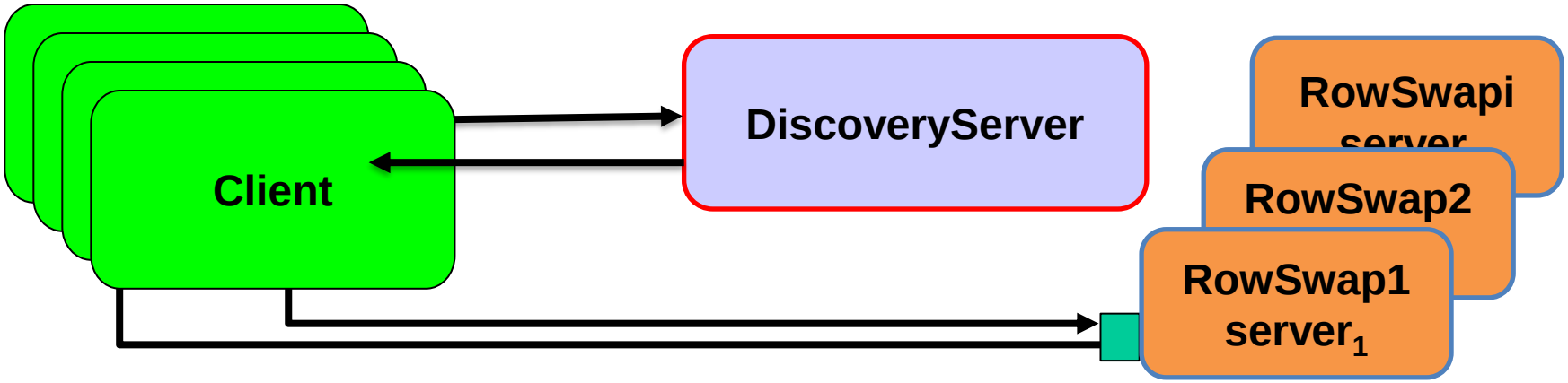

**Università degli Studi di Bologna**
**Scuola di Ingegneria**

# Corso di
# Reti di Calcolatori T

## *Esercitazione 1 (proposta)*
## *Scambio Righe*
## *Socket Java senza connessione*

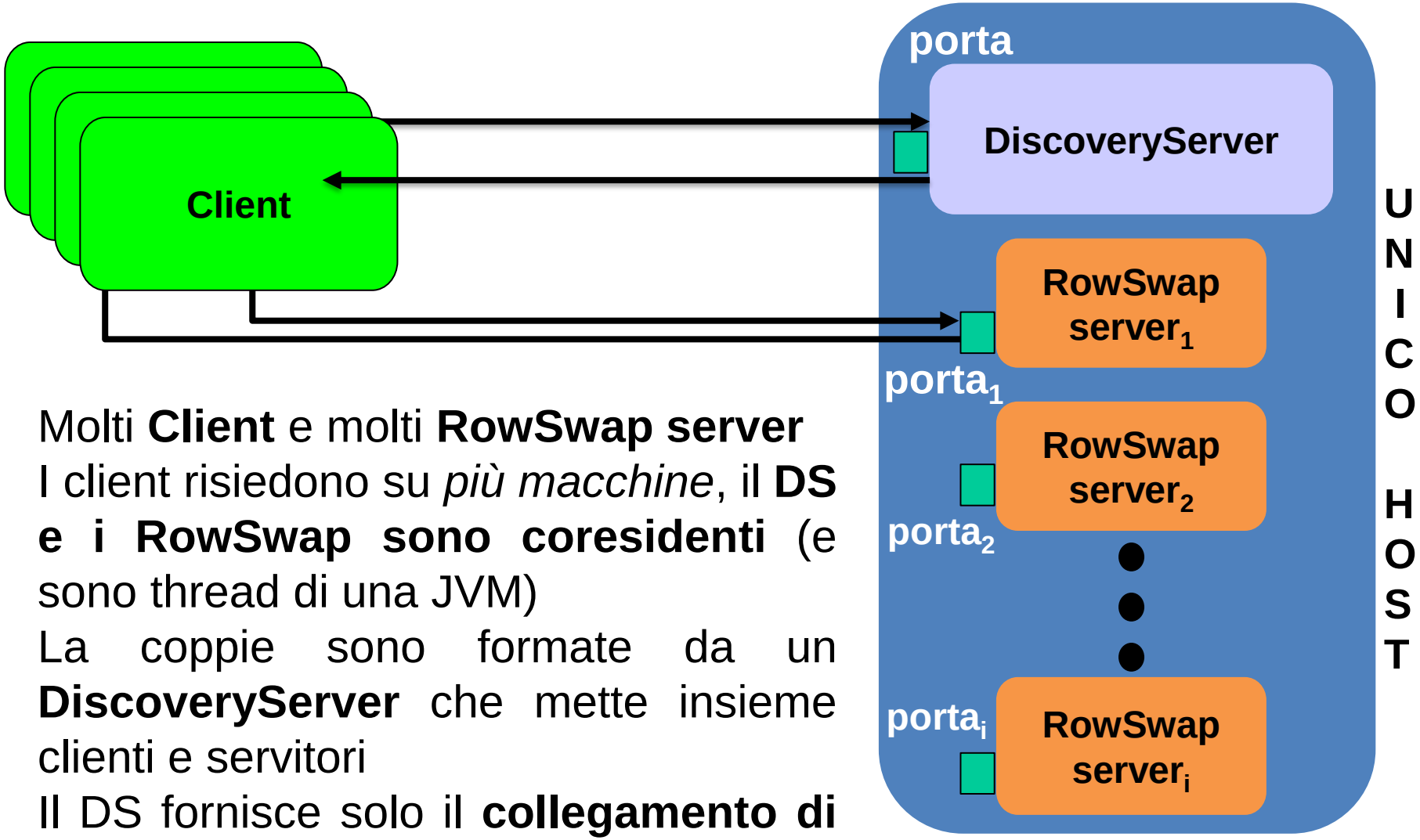
**Antonio Corradi, Armir Bujari**
**Lorenzo Rosa, Giuseppe Martuscelli, Andrea Sabbioni**
Anno Accademico 2023/2024

# ARCHITETTURA CON SISTEMA DI NOMI




Consideriamo **molti clienti** che devono fare richieste a molti **servitori** che non conoscono per **scambiare linee di file testo**
Molti **Client** e molti **RowSwap server**
Per consentire che i clienti possano riferire i servitor relativi definiamo un **Sistema di Nomi** che chiamiamo **DiscoveryServer** che viene interrogato dai clienti per conoscere il servitore relativo
Il DS fornisce solo il **collegamento al cliente per i servizi**

# ARCHITETTURA DISTRIBUITA




Molti **Client** e molti **RowSwap server**
I client risiedono su *più macchine*, il **DS e i RowSwap sono coresidenti** (e sono thread di una JVM)
La coppie sono formate da un **DiscoveryServer** che mette insieme clienti e servitori
Il DS fornisce solo il **collegamento di ogni coppia**

# SERVIZIO DI SCAMBIO RIGHE IN UN FILE

L'applicazione, costituita da più processi, permette **ai clienti di ottenere l'effetto di scambiare due righe dei file di testo per ogni richiesta di scambio ad un server specifico**

L'interazione tra i **molteplici clienti e servitori** è mediata da un **front-end** detto **Discovery Server**, il cui nome globale (**IP e porta**) è noto a tutti. I clienti lo interrogano per avere la indicazione del server di loro interesse (che dipende dal **file** su cui vogliono operare) a cui dopo chiedono in modo ciclico lo **scambio di due righe nel file**

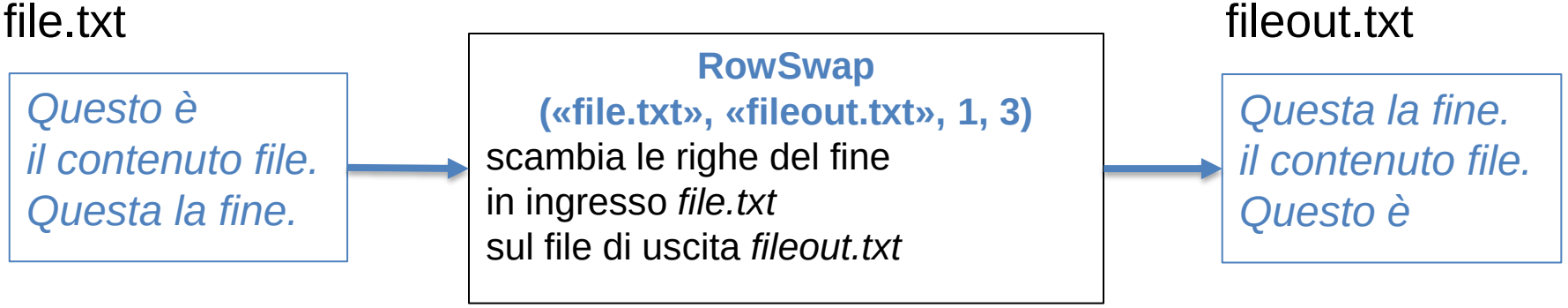
Il **Discovery Server** (**DS**) filtra ogni richiesta e smista il cliente **verso un appropriato processo RowSwap server** (**RS**); il RS gestisce poi ciclicamente le richieste, una alla volta, del cliente
I file da modificare risiedono tutti sull'host dei Server

**Tutta la comunicazione tra cliente e servitori (Discovery Server e RowSwap Server) è implementata tramite socket datagram in Java**

# ESEMPIO SWAP ROW

La funzione di **Row Swap** è attuata da un **RowSwap (RS) server che ha l'obiettivo** di lavorare per scambiare due righe di un file dato in ingresso producendo una uscita che attua lo scambio delle due righe

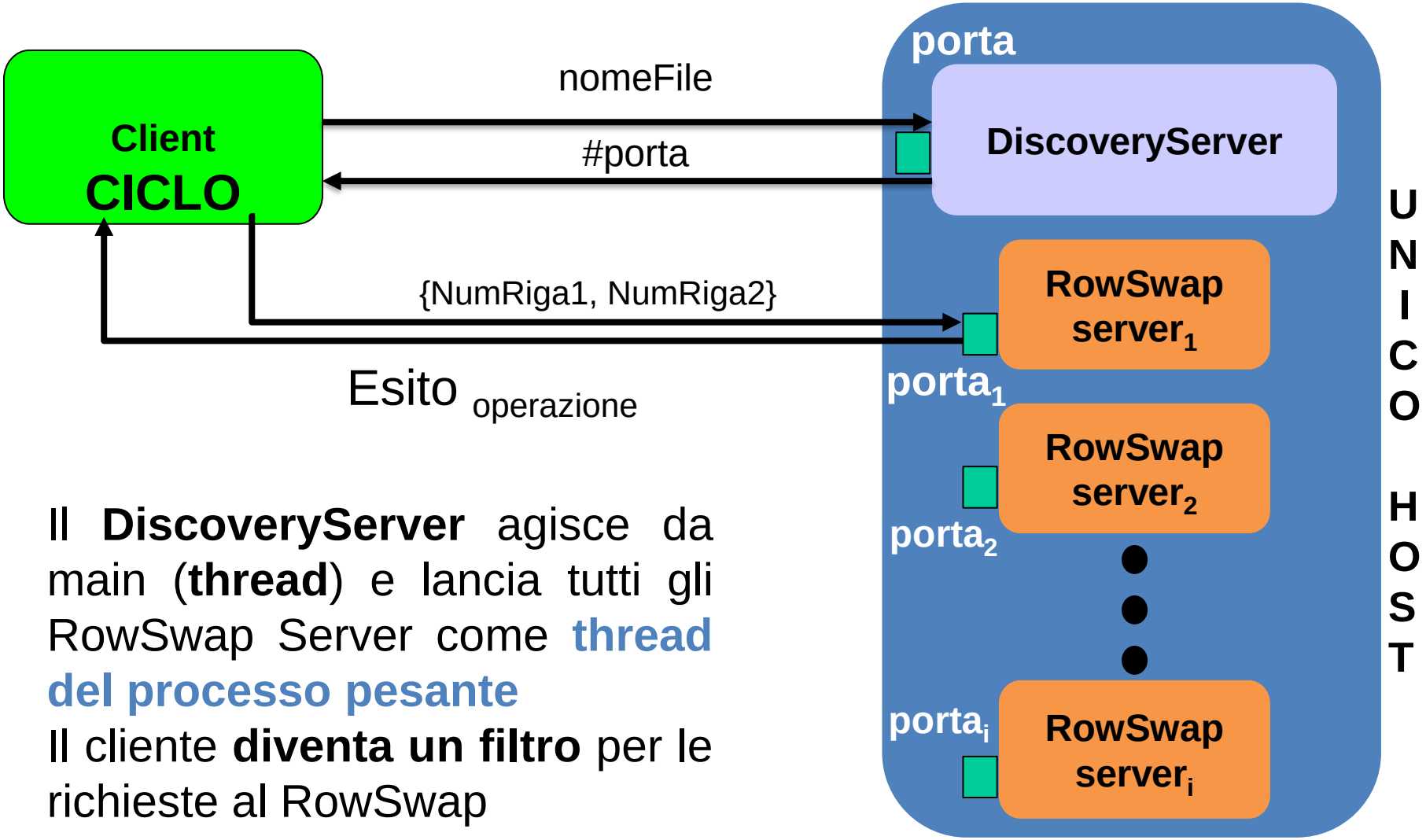
Ad esempio, invocato con *RowSwap file.txt fileout.txt* 1 3




L'**esito dell'operazione** verrà stampato su standard output, gestendo casi di errore (es. una richiesta contenente un file inesistente, un problema nelle linee, etc.)

Nell'implementazione, si pensi ad un cliente che chiede ad ogni giro sempre lo stesso file e lo scambio di due righe per ogni richiesta, modificando sempre lo stesso file

# ARCHITETTURA PER UN CLIENTE

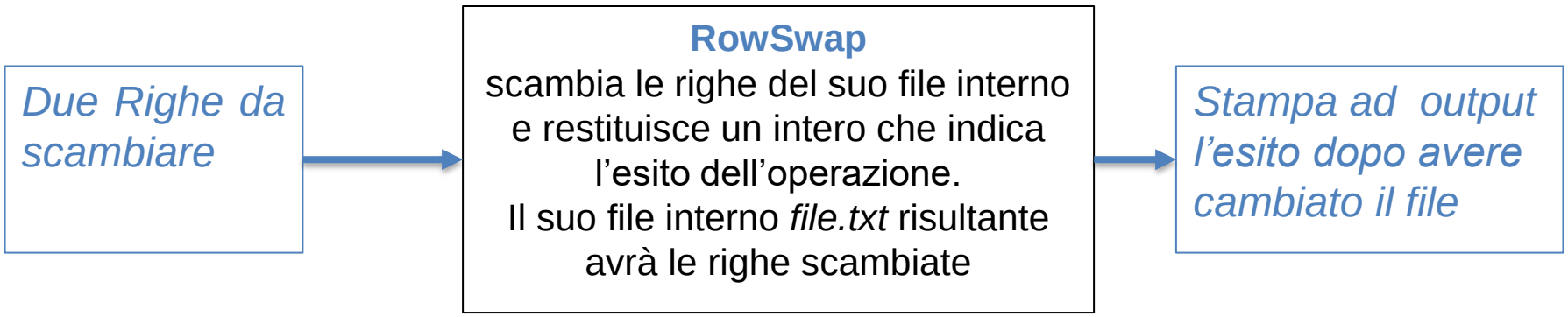



Il **DiscoveryServer** agisce da main (**thread**) e lancia tutti gli RowSwap Server come **thread del processo pesante**

Il cliente **diventa un filtro** per le richieste al RowSwap

# DETTAGLI ROWSWAP SERVER

Ogni **RS server** (sequenziale), al momento della creazione della socket, viene attivato sulla **porta UDP** (costruttore con porta `DatagramSocket(port)`) e **si occupa del file** indicato dall'utente all'invocazione del **DiscoveryServer**
Ad es., il primo RS server leggerà da **nomefile1** e sarà avviato sulla porta UDP **port1**
Gli SR server eseguono la procedura **scambiaRighe** che scambia due righe di un file residente sul server remoto. Ad esempio, per il *file.txt* con contenuto:




L'**esito dell'operazione** verrà stampato a video dal processo server che ha eseguito l'operazione, inclusi i casi di errore detti

# SPECIFICA DISCOVERY SERVER

Il **DiscoveryServer** è un server sequenziale (main thread) e viene attivato da *riga di comando* con la seguente invocazione (a coppie dopo il primo argomento):

```
DiscoveryServer portaDiscoveryServer
nomefile1 port1...   (tutte coppie di argomenti file e porta)
nomefileN portN
```

**In base al numero di file passati come argomento**, il DiscoveryServer **attiva un RS thread per ogni file**. Ciascun RS è realizzato come **thread** coresidente lanciato dal DiscoveryServer e in ascolto sulla porta **UDP** specificata

**(N.B. controllare che tutte le porte siano diverse)**

**In base alla richiesta** del cliente, il DiscoveryServer **risponde con le informazioni di connessione (*porta*)** utili al cliente per contattare il RS server adatto

# DETTAGLI DISCOVERY SERVER

Il **DiscoveryServer**, dopo l'attivazione dei processi RS, si pone in ascolto delle **richieste da parte dei client** sulla **sua socket datagram (*portaDiscoveryServer*)**

Le richieste sono messaggi dai clienti contenenti un **nome file**
Se il nome file è legato a un **RS attivo**, il DiscoveryServer restituisce al client un messaggio con **la porta** del RS responsabile del file, altrimenti restituisce un intero che indica *esito negativo* dell'operazione, identificabile come **file non disponibile (il cliente esce)**

Si presti attenzione al controllo degli argomenti di attivazione del DiscoveryServer, **evitando possibili sovrapposizioni** specialmente sulle porte

# SPECIFICA CLIENT

`RSClient IPDS portDS fileName`

Il **cliente** viene avviato con argomento l'**indirizzo IP** e la **porta** su cui è in esecuzione il DiscoveryServer e il suo unico **file di interesse**
Il DS fa da intermediario nella comunicazione cliente-servitore

Il **Cliente**, usando gli argomenti per trovare il **DS**, gli invia il nome del file di cui vuole scambiare righe
Il Discovery Server, in caso di successo risponde con **la porta** sulla quale è in esecuzione lo **RowSwap** Server di interesse.
Se il nome file non fosse fra quelli noti al DiscoveryServer, il DiscoveryServer invia esito negativo e il client termina

Il **Cliente** fa ogni chiamata successiva **allo stesso RS**, indicando i **due interi** che rappresentano le righe da scambiare
Il cliente riceve in risposta un intero con l'esito dell'operazione
Al termine di un ciclo di richieste, il cliente termina come un filtro

# ESEMPIO CONFIGURAZIONE

Il **DiscoveryServer** (avviato su localhost) rappresenta la classe main che gestisce tutti i processi **RowSwap**.
Di seguito è riportato un esempio di esecuzione coerente con i nomi dei file presenti nella directory del server Java mostrata in figura


Discovery Server:

```
DiscoveryServer 54321 file1.txt 54320
                      file2.txt 54319
```

Client 1:

```
SwapClient localhost 54321  file1.txt
```

Client 2:

```
SwapClient localhost 54321  file2.txt
```

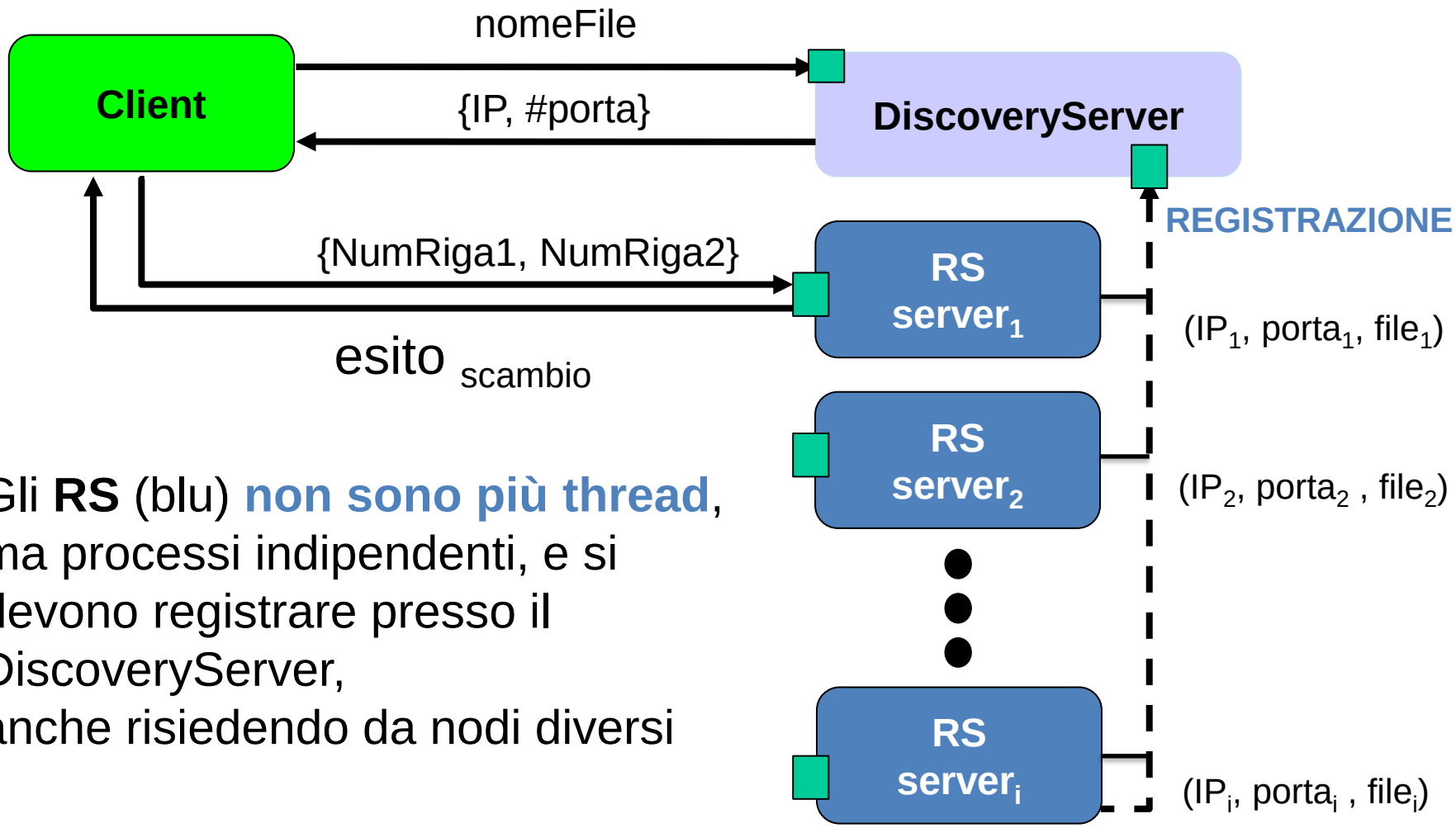
# PROPOSTA DI ESTENSIONE DISTRIBUITA

**Si assuma che il DiscoveryServer e i RS non siano tutti in esecuzione sulla stessa macchina**

Inoltre, si estenda la applicazione in modo che **gestisca dinamicamente la registrazione dei RS anche da macchine diverse**, consentendo sempre ai Client di interrogare il DiscoveryServer per ottenere l'elenco dei file distribuiti che siano attivi

Quello che vogliamo è che il **DiscoveryServer** possa diventare molto dinamico e **sia fornire una lista dei suoi server correntemente attivi** e **sia consentire attivazioni e disattivazioni dei RS server** durante l'esecuzione dei servizi

# ARCHITETTURA DISTRIBUITA




Gli **RS** (blu) **non sono più thread**, ma processi indipendenti, e si devono registrare presso il DiscoveryServer, anche risiedendo da nodi diversi

# PROPOSTA DI ESTENSIONE: DETTAGLI

**DiscoveryServer**: viene esteso in modo da abilitare la registrazione dinamica dei SR e invocato con i numeri di porta:

***DiscoveryServer portaRichiesteClient portaRegistrazioneRS***

Il DiscoveryServer mantiene una **tabella con le corrispondenze fra il nome file richiesto e la macchina e la porta su cui il RS è in ascolto**

L'operazione di **registrazione** verifica l'unicità del nome file e dell'IP e della porta usata dal RS server, ovvero controlla che **non esista un altro RS registrato per lo stesso nome file**, e che **non esista un altro RS collegato sullo stesso endpoint** (IP e porta)

L'operazione di **visualizzazione** restituisce i RS registrati in tabella (nome file ed endpoint)

# DETTAGLI ATTIVAZIONE: RS

Il RS viene realizzato come **processo remoto separato** (lanciato in una macchina e JVM diversa) e viene esteso in modo da consentire la registrazione dinamica presso il DiscoveryServer
Si propone la seguente interfaccia di riga di comando:

***RS IPDS portDS portRS nomeFile***

Una volta attivato, il RS si registra presso il DiscoveryServer passandogli endpoint (IP-RS, portRS) e nomeFile
Se l'esito della registrazione è positivo, il RS sarà raggiungibile dal client

Si estenda ulteriormente la soluzione proposta in modo da consentire anche la ***terminazione*** e la ***de-registrazione dinamica degli SR*** presso il DiscoveryServer

# DETTAGLI ATTIVAZIONE: CLIENT

Il **Client** deve consentire di interrogare il DiscoveryServer e di scegliere il file attraverso l'interazione con l'utente

Prima di attivare qualunque ricezione dal RS, il client richiede al DiscoveryServer la visualizzazione della **lista dei nomi file correntemente attivi**, e permette all'utente di scegliere il nome file a cui è interessato, abilitando poi la comunicazione con il RS

# ANCORA IN TERMINAZIONE RS

Estendere RS in modo da consentire l'interazione col programma da riga di comando e l'invio di un comando di terminazione da console.

Sono possibili **altre alternative**?

**Quali**?