الجمهورية العربية السورية
جامعة دمشق
كلية الهندسة المعلوماتية
السنة الخامسة

مشروع النظم الموزعة
نظام مشاركة ملفات موزع

إعداد الطلاب:

بهاء الدين النقطه
عبد الرحمن خرزوم
عبد الله رحمون
عبد الله موسى
علاء شيباني

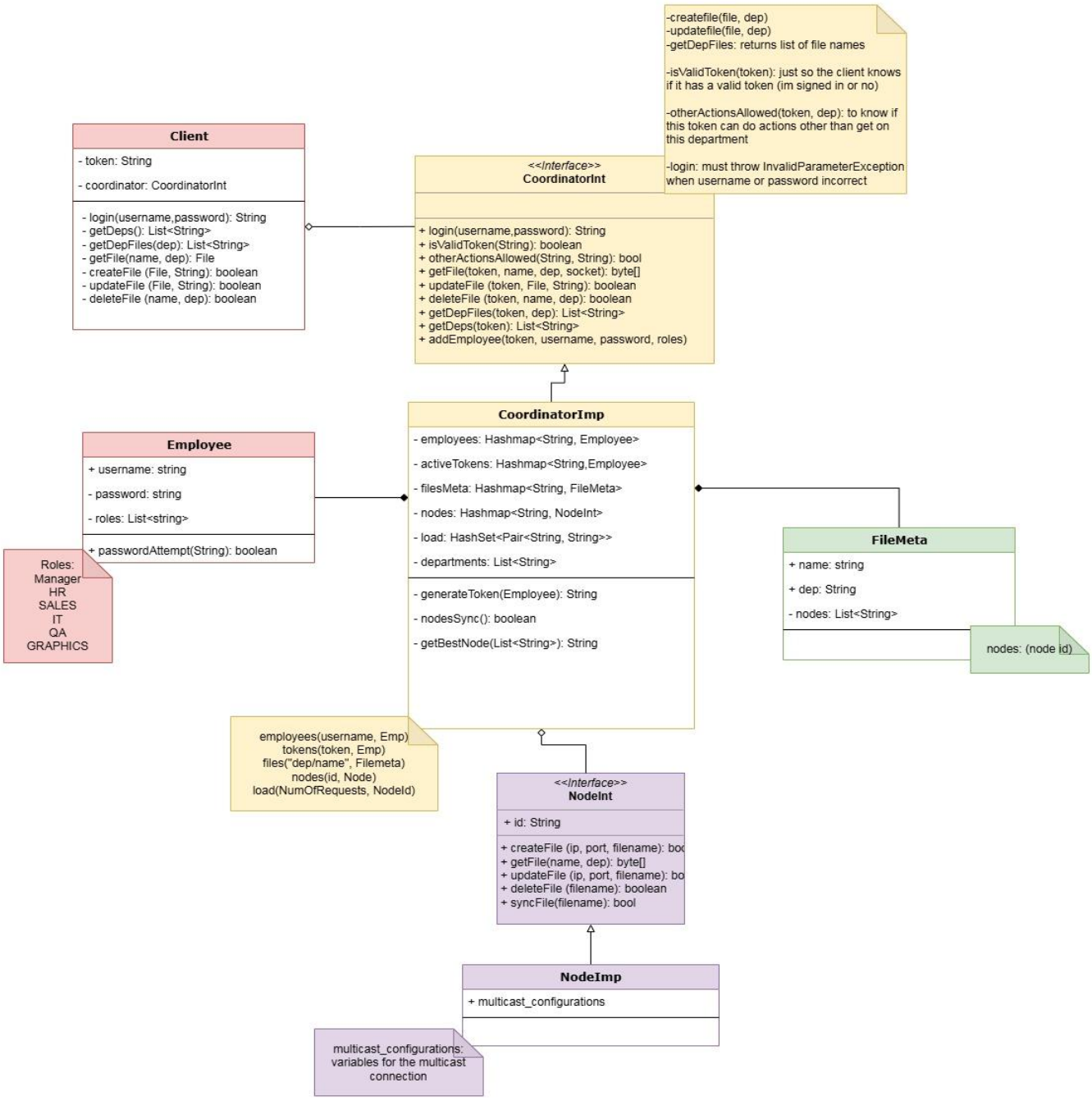
الفهرس:

المقدمة

يتكون النظام من 3 مكونات وصفوف أساسية:

1. Coordinator: المسؤول عن تنظيم الاتصالات بين العقد والمستخدمين، وضمان الصلاحيات والمعلومات الأساسية عن الملفات الموجودة في النظام وتحقيق الLoad Balancing.
2. Node: العقد التي تقوم بتخزين الملفات فعلياً، يقوم الCoordinator بالتواصل معها عن طريق الRMI والواجهة الخاص بهاNodeInt ، وتقوم بتبادل الملفات مع المستخدمين مباشرة باستعمال الSockets.
3. Client: الصف الذي يتعامل مع المستخدم، ويقدم له واجهة التعامل مع النظام والقيام بالمهام المتعددة المخولة له، ويتخاطب مع الCoordinator عن طريق الRMI عبر الواجهة CoordinatorInt.

وهذا هو الClass Diagram الذي يعبر عن هيكلية وتصميم النظام داخلياً (بعض أسماء ومتغيرات التوابع تم تغييرها أثناء بناء النظام ولكن الهيكلية بقيت كما هي)

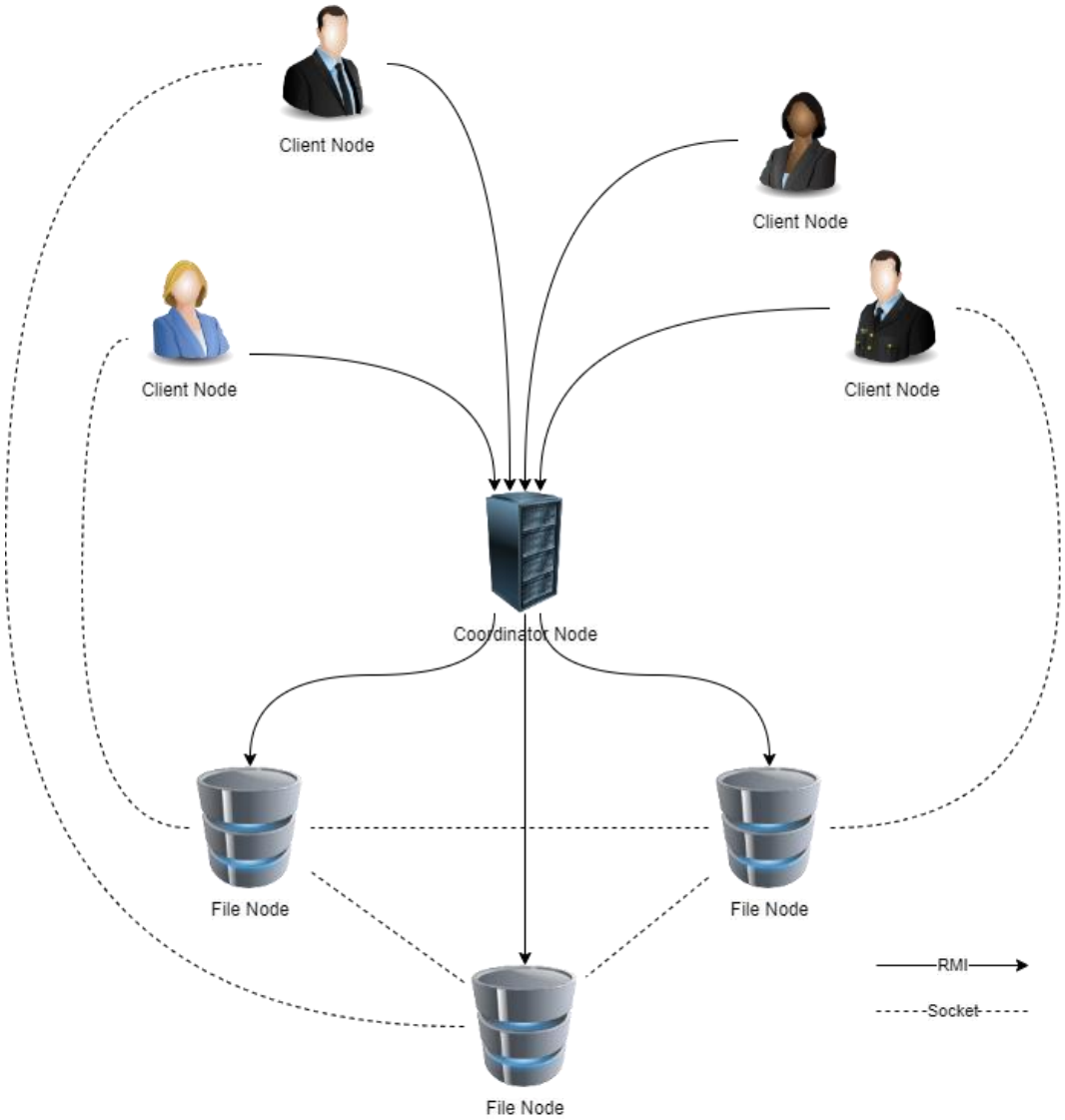


أهم النقاط في تصميم النظام:

نعتبر أن كل مكون من مكونات النظام الثلاثة سيعمل على جهاز خاص به، حيث أن المستخدمين سيقومون بتشغيل instance من الصف Client كل مستخدم على جهازه، أما الCoordinator فسيتم تشغيله من الصف CoordinatorImp على جهاز مستقل، وأخيراً سيتم تشغيل العدد المراد من العقد عن طريق انشاء instances من الصف NodeImp، كل instance على جهاز منفصل، وهكذا.

تتم عملية التواصل بالشكل التالي:

الClient يستدعي توابع الCoordinatorInt عن طريق الRMI، والCoordinatorImp يستدعي توابع الNodeInt عن طريق الRMI أيضاً، وأماعمليات نقل وتحميل الملفات النهائية تتم بين الClient والNode مباشرة باستخدام Socket، وذلك لتوفير الضغط على الCoordinator وعدم إضاعة الوقت في نقل الملف مرتين عبر الشبكة، وإنما نقله مرة واحدة



العناصر الأساسية في صف الCoordinator والتي سنعتمد عليها في مجمل عمليات النظام:

- Load Hashmap: تخزن كمية الضغط على كل عقدة، بحيث نستفيد منه في علمية الload balance، حيث يربط كل عقدة برقم يعبر عن عدد العمليات التي تتم عليها في الوقت الحالي.
- Nodes hashmap: تخزن العقد الموجودة على النظام حتى يتعامل الCoordinator معها.
- FilesMeta hashmap: تخزن أسماء الملفات الموجودة في النظام، وتربط كل واحد بكائن من صف FileMeta، يخزن هذا الكائن المعلومات الأساسية عن الملف كاسمه والقسم التابع له، والأهم أنه يحوي مصفوفة تخزن بداخلها ID جميع العقدة التي تخزن هذا الملف، وتساعدنا هذه الHashmap على معرفة الملفات الموجودة لدينا في النظام بالإضافة لمعرفة العقد التي تخزن هذا الملف بسرعة حينما نريد استرداده.
- FilesStatus hashmap: تخزن أسماء الملفات التي حصل lock عليها (سواء لان أحداً ما يقرؤها أو يعدل عليها)، حيث تفيد في الحماية في اثناء القراءة أو التحديث المتزامنين.
- مصوفة Employees: تربط اسم الموظف (المستخدم) مع الobject الخاص به، نحتاجها للlogin.
- مصفوفة Tokens: تربط الtoken مع الموظف التابعة له، فنعرف من هو وصلاحياته عند وصول الطلب منه.

```
static final HashMap<String, Integer> load = new HashMap<>();  10 usages
static final HashMap<String, NodeInt> nodes = new HashMap<>();  8 usages
static final HashMap<String, FileMeta> filesMeta = new HashMap<>();  16 usages
private static final HashMap<String, Character> filesStatus = new HashMap<>()
private final List<String> departments;  2 usages
private final HashMap<String, Employee> employees;  5 usages
private final HashMap<String, Employee> tokens;  4 usages
```

- **اختيار العقدة الأفضل :**يختار المنسق العقدة التي تحمل أقل عبء (load) لتنفيذ عمليات إنشاء، جلب، تحديث، وحذف الملفات.
- **التحكم بالتزامن :(Locks)** يستخدم المنسق خريطة filesStatus لتتبع حالة الملفات (قراءة 'R' أو كتابة 'W') ويقوم بفحص الوصول (checkRWAccess) قبل السماح بالعمليات لمنع تضارب القراءة/الكتابة المتزامنة لنفس الملف.

- **التحكم بالوصول والصلاحيات:(Access Control) :**

يتحقق الCoordinator بشكل مستمر من صلاحية التوكن (isValidToken) ومن صلاحيات المستخدم (هل هو مدير isManager)، هل لديه صلاحية للقسم otherActionsAllowed قبل السماح بأي عملية كتابة أو حذف ملفات.

- **التسامح مع الأخطاء (Fault Tolerance)** : لضمان استمرارية تنفيذ الطلبات في حال خروج عقدة أو أكثر عن الخدمة
  1. نسخ البيانات:(Data Replication) بما أن الملفات يتم حفظها بالفعل في عدة عُقد وتتم مزامنتها يوميًا.
  2. اكتشاف الأعطال في العُقد (Ping): يقوم Coordinator بإرسال طلبات اختبار بسيطة (ping) إلى العُقد بشكل دوري للتحقق من نشاطها.
  3. آلية تكرار المحاولة عبر العُقد:
     - عندما يطلب المستخدم عملية معينة، يقوم النظام بالمرور على العُقد المتاحة من قائمة العقد الأفضل بالتتابع.
     - إذا فشلت العقدة الأولى في تنفيذ الطلب ، ينتقل النظام تلقائيًا لتجربة العقدة التالية في القائمة.

شرح العمليات الأساسية في النظام:

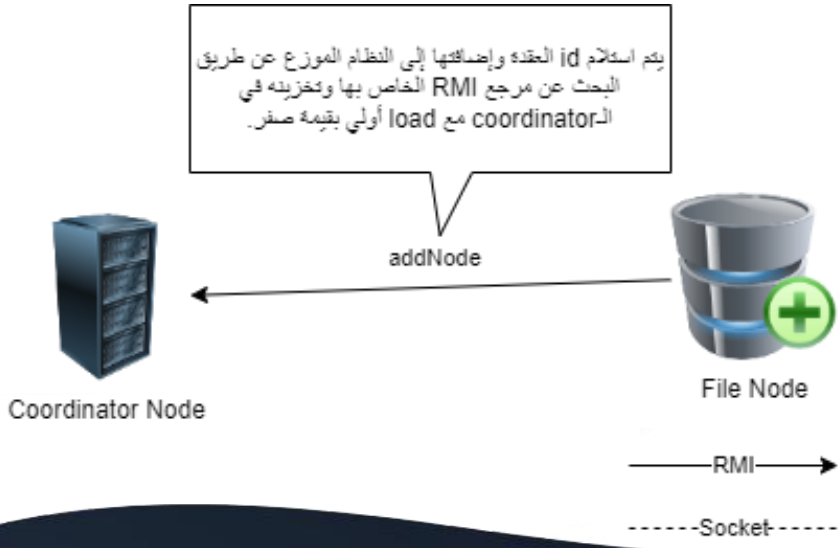
أولاً: تهيئة الاتصال بين المكونات:

يقوم الCoordinator بتشغيل الregistry وتعريف نفسه كRemote object عبر الRMI و rebind:

```
public static void main(String[] args) {  BAHAA-THE-KING +2
    try {

        CoordinatorImp coordinator = new CoordinatorImp();
        Employee manager = new Employee( username: "man",  password: "123", List.of( e1: "MANAGER"));
        coordinator.employees.put("man", manager);

        LocateRegistry.createRegistry( port: 5000);
        Naming.rebind( name: "rmi://localhost:5000/coordinator", coordinator);


        System.out.println("coordinator is running");
    } catch (Exception e) {
        System.out.println(e);
    }
```

ثم عند تشغيل عقدة ما، تعرف نفسها كRemote Object ثم تتصل مع الCoordinator لتخبره أنها جاهزة للعمل حتى يضيفها لمصفوفة الnodes لديه:

```
public static void main(String[] args) throws RemoteException {  AbDulla Musa +1
    try {
        NodeImp node = new NodeImp( nodeId: "node_1");

        Naming.rebind( name: "rmi://localhost:5000/node_1", node);
        CoordinatorInt coordinator =(CoordinatorInt) Naming.lookup( name: "rmi://localhost:5000/coordinator");
        coordinator.addNode(node.id);
        System.out.println(node.id + "is running");
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
```

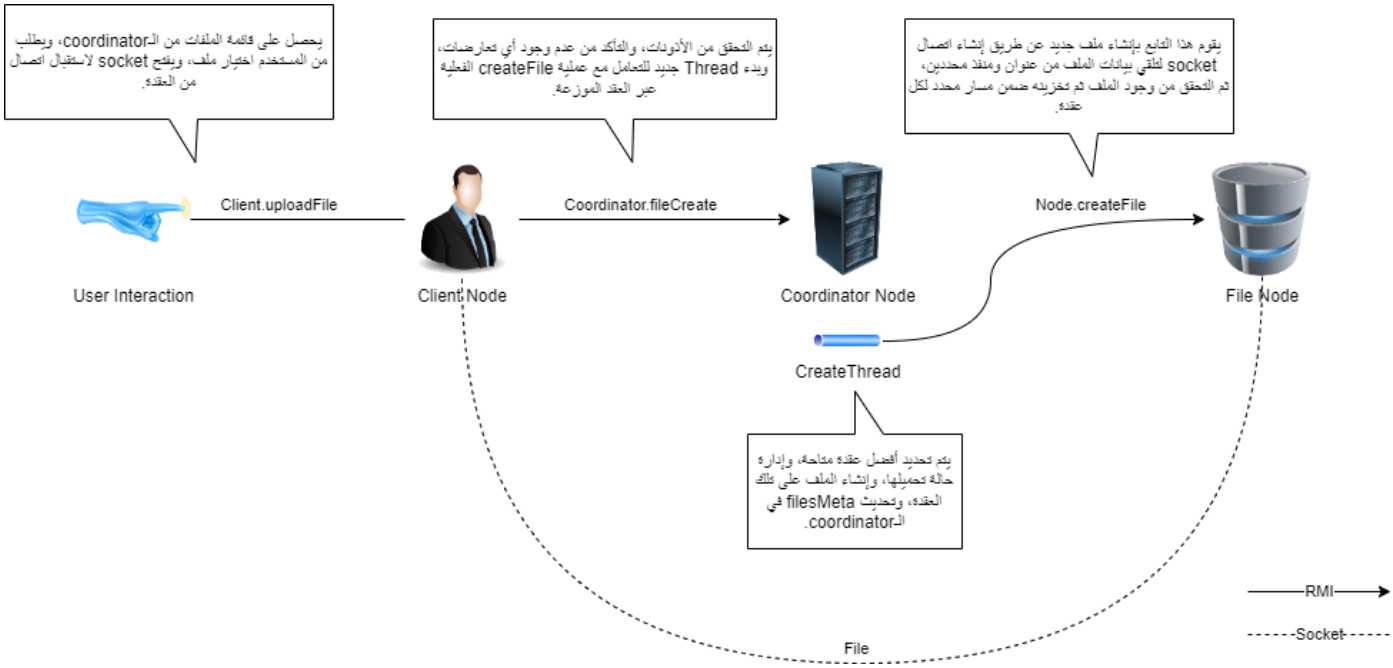


وعندما يتصل مستخدم جديد بالنظام، يقوم بأخذ ال Coordinator remote object وتهيئة مجلدين لرفع وتنزيل الملفات:

```
public static void main(String[] args) throws Exception {  AbDulla Musa *

    CoordinatorInt coordinator = (CoordinatorInt) Naming.lookup(
            name: "rmi://localhost:5000/coordinator"
    );
    Client client = new Client(coordinator);
    File uploadDir = new File(client.userUploadPath);
    File downloadDir = new File(client.userDownloadPath);
    uploadDir.mkdirs();
    downloadDir.mkdirs();
    client.run();
}
```

ثانياً: ميزات النظام:

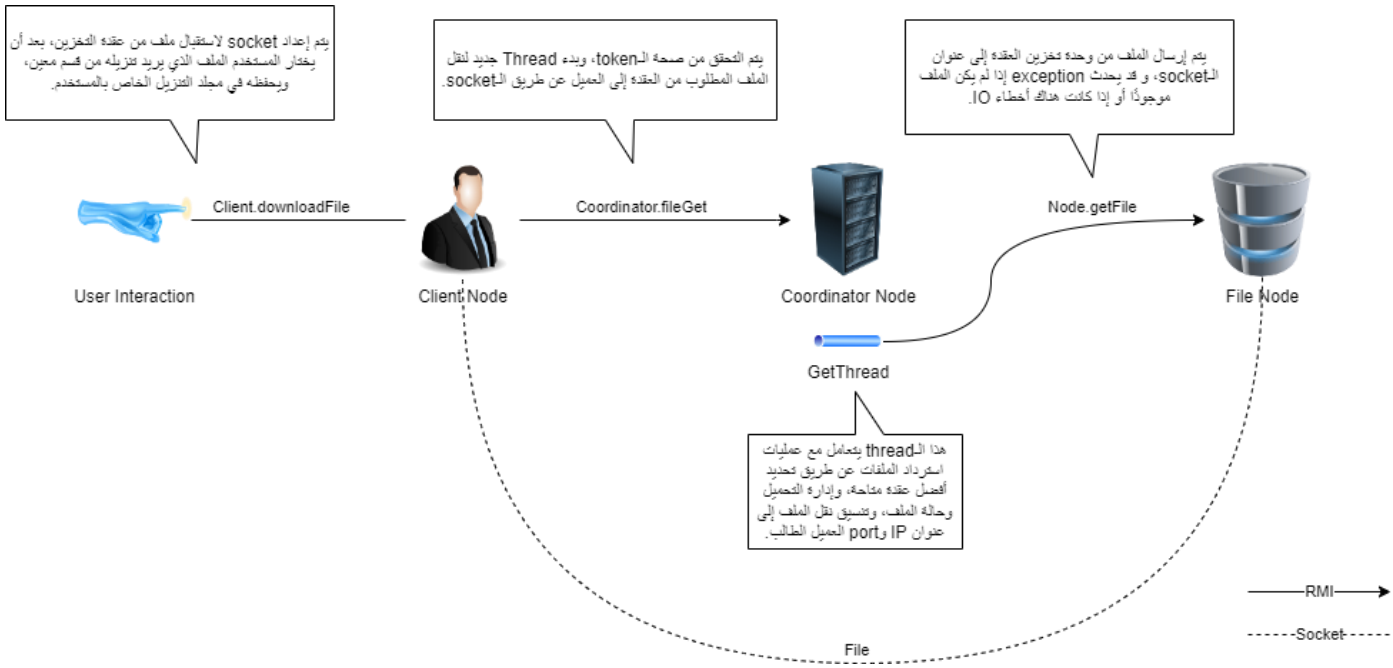
إنشاء ملف:



ملاحظة:

- يجب وضع الملف المراد رفعه ضمن مجلد الـupload.
- يتم إضافة معلومات الملف الجديد و العقدة الموجود عليها في الـfilesMeta في الـcoordinator.

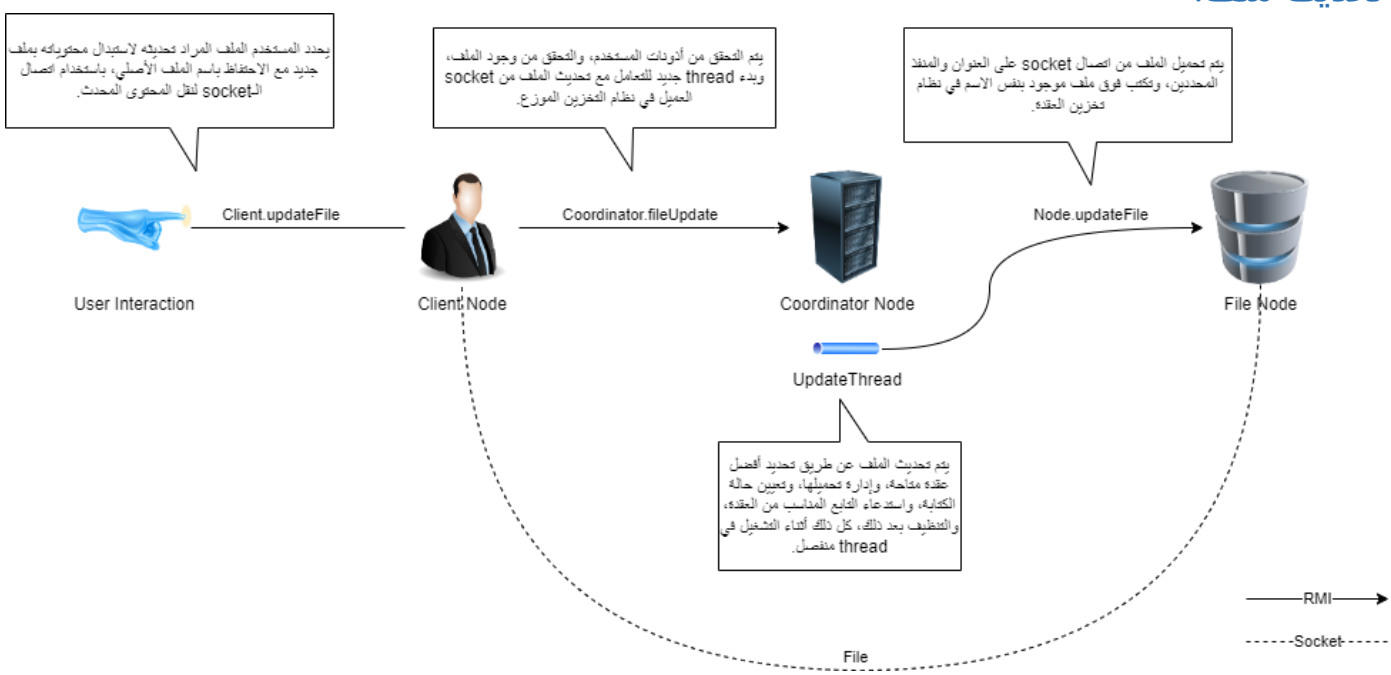
جلب ملف:



ملاحظة:

- عند اختيار العقدة ذات الحمل الأقل يتم مراعاة كون الملف موجود على العقدة.

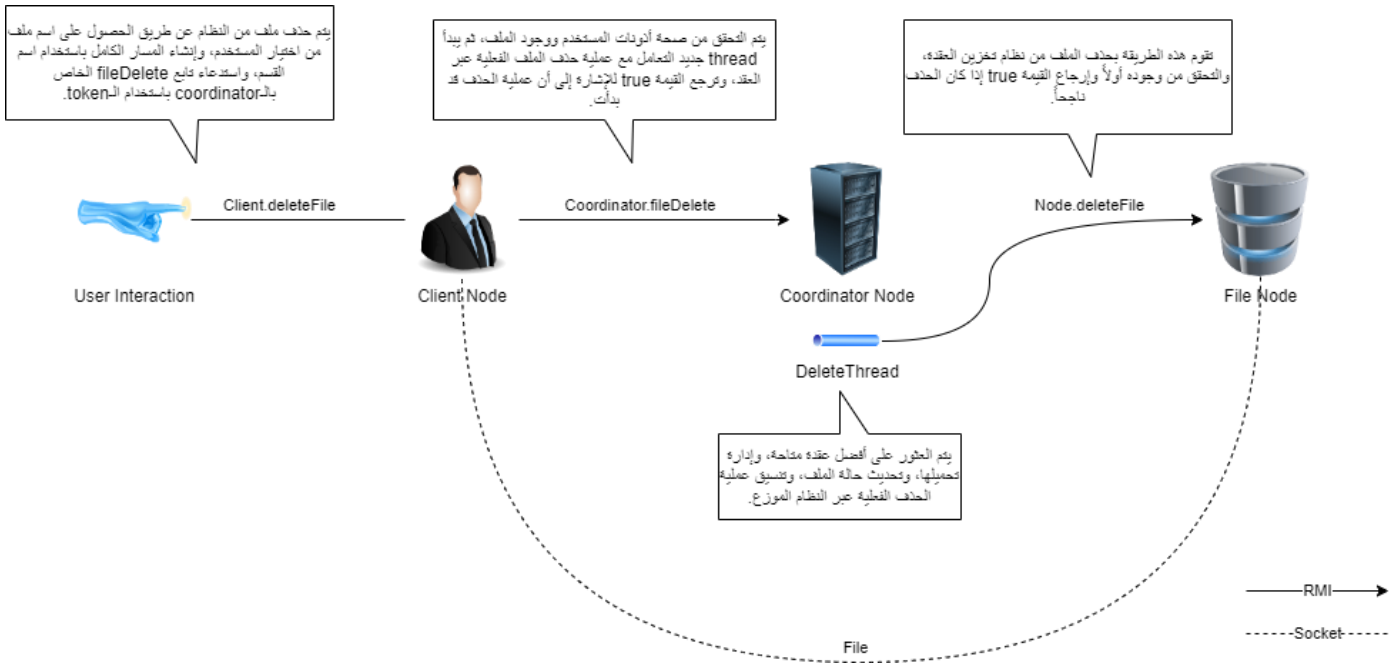
تحديث ملف:



ملاحظة:

- بعد رفع الملف الجديد على عقدة ما يتم عمل invalidate للملف على العقد الأخرى، و ذلك عن طريق إزالتها من الـfileMeta الخاص بالملف المعدل، و لاحقاً عند المزامنة يتم عمل overwrite للملف على العقد الأخرى.

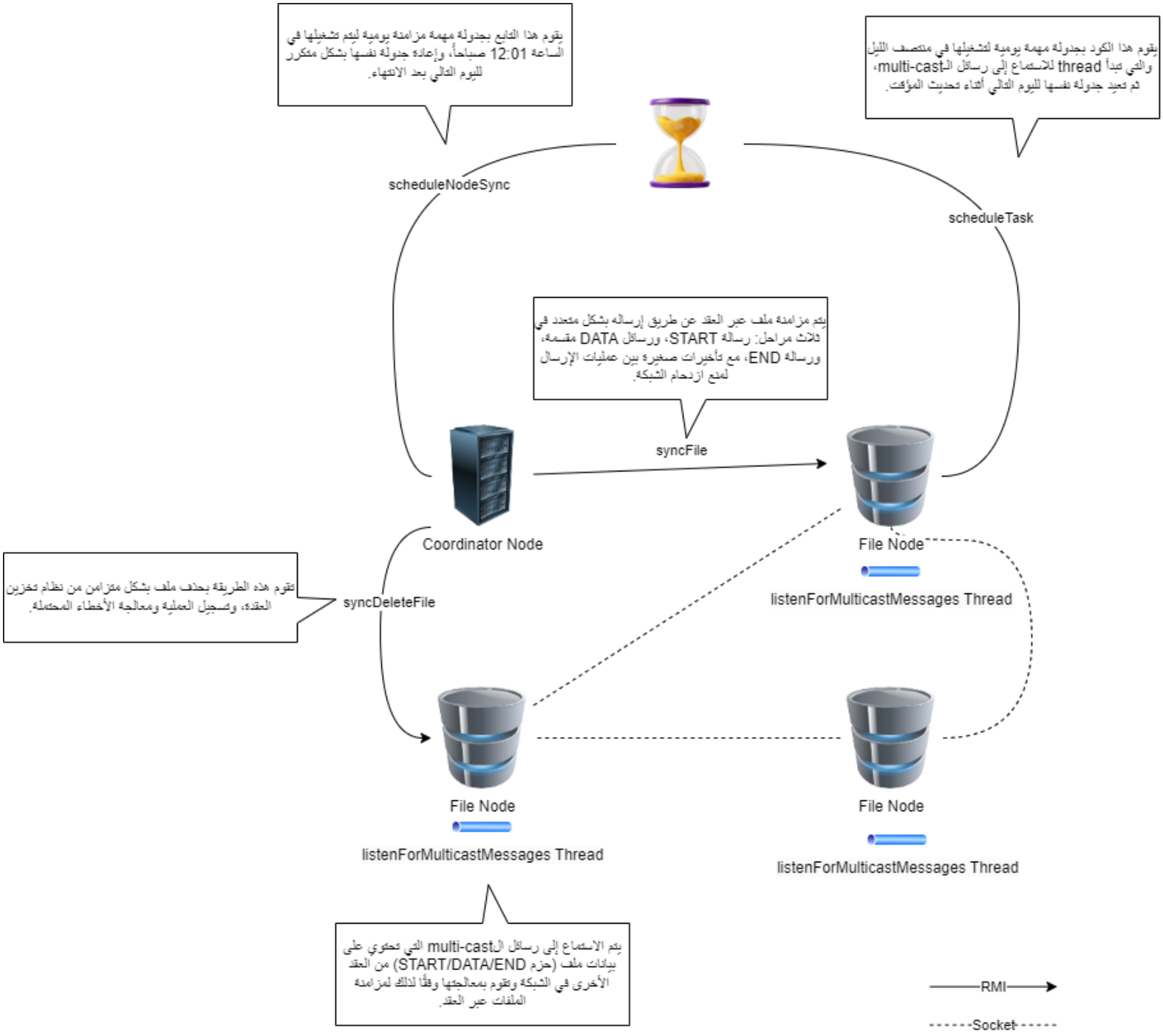
حذف ملف:



ملاحظة:

- بعد حذف الملف يتم إزالة جميع العقد من الـfileMeta الخاص بالملف المحذوف (وهكذا لن يظهر الملف عندما يستعرض المستخدم قائمة الملفات)، و لاحقاً عند المزامنة يتم حذف الملف الفعلي من جميع العقد.

مزامنة الملفات بين العقد:



ملاحظة:

- إذا كان هناك ملف مسجل بالـfilesMeta بأنه ليس موجود بأي عقدة فيتم حذفه من جميع العقد.
- إذا كان هناك ملف مسجل بالـfilesMeta بأنه موجود ببعض العقد (ليس كلها) فيتم اختيار عقدة تحتوي الملف و إعلامها بأن ترسل الملف لباقي العقد.

آلية مزامنة الملفات بين العقد باستخدام Multicast:

تُستخدم للسماح للعقدة ببث ملف معين إلى مجموعة من العقد الأخرى في النظام دفعة واحدة، بهدف مزامنة الملفات.

1. إعداد البث المتعدد في العقدة (NodeImp)

عند إنشاء نسخة من الصنف NodeImp، تتم تهيئة إعدادات البث المتعدد كالتالي:

- الحصول على عنوان مجموعة البث المتعدد (InetAddress) والPort.

حساب حجم المؤقت للحزمة: MULTICAST_PACKET_BUFFER_SIZE.

2. عملية مزامنة الملفات كمرسل (تابع syncFile)

عندما تقوم عقدة بإرسال ملف للمزامنة:

- يتم التحقق من وجود الملف المحلي المحدد.
- تُستخدم FileInputStream للقراءة و MulticastSocket للإرسال.
- خطوات البث:

1. رسالة START: للإعلان عن بدء إرسال ملف (تحتوي مُعرّف المرسل ومسار الملف).
2. رسائل DATA: يُقرأ الملف كأجزاء (DATA_CHUNK_MAX_PAYLOAD_SIZE) وتُرسل كل جزء كرسالة DATA.
3. رسالة END: تُرسل بعد جميع أجزاء البيانات للإشارة إلى اكتمال البث.

3. استقبال رسائل البث المتعدد (تابع listenForMulticastMessages)

يعمل هذا التابع عندما يتم استدعاء scheduleTask والذي بدوره يعمل كل يوم عند الساعة 12 AM

يقوم التابع بإنشاء thread (multicastListenerThread) للاستماع المستمر لرسائل المزامنة الواردة.

يتم تجاهل الرسائل التي أرسلتها العقدة نفسها.

- عند استلام رسالة START:

تستعد العقدة لاستقبال ملف جديد وتفتح FileOutputStream له، وتخزنه في خريطة receivingFilesMap.

- عند استلام رسالة DATA:

تتم كتابة محتوى حمولة الرسالة (payload) إلى تدفق الملف المقابل.

- عند استلام رسالة END:

يتم إغلاق FileOutputStream للملف، مشيراً إلى اكتمال استقباله.

- إعادة جدولة نفسها لليوم التالي.
- إلغاء المؤقت الحالي وإنشاء مؤقت جديد.

النهاية