X36DSV – 2. cvičení

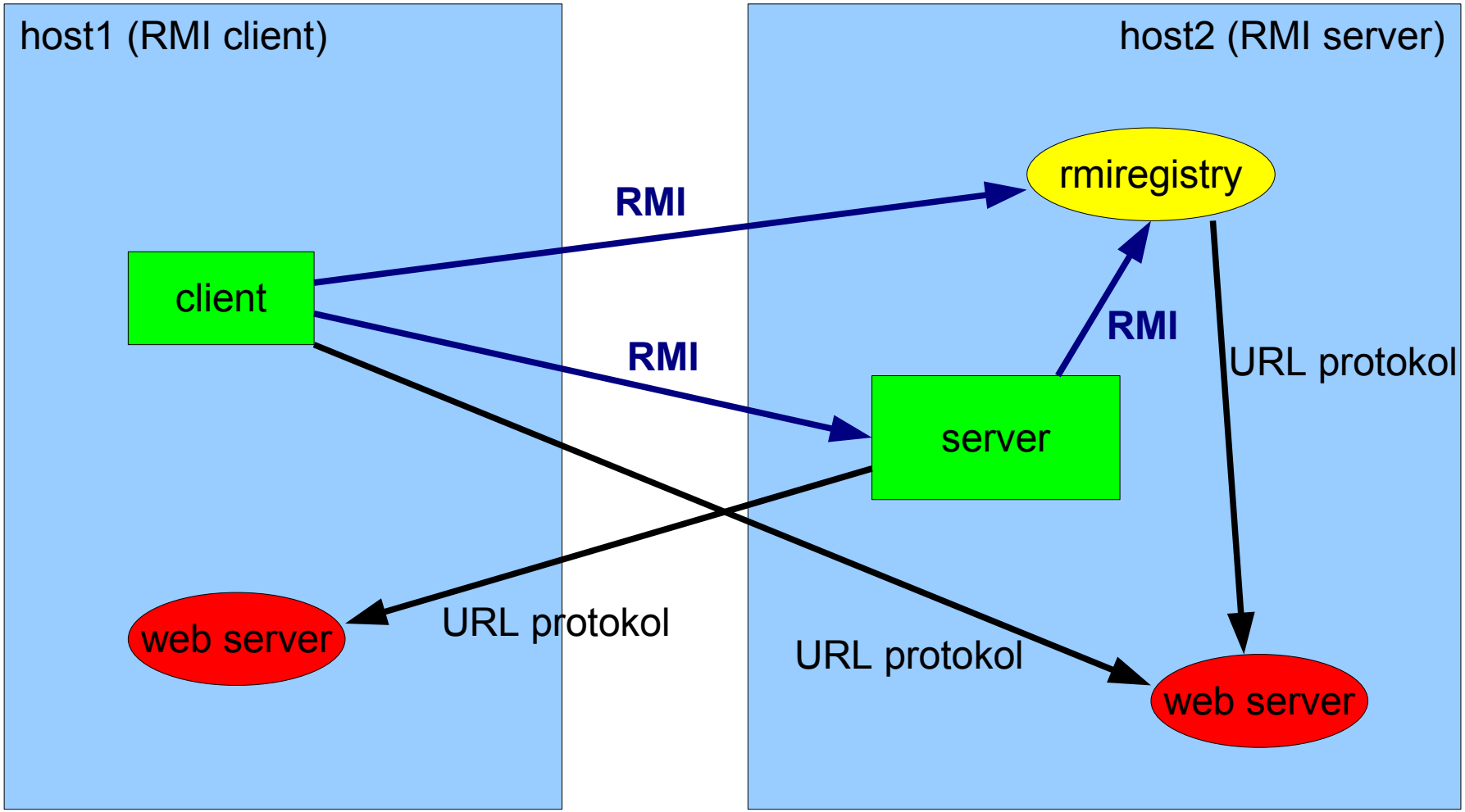
RMI

Remote Method Invocation

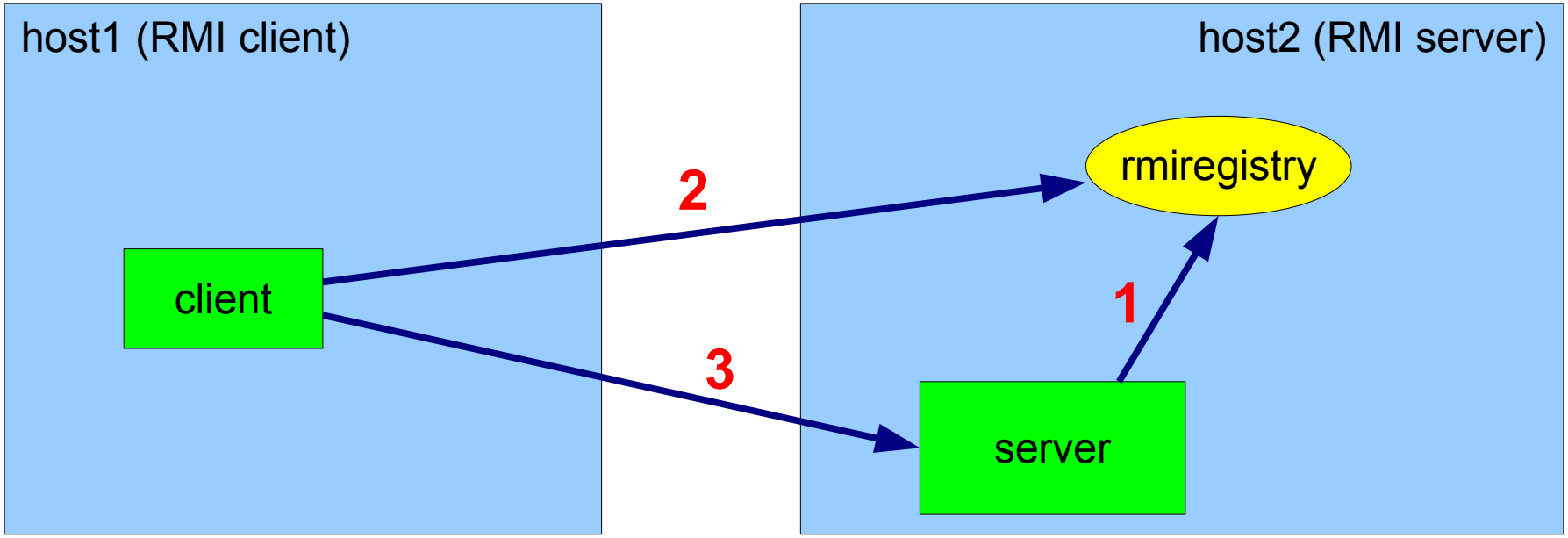
RMI – co to je?

- vyvolání metody z jiné JVM
  - lokalizace vzdáleného objektu
  - komunikace se vzdálenými objekty
  - přenos objektu v bytecode
- typicky klient – server
- http://java.sun.com/docs/books/tutorial/rmi/index.html

RMI aplikace



RMI aplikace



1 – registrace objektu (svázání objektu se jménem) - `Registry.(re)bind`
2 – dotaz na objekt příslušící danému jménu- `Registry.lookup`
3 – práce se vzdáleným objektem - `Remote_object.remote_method`

Části Java RMI aplikace

- vzdálený interface
  - deklarace metod vzdáleného objektu vyvolávaných klientem
- vzdálený objekt
  - implementace vzdáleného interface
- klient
- komunikace
  - stub, skeleton (od Java 1.5 plně transparentní)

Remote

- remote methods
  - metody volané mezi JVM
- remote objects
  - objekty s remote metodami
- remote interface
  - určuje remote objekty, rozšiřuje interface *java.rmi.Remote*
  - všechny remote metody musí deklarovat *java.rmi.RemoteException*
- remote stub
  - nahrazuje remote objekt lokální reprezentací (proxy)
  - stejná množina remote interface, marshaling, unmarshaling

Tvorba aplikace

- remote interface
- implementace remote objektů
- implementace klienta
- překlad zdrojových kódů
- vytvoření stubu
- distribuce aplikace
- start aplikace

Implementace

- deklarace remote interface
- definice konstruktoru remote objektu
- implementace všech remote metod
- vytvoření a instalace remote objektů
  - main metoda
  - vytvoření a instalace security manageru
  - vytvoření instancí remote objektů
  - registrace objektů ve jmenných službách
    - rmiregistry, JNDI

RMI - Interface

```
package compute;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface MathServer extends Remote {
    public int secti(int a, int b) throws RemoteException;
}
```

RMI – Remote object

```
package server;

import java.rmi.*;
import compute.*;

public class MathServerImpl implements MathServer {

    // pro potrebu serializace (marshalingu)
    private static final long serialVersionUID = -386L;

    public MathServerImpl() throws RemoteException {
        super();
        }

    public int secti(int a, int b) throws RemoteException {
        int result;
        result=a+b;
        System.out.println(a + " + " + b + " = " + result);
        return result;
    }
}
```

RMI - Server

```
package server;

import java.rmi.*;
import compute.*;

public class Server {
    public static void main(String[] args) {
        if (System.getSecurityManager() == null)
            System.setSecurityManager(new RMISecurityManager());
        String name = "MathD";
        try {
            MathServer msi = new MathServerImpl();
            MathServer stub =
                  (MathServer) UnicastRemoteObject.exportObject(msi);
            Registry registry = LocateRegistry.createRegistry(2010);
            registry.rebind(name, stub);
        }
        catch (Exception e) {
            System.err.println("Data exception: " + e.getMessage());
        }
    }
}
```

RMI - Client

```
package client;

import java.rmi.*;
import compute.*;

public class Client {
    public static void main(String args[]) {
        if (System.getSecurityManager() == null)
            System.setSecurityManager(new RMISecurityManager());
        try {
            MathServer mth;
            String name = "MathD";
            Registry registry = LocateRegistry.getRegistry("localhost",
2010);
            mth = (MathServer) registry.lookup(name);
            int a = Integer.valueOf(args[0]).intValue();
            int b = Integer.valueOf(args[1]).intValue();
            System.out.println(a + " + " + b + " = " + mth.secti(a, b));
        }
        catch (Exception e) {
            System.err.println("Data exception: " + e.getMessage());
        }
    }
}
```

Bezpečnost

- defaultní metody
- policytool.exe || policytool
- java.security.policy

```
grant {
  permission java.net.SocketPermission "*:1024-65535","connect,accept";
  permission java.io.FilePermission "\data\-","read";
  permission java.io.FilePermission "c:\\home\\ann\\classes\\-","read";
};
```

```
grant {
  permission java.security.AllPermission;
};
```

Překlad a spuštění

```
# kompilace kódů (Win i Lin)
javac compute/MathServer.java
javac server/MathServerImpl.java server/Server.java
javac client/Client.java

# pokud nespouštíte v rámci serveru vlastní RMIregistry tak
start rmiregistry
rmiregistry &

# spuštění serveru (pod Lin ';' -> ':')
java -cp compute.jar;server.jar
    -Djava.security.policy=java.policy server.Server

# spuštění klienta (pod Lin ';' -> ':')
java -cp compute.jar;client.jar
    -Djava.security.policy=java.policy client.Client 64 46
```