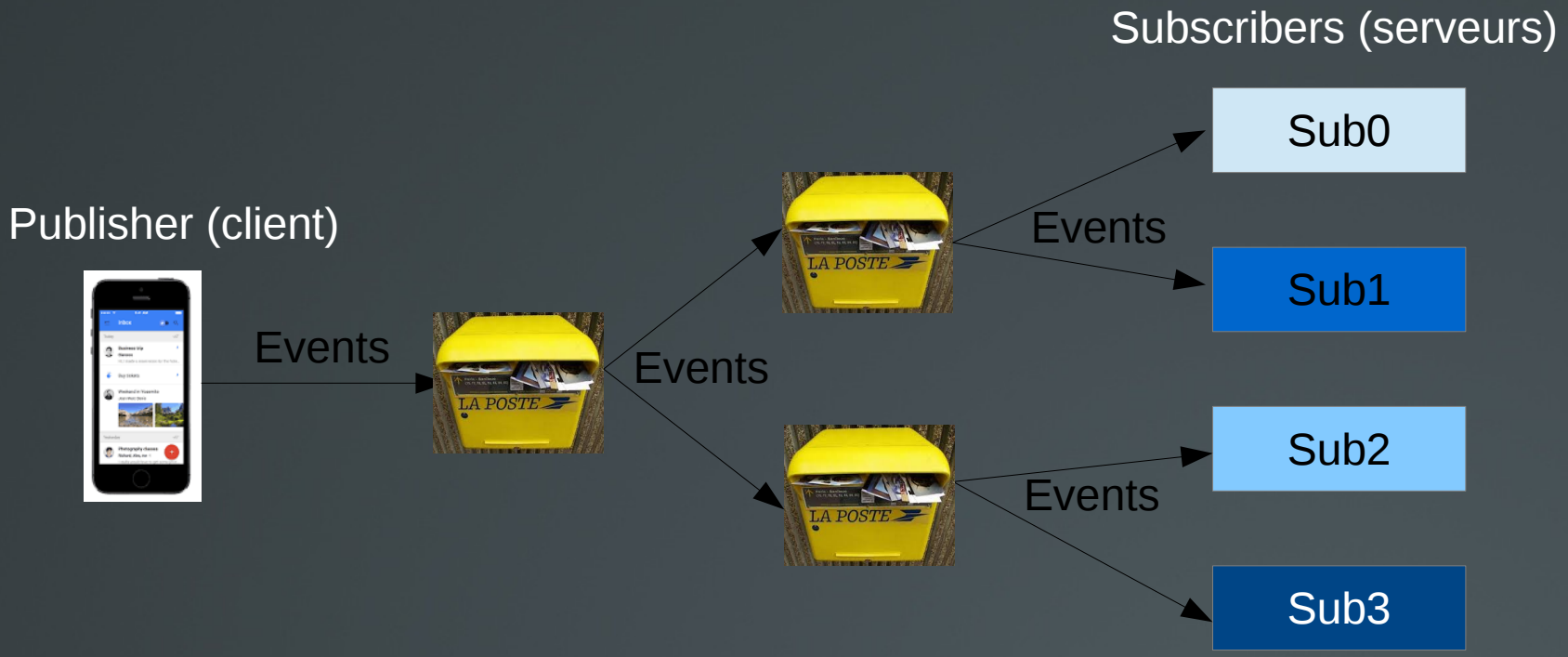
Sub1

Sub2

Sub3

IceStorm

Fédération de messageries



IceStorm

Exemple : monitoring météo

collecteur

IceStorm

Exemple : monitoring météo

collecteur

IceStorm

- Concepts de base : **messages**
  - Information structurée, fortement typée
  - Un message est représenté par une **opération**
    - Le nom de l'opération correspond au type du message
    - La valeur des paramètres est les contenu du message
    - L'invocation de la méthode est une **publication** du message
    - La publication est faite sur un proxy d'IceStorm
    - Les souscripteurs (serveurs) reçoivent le message par retour d'invocation

IceStorm

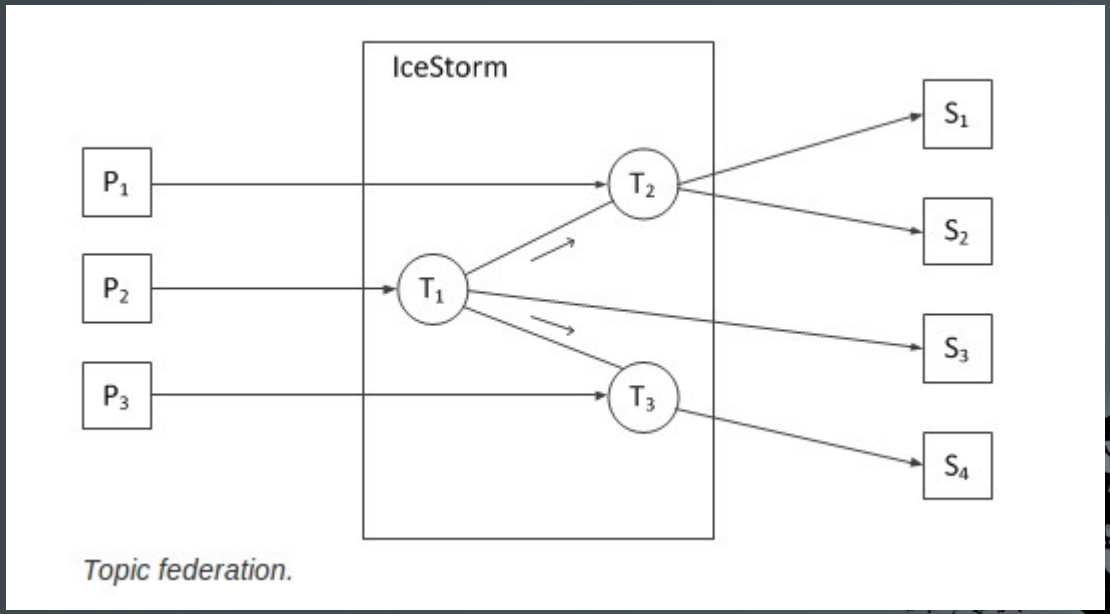
- Concepts de base : **messages**
  - Les messages sont des méthodes unidirectionnelles
    - Void en retour
    - Pas de paramètres de sortie
    - Pas d'exceptions

IceStorm

- Concepts de base
  - *Topic* (thème):
    - Sujet d'intérêt pour une application → type de message que l'application « écoute »
    - Une application manifeste son intérêt pour un type de message en s'inscrivant à un *topic*.
    - Un *topic* supporte plusieurs publiants/souscripteurs

IceStorm

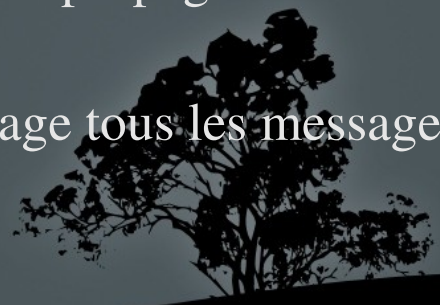
- Concepts de base
  - *Fédérations de Topic*:
    - Structure en graphes de Topics



Topic federation.

IceStorm

- Concepts de base : *Fédérations de Topic*:
  - Les graphes sont formés par création de liens entre les *topics*
  - Les messages sont propagés... sur au plus un lien, SAUF si on utilise les coûts :
    - Le message est alors propagé en fonction de son coût :
      - un message de coût nul est propagé systématiquement
      - Un lien de coût nul propage tous les messages systématiquement



IceStorm

- Concepts de base : Qualité de service
  - Chaque souscripteur peut définir sa propre QoS
  - La QoS détermine la façon dont les messages sont délivrés
  - Elle est déterminée à la souscription
  - Concrètement : des couples propriété-valeur

IceStorm

- Concepts de base : Qualité de service
  - Propriétés :
    - Reliability : ordonnancement des requêtes
    - RetryCount : détermine quand supprimer un souscripteurs inopérant

```
C++ IceStorm::QoS qos;
qos["reliability"] = "ordered";
topic->subscribeAndGetPublisher(qos, proxy->ice_twoway());
```

IceStorm

- Concepts de base : réplication
  - Objectif : haute disponibilité
  - Principe :
    - Modèle maître-esclave
      - Recours aux répliques lorsque le maître est défaillant
      - Les répliques sont priorisées
      - Les états des répliques :
        - Inactive, élection, ré-organisation, normal

IceStorm

- Concepts de base : réplication
  - Objectif : haute disponibilité
  - Principe :
    - Modèle maître-esclave
      - Recours aux répliques lorsque le maître est défaillant
      - Les répliques sont priorisées
      - Les états des répliques :
        - Inactive, élection, ré-organisation, normal

IceStorm

- Administration :
  - En ligne de commande : icestormadmin
    - Gestion des topics, de liens, des états, des répliques, etc.
- Démarrer et configurer le service :
  - Démarrer : par icebox
    - $ icebox --Ice.Config=config

IceBox

- Application serveur
  - Permet aux applications C++, java, .NET d'être hébergés comme des services dans un même espace d'adressage
  - Optimisations liées à la colocalisation
  - Interface d'administration commune
- Usage: iceboxadmin [opts] [command : stop| start|shutdown]

IceStorm : exemple

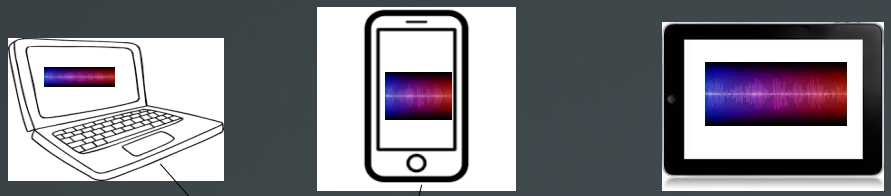
- Un exemple d'architecture distribuée basée sur la messagerie : un player vidéo distribué

IceStorm : exemple

- Un exemple d'architecture distribuée basée sur la messagerie : un player vidéo distribué

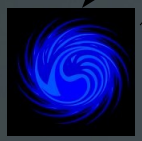
Application

Serveurs de flux multimédia : Ice



Client Android

Reconnaisseur de parole
Techno : Au choix
Entrée signal (tableau de float)
Sortie : chaîne de caractères



Analyseur de requête :
Techno : au choix
Entrée : texte en langage naturel
Sortie : commande <action> <objet>

Application

- Scénario applicatif :
  - Accès en langage naturel à la bibliothèque distribuée
  - Via un smart phone *par exemple*
  - Bibliothèque éventuellement partagée

Application

- Difficultés :
  - Techniques :
    - Panacher les technologies
    - Utiliser des serveurs existants
    - Mise à jour à chaud : messages !
  - Méthodologiques :
    - Reconnaissance de la parole
    - Analyse des contenus
    - Dialogue ?