Jamus : une plate-forme d'accueil sécurisée pour le code mobile

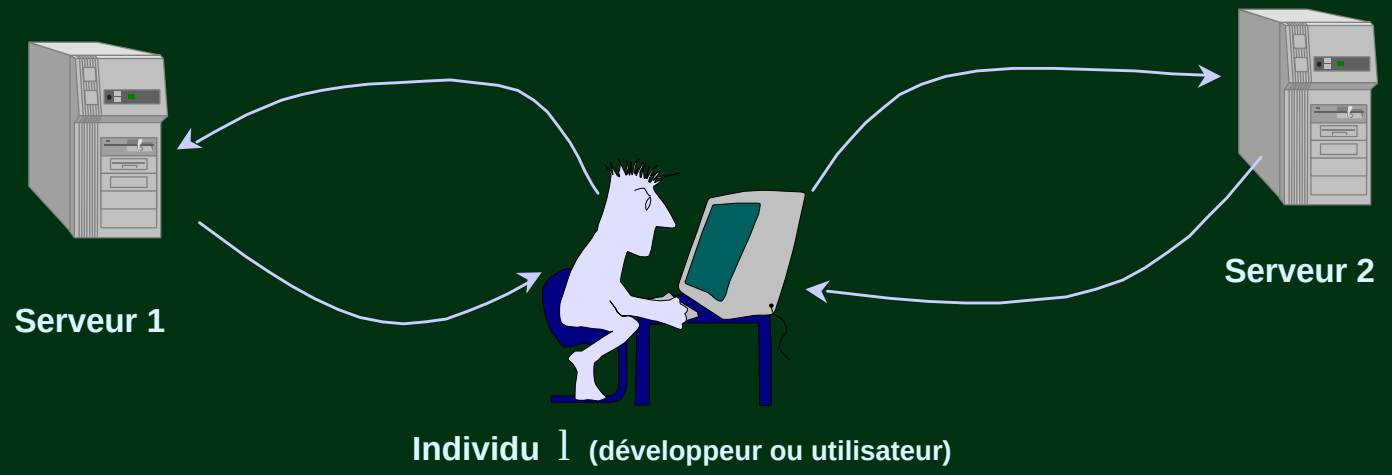
Nicolas Le Sommer
Nicolas.LeSommer@univ-ubs.fr

Plan

- Problématique
- Approche du problème
- Jamus
  - Concept de profils d'utilisation des ressources
  - Concept de courtier de ressources
  - Principe de supervision des codes mobiles
  - Restrictions actuelles
- Perspectives
- Conclusion

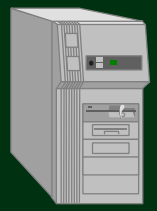
Problématique

- Problèmes posés par le déploiement de composants logiciels déployés via un mécanisme de code mobile :

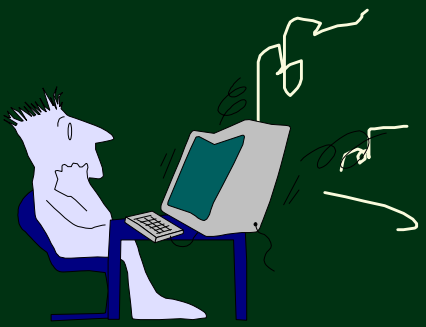


Problématique

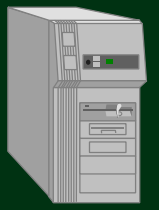
- Problèmes posés par le déploiement de composants logiciels déployés via un mécanisme de code mobile :



Serveur 1



Individu 1 (développeur ou utilisateur)



Serveur 2

Problématique

- Problèmes liés à la sécurité de l'hôte d'accueil
  - Destruction de données cruciales
  - Consommation abusive des ressources

- Problèmes portant sur la qualité de service demandée par les composants logiciels vis-à-vis de la disponibilité des ressources nécessaires à leur exécution

Approche du problème

- Prise en compte de certaines contraintes non-fonctionnelles des composants logiciels
  - Celles portant sur les ressources nécessaires à leur exécution

Approche du problème

- Connaître les besoins propres de chaque composant logiciel serait un « plus »...

  - Pour qu'un composant logiciel puisse négocier l'accès aux ressources offertes par la plate-forme sur laquelle il s'exécute (contractualisation et QoS en retour).
  - Pour que la plate-forme puisse contrôler l'utilisation des ressources (prévention du déni de service)
  - Pour aider à choisir parmi plusieurs composants logiciels
  - etc.

La plate-forme Jamus

- Jamus (Java Accommodation of Mobile Untrusted Software)
- Plate-forme d'accueil pour « composants logiciels mobiles » capables d'exprimer leurs besoins vis-à-vis des ressources
- Met en œuvre une approche contractuelle de l'accès aux ressources
- Offre de la qualité de service aux composants hébergés vis-à-vis de la disponibilité des ressources

Présentation de Jamus

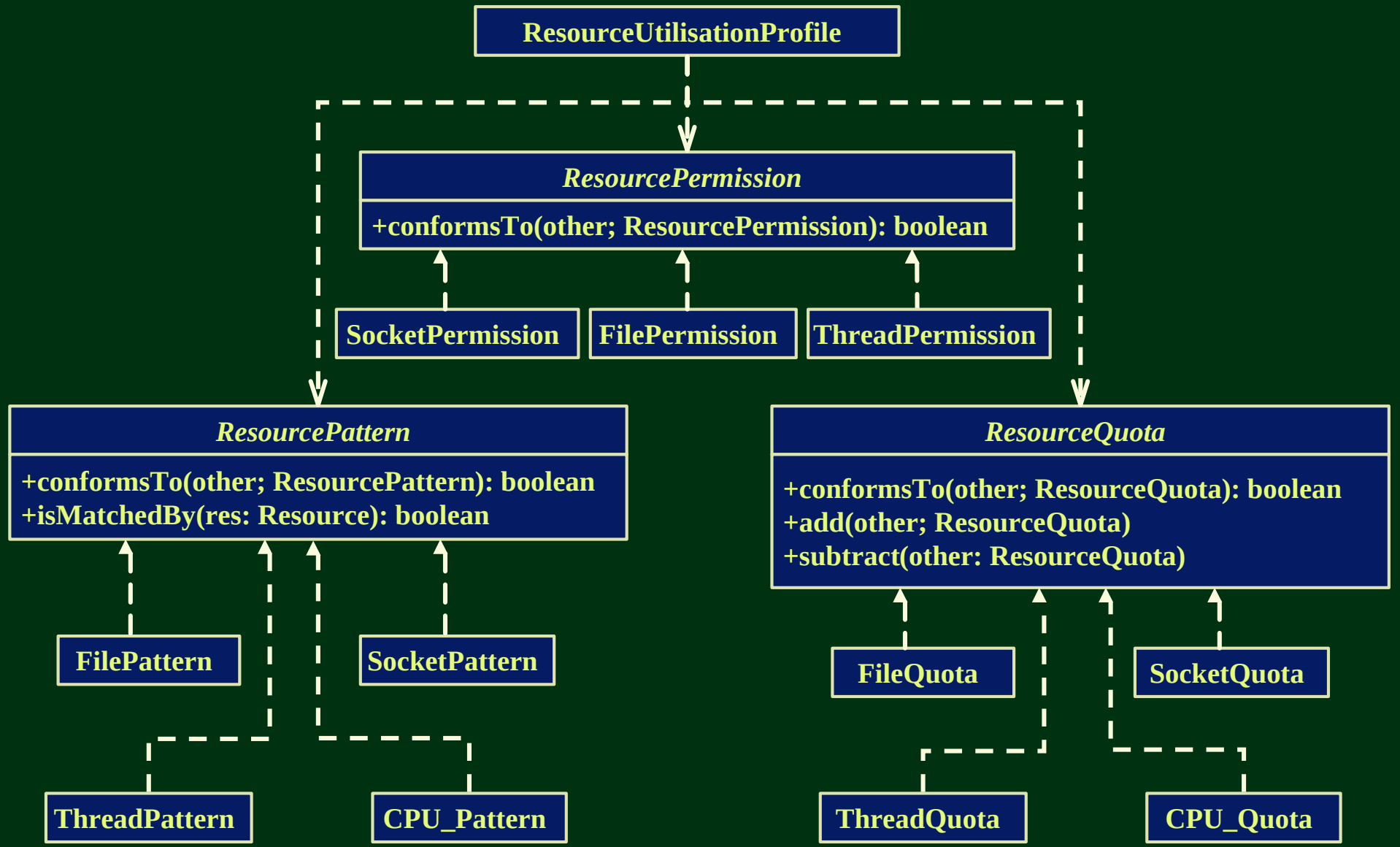
- Repose sur les éléments suivants :
  - Concepts de profil d'utilisation des ressources
  - Concept de courtier de ressources
  - Principe de supervision des composants logiciels
- S'appuie sur l'environnement RAJE (Resource-Aware Java Environment)
  - Classes réifiant certaines ressources système (CPU, mémoire, réseau,...)
  - Extension des classes de l'API Java afin de permettre l'observation et le contrôle des ressources conceptuelles (Sockets, Threads, File, ...)
  - Basé sur Kaffe-1.0.6 + Linux

Profil d'utilisation des ressources

- Rôle :
  - Définition des contraintes portant sur les ressources offertes par la plate-forme Jamus
  - Description des besoins des composants

- Objets Java possédant trois attributs :
  - pattern
  - permission
  - quota

Profils d'utilisation des ressources



Profils d'utilisation des ressources

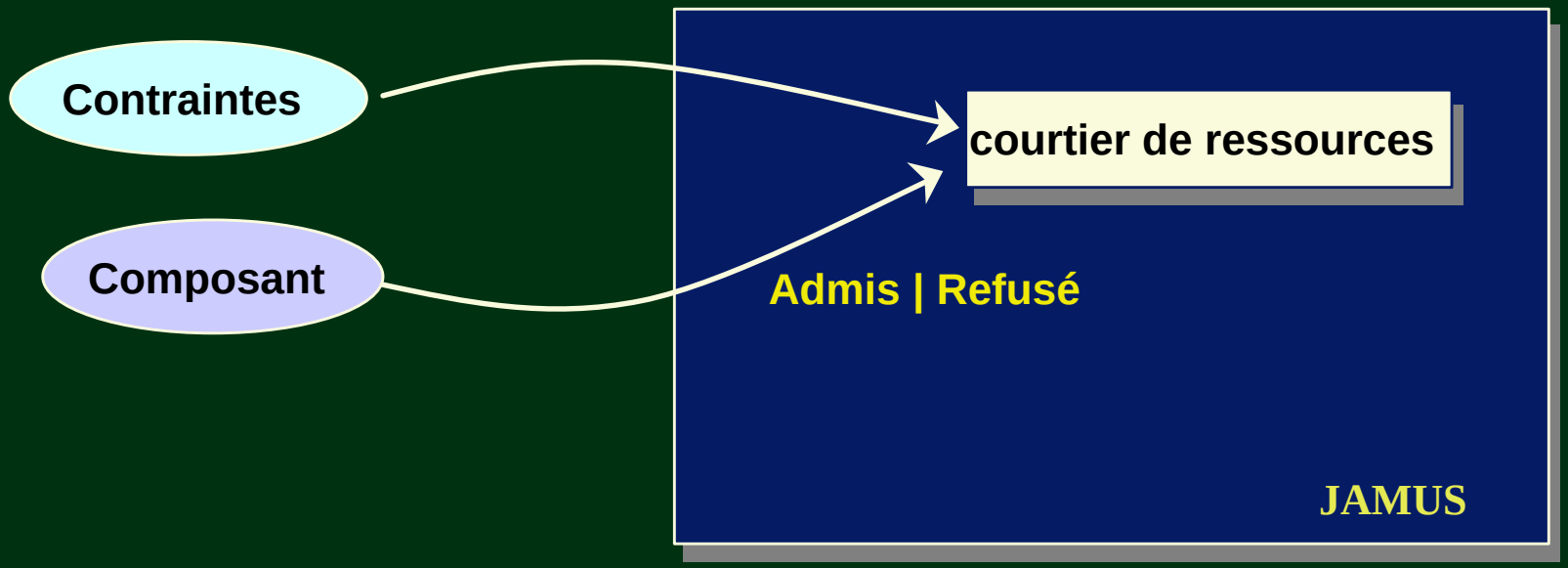
```
public class MyProgram{
public static Set getResourceRequirements(String[] args) {
Set requirements = new HashSet();
int Ko = 1024; int Mo = 1024*1024;

// Profil d'utilisation sélectif ne concernant que les connexions établies
// vers des hôtes du domaine 195.83.160/24:
//quota de 512 Koctets en émission, 128 Koctets en réception.
requirements.add(new ResourceUsageProfile(new SocketPattern("195.83.160/24"),
                                          new SocketPermission(SocketPermission.all),
                                          new SocketQuota(512*Ko, 128*Ko)));

 // Profil d'utilisation global s'appliquant  sur /tmp
 // quota de 1 Mo en écriture, 2 Mo en lecture.
requirements.add(new ResourceUsageProfile(new FilePattern("/tmp"),
                                          new FilePermission(FilePermission.all),
                                          new FileQuota(1*Mo, 2*Mo)));
return requirements;  }
 public static void main(String[] args) { . . .  }
}
```

Courtier de ressources

- Rôles :
  - Contrôle d'admission
  - Allocation et libération des ressources



Courtier de ressources

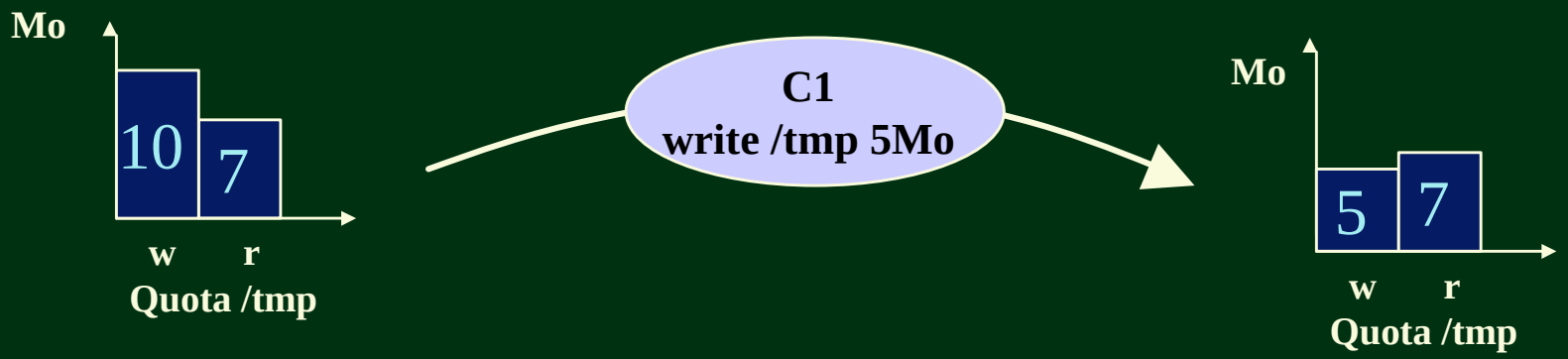
- Composant logiciel admis = tous les besoins demandés par le composant peuvent être satisfaits

- Allocation et libération des ressources :
  - Politique de gestion des ressources = réservation

Courtier de ressources

- Composant logiciel admis = tous les besoins demandés par le composant peuvent être satisfaits

- Allocation et libération des ressources :
  - Politique de gestion des ressources = réservation

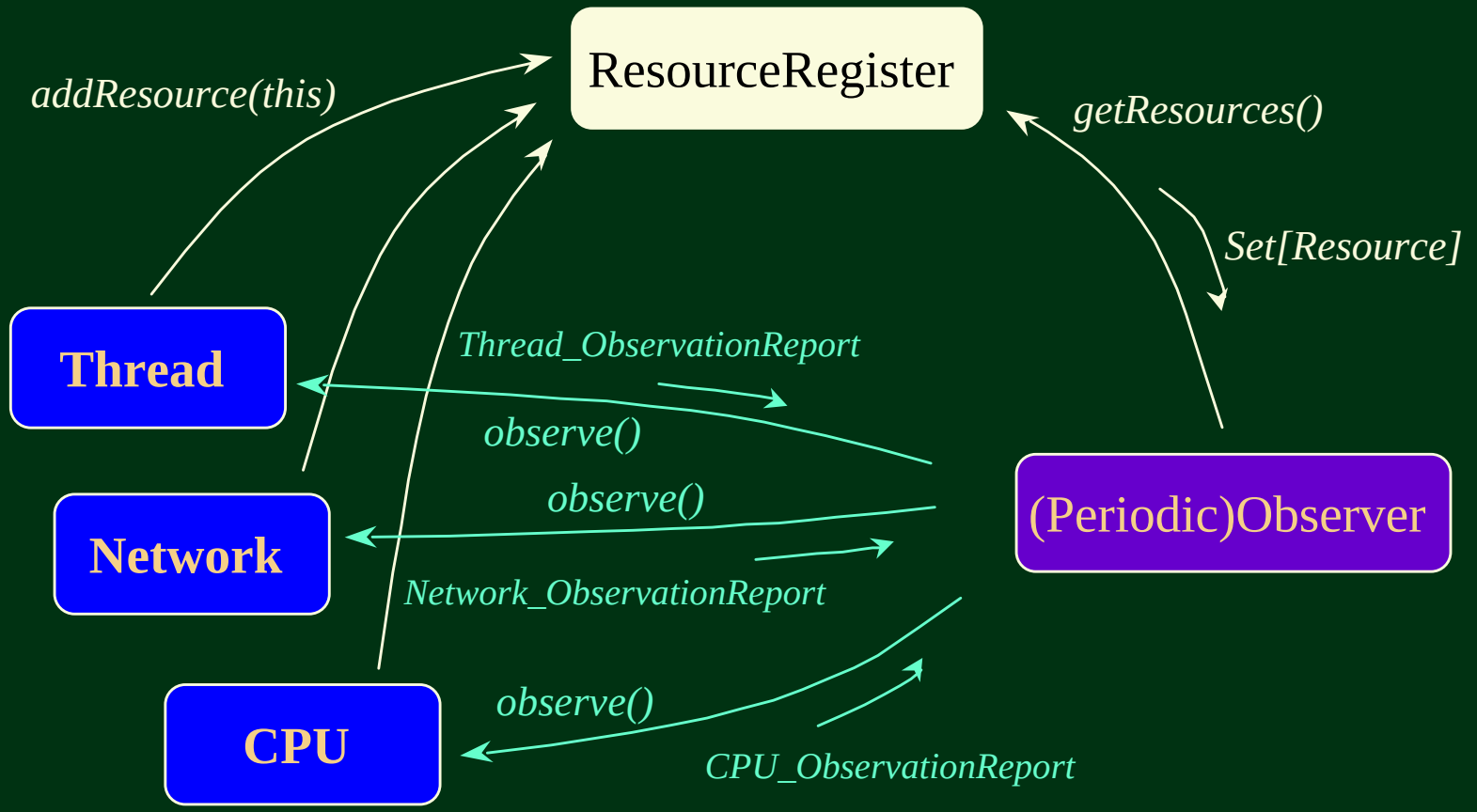


Principe de supervision

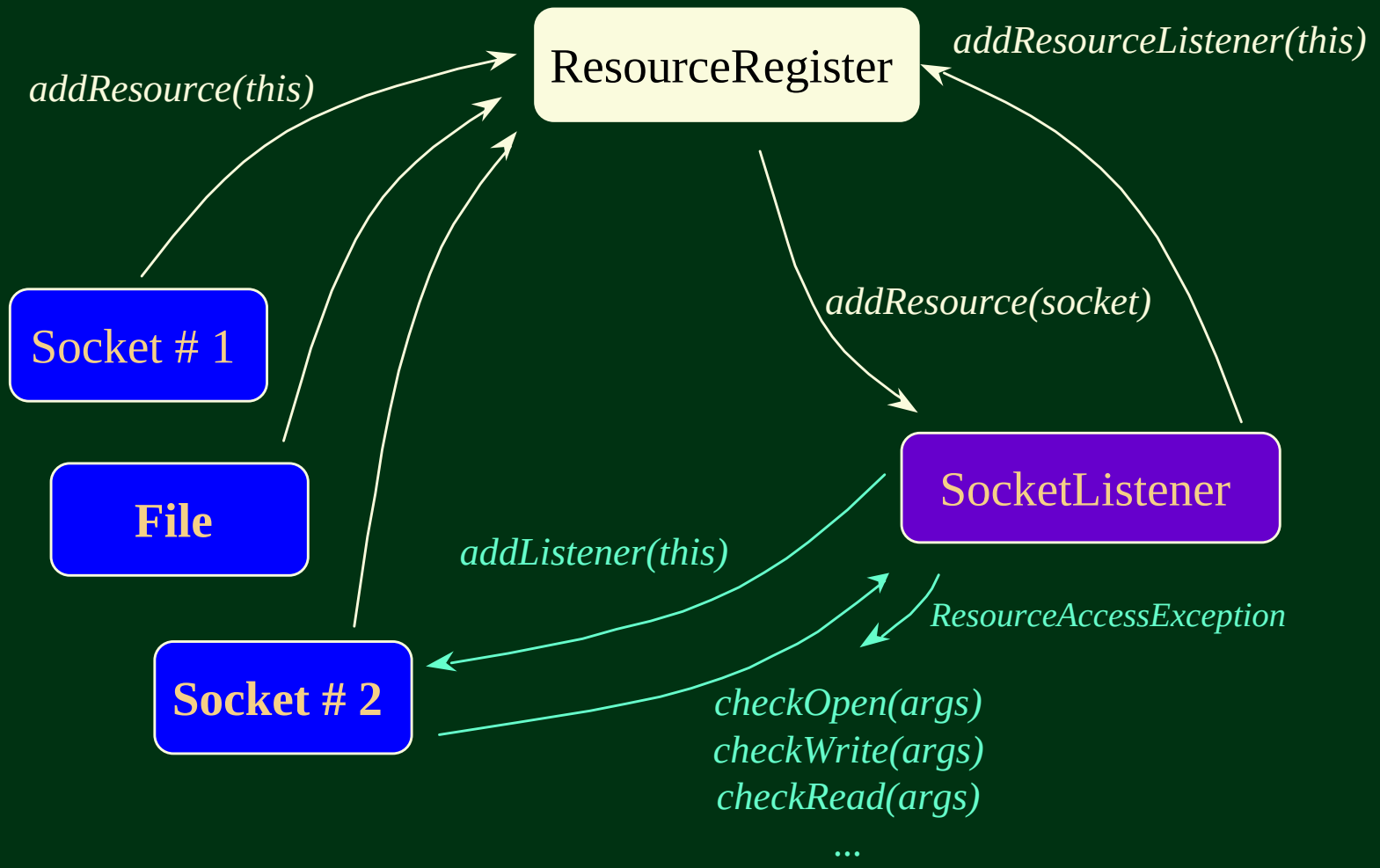
- Supervision des codes mobiles
  - Vérification du respect des contrats définis lors du contrôle d'admission

- Deux modes de supervisions
  - synchrone
    - Adapté pour les ressources conceptuelles (Socket, File, DatagramSocket,..)
    - Interception des appels aux ressources
  - asynchrone
    - Adapté pour les ressources systèmes (CPU, mémoire, …) et pour les ressources conceptuelles (Socket, File, DatagramSocket,..)
    - Observation périodique de l'état des ressources

Supervision : observation asynchrone



Supervision : observation synchrone



Restrictions actuelles

- CPU, réseau et disque sont les seules ressources traitées
- Tous les besoins doivent être connus avant l'exécution
- Programmes d'application (main)

Perspectives

- Pour Jamus
  - Traiter la ressource mémoire
  - Intégrer un mécanisme de négociation dynamique
  - Envisager des politiques de gestion des ressources autres que la réservation
  - etc.

- Autres
  - Définition d'un formalisme d'expression des profils d'utilisation des ressources plus « élégant » que celui des objets Java (XML, ...)
  - Intégrer ces aspects non-fonctionnels dans les phases de test et de validation des composants logiciels
  - etc.

Conclusion

- Plate-forme prototype permettant la contractualisation des ressources avec qualité de service en retour
- Plate-forme permettant de « contrôler » l'utilisation des ressources (prévention du déni de service)
- Faible coût de l'observation (premières mesures)
  - Ex : Serveur JFTP du MIT (transfert de 20 Mo en Fast Ethernet 100 Mbps Full Duplex)
    - sans observation (Kaffe standard) : 8,5 Mo/s (68 Mbps)
    - avec observation (Jamus) : 8,4 Mo/s (67,2 Mbps)
    - surcoût CPU négligeable (~ 0 ms)
  - Serveur WU-FTPD : 8,5 Mo/s (68 Mbps)