PRiR I

Programowanie Równoległe i Rozproszone

Wykład 1 – Wstęp i RMI

Marek Kasztelnik
WSZIB Krakow

Wstep do PRiR + RMI

- O czym ten przedmiot?
- System rozproszony/Distributed System
- Remote Procedure Call/RPC
- Java Remote Method Invocation/RMI

Programowanie Równoległe i Rozproszone

- Tematy
  - RMI /Java – *programowanie rozproszone*
  - MPI /C – *programowania równoległe*
  - Wątki w Java - *programowanie współbieżne*
  - Web Services /Java i inne - *programowanie rozproszone*
  - PCAM – metoda zrównoleglania

Rozproszone vs. Równoległe

- Aplikacje rozproszone (ang. distributed)
- połączone siecią różne (heterogeniczne) jednostki
  - typowo mające różne funkcje w systemie (np. client-serwer)
  - rozległe
- Aplikacje równoległe (ang. parallel)
  - specjalizacja: obliczenia
  - przeważnie homogeniczne
  - typowo wiele jednostek blisko (siec lokalna)
  - wiele procesów/wiele procesorów/ wiele systemów operacyjnych

... vs. Współbieżne

- Aplikacje współbieżne (ang. concurent)
  - wielokorowe i wieloprocesowe systemy
  - jeden system operacyjny/jeden proces/wiele wątków
  - typowe zastosowania: serwery do obsługi masowej
  - typowe zastosowania: obliczenia

Rozproszone/Równolegle/ Współbieżne - różnice

- **parametry komunikacji**
- dedykowane architektury sprzętowe
- typowe schematy komunikacji
- typowe zastosowania

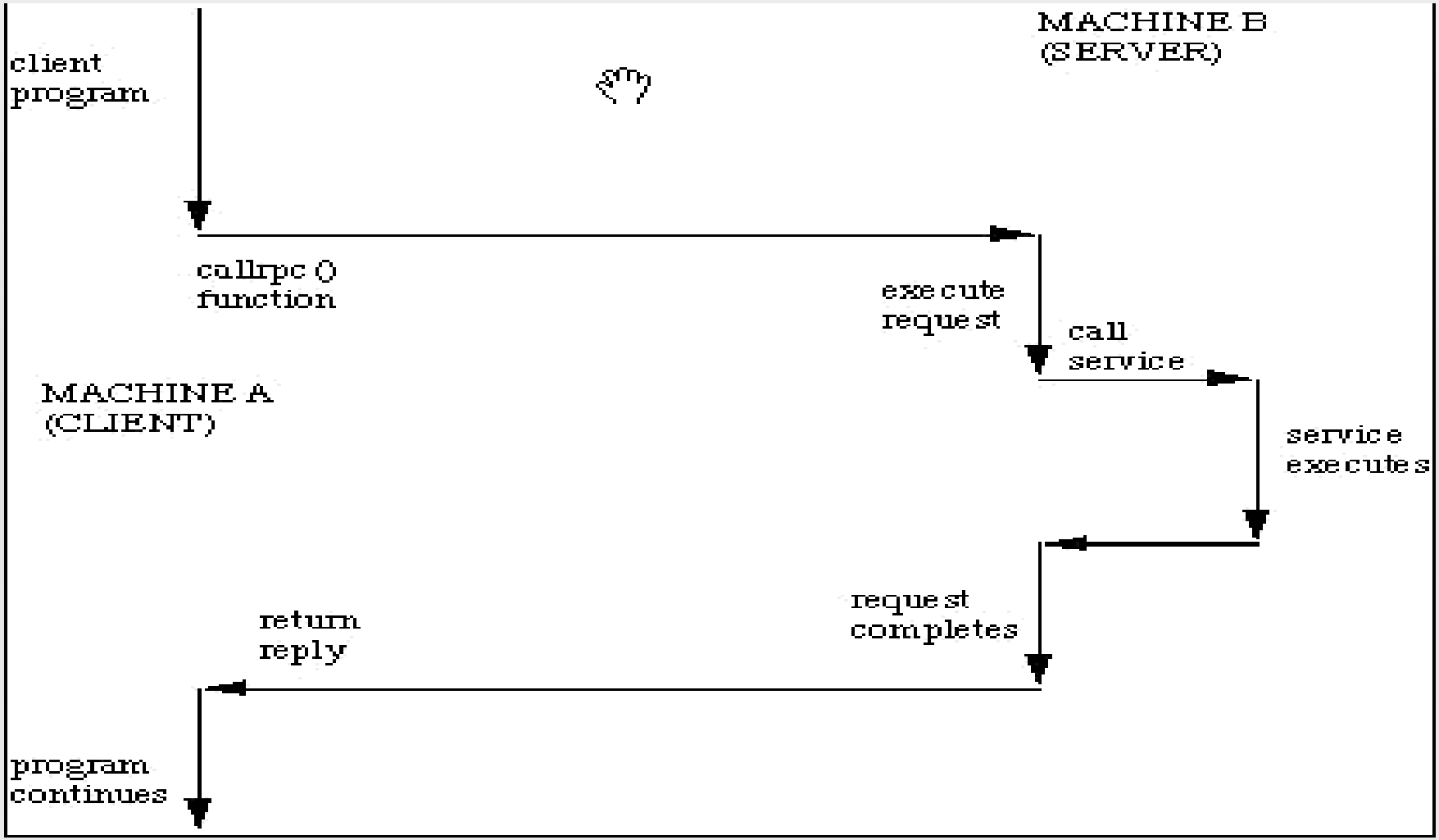
Rozproszone/Równolegle/ Współbieżne – cechy wspólne

- Concurrency yields complex behaviors
  - Reasoning about non-determinism
- Consistency/coordination despite lack of global state/time
- Failure is the “norm” and not the “exception”
  - Spreading effects of corruption
  - Economic impact of denial of service

Systemy rozproszone - przykłady

- Peer-to-peer services
- Web servers (http)
- Internet clusters
- Wireless sensor networks
- Three tier database servers
- Client-server systems

RPC – o co chodzi



Jak sie do tego zabrać?

- Kroki do stworzenia programu RPC
  1. Specify the protocol and interface for client server communication
  2. Develop the server program
     1. funkcjonalnosc
     2. obsługa protokołu
  3. Develop the client program
     1. procedura
     2. obsługa protokołu

Należy pamiętać o:

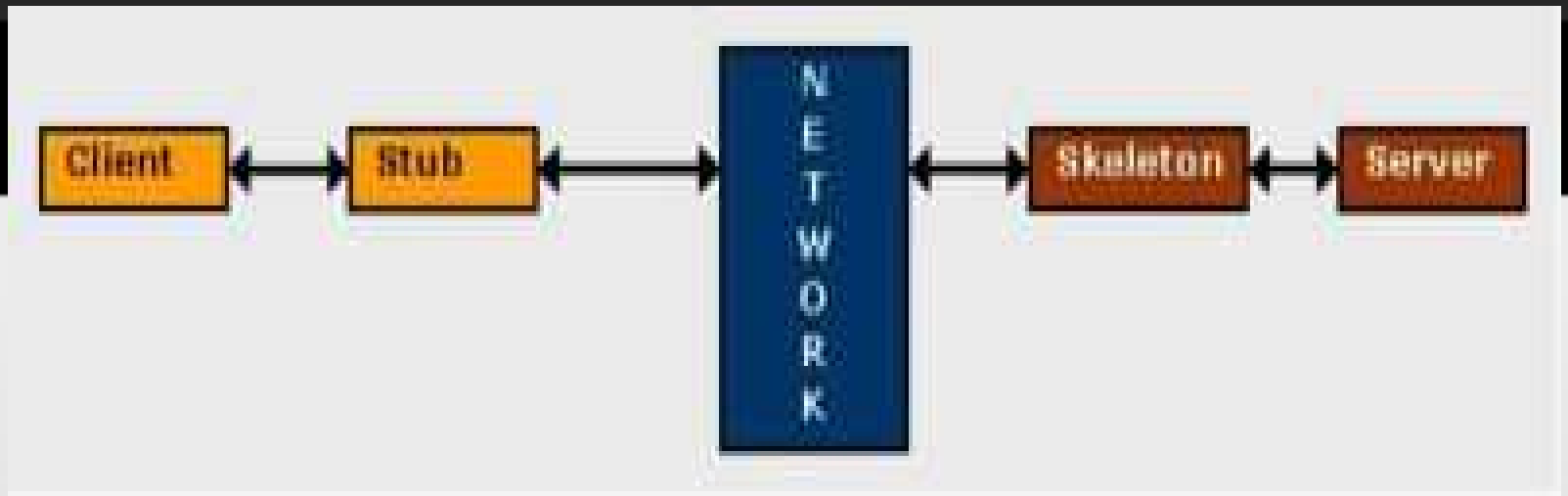
- Należy pamiętać o:
  - system przekazywania danych (serializacja)
  - obsługa wersji
  - obsługa błedów w sieci
  - bezpieczeństwo
  - heterogeniczne systemy (?)
  - właściwości wartwy sieciowej

Błędne założenia

- 1. The network is reliable.
- 2. Latency is zero.
- 3. Bandwidth is infinite.
- 4. The network is secure.
- 5. Topology doesn't change.
- 6. There is one administrator.
- 7. Transport cost is zero.
- 8. The network is homogeneous.

http://en.wikipedia.org/wiki/Fallacies_of_Distributed_Computing

Typowa Architektura



- Stub - jest przedstawicielem (namiastką) serwera na odległość, tj. klient de facto wywołuje operacje używając stub, natomiast ten przekazuje takie wywołanie do stubu po stronie serwera Stuby są odpowiedzialne za odpowiedni marshalling obiektów przekazywanych jako argumenty czy przekazywanych jako odpowiedzi.
- Skeleton - nazwa na stub znajdujący się po stronie serwera

Implementacje RPC

- CORBA (OMG)
- DCOM obecnie element .NET (Microsoft)
- Java RMI (Sun)
- Globe (Vrije Univesitate)
- Web Services

Dwa słowa o CORBA

- CORBA – (Common Object Request Broker Architekture)
- Zbiór standardów opracowanych przez OMG (Object Management Group)
- Implementcje nazywamy *Pośrednikiem ORB (Object Request Broker)*
- *Interface w IDL*
- *Niezależność od platformy*
  - *protokół IIOP*

Java RMI

- RMI to podstawowy mechanizm typu RPC w języku Java.
- W pełni obiektowy
- Językiem specyfikacji interfejsu jest tu sama Java.
- Szeregowanie danych odbywa się przez interfejs, który musi być implementowany przez przekazywane obiekty

Interfejs RMI

- Interfejs w RMI jest po prostu interfejsem w sensie języka Java.
- Można jedynie wołać jego metody (brak dostępu do pól obiektu)
- Każdy interfejs zdalny musi:
  - Rozszerzać interfejs java.rmi.Remote
  - Wszystkie jego metody deklarują klauzulę throws java.rmi.RemoteException

Strona Klienta

- Do obiektów zdalnych odwołuje się przez referencję do odpowiedniego interfejsu
- Referencje pobieramy z rejestru na podstawie nazwy (o tym za chwile)
- Można też realizować różne wersje interfejsów i, mając odpowiednią referencję, sprawdzać, którą wersję implementuje operatorem instanceof.
- W rzeczywistości referencja do obiektu zdalnego jest po prostu referencj¡ na skojarzon¡ z nim lokalną namiastką (stub)

Klient musi mieć

- Interface
- Lokalizacje serwera/Nazwe serwera
- Stub (dla javy < 1.5)
- Obsługiwać RemoteException

Implementacja serwera

- Implementujemy interface zdalny faktyczną funkcjonalnością
- Aby metody obiektu mogły być zdalnie dostępne, musi on spełniać dwa warunki:
  - Implementować interfejs zdalny (remote)
  - Być udostępniony w sieci
- Najprościej uzyskać ten efekt jest dziedzicząc po którejś z podklas klasy java.rmi.server.RemoteServer, najczęciej java.rmi.server.UnicastRemoteObject.
- Inne rozwiazanie to zajestrowanie servera uzywajac UnicastRemoteObject.exportObject(obj, 0)
  - Jaki jest plus tego rozwiazania?

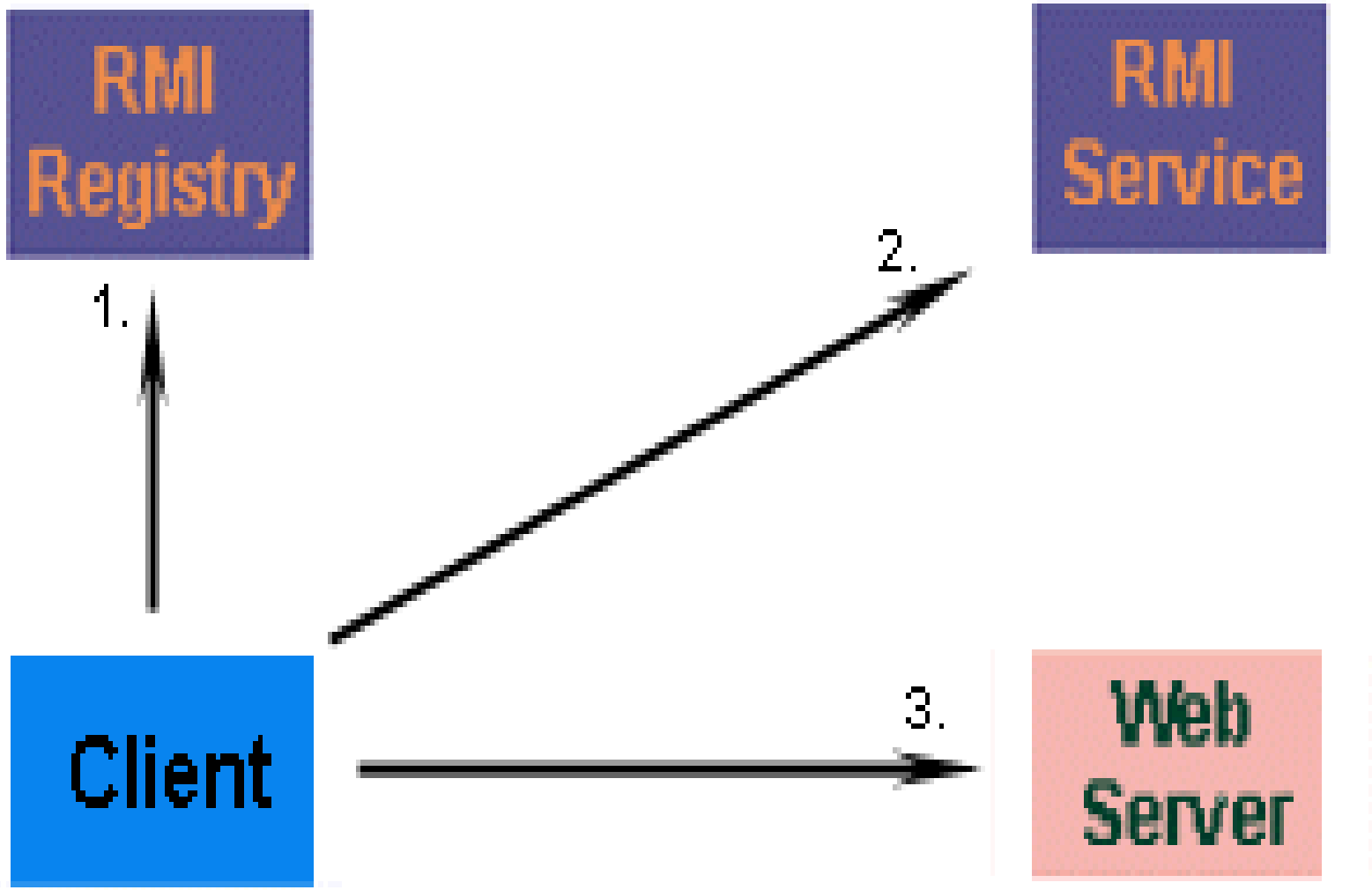
Przygotowanie serwera

- Gotowe klasy kompiluje się
- Następnie wywołuje się generator namiastek rmic. Jego argumentem jest nazwa klasy (dla javy < 1.5)
  - Dla klasy Klasa z pliku Klasa.class zostan¡ wtedy wygenerowane klasy:
  - Klasa_Stub - namiastka dla klienta
  - Klasa_Skel – namiastka dla serwera (java < 1.5)

Rejestr

- rmiregistry - rejestracja serwerów, nawiązywanie łączności między klientem a serwerem RMI
- Polecenie: rmiregistry
- bardzo proste wyszukiwanie serwisów na podstawie nazwy.
- Serwis identykowany jest na podstawie URL o postaci [rmi:]//<komputer>[:<port>]/<nazwa usługi>,
- np. //komputer.w.domenie.pl/MojSerwer, czy rmi://127.0.0.1/Zegar.
- Pominięcie nazwy komputera oznacza localhost, a port domyślny to 1099.

Sekwencja ze strony clienta



O serializacji

- **Serializacja** to proces zapisu obiektu w postaci sekwencyjnego strumienia danych (bajtów). Dzięki takiej transformacji obiekt można przechowywać w pamięci masowej w prostej postaci tablicy danych. Serializacja ułatwia też transport obiektu poprzez sieć. Procesem odwrotnym do serializacji jest deserializacja, czyli odtworzenie oryginalnego obiektu z jest zapisu w sekwencji bajtów.
- **Marshalling** obiektu jest procesem zbliżonym do serializacji. Poprzez marshalling rozumie się przekształcenie obiektu z postaci w jakiej jest przechowywany w pamięci do jakiejkolwiek postaci pozwalającej przechować go poza pamięcią. Dla przykładu zapis obiektu w relacyjnej bazie danych także jest procesem marshallingu. W tym znaczeniu serializacja jest jednym z rodzajów marshallingu.

Serializacja z Java

- Każdy obiekt, który ma być przekazywany przez RMI powinien implementować (implements) interfejs java.io.Serializable.
- Interfejs ten nie zawiera żadnych metod, (nie trzeba robić nic więcej oprócz deklaracji). Jego implementacja to jedynie zgoda na serializowanie danego obiektu.
- Klasy pochodne również będą serializowalne

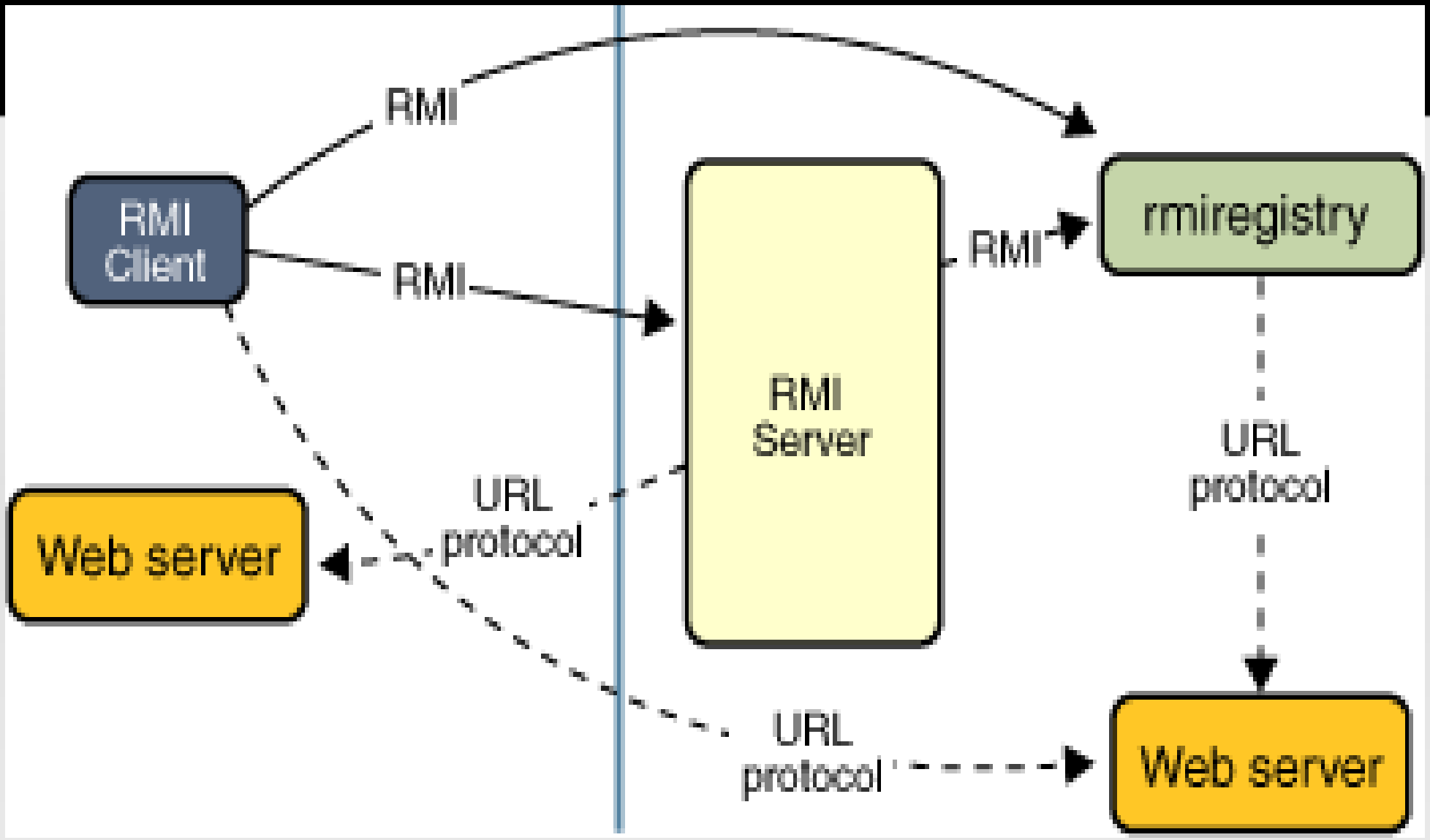
Serializowanie niestandardowe

- Jeżeli potrzebne jest nietypowe szeregowanie, można je osiągnąć implementując ten interfejs oraz deniując następujące metody:
  - private void writeObject(java.io.ObjectOutputStream out) throws IOException
  - private void readObject(java.io.ObjectInputStream in) throws IOException, ClassNotFoundException;
- W wypadku napotkania przy szeregowaniu dynamicznej struktury danych na nieszeregowalny obiekt, rzucany jest wyj¡tek java.io.NotSerializableExcept ion.

RMI i zarządzanie pamięcią

- Choć RMI jest proste w użyciu, jego protokół jest dość złożony. Główną tego przyczyną jest rozproszone zwalnianie pamięci (distributed garbage collection).
- Dla każdego obiektu zdalnego utrzymywany jest licznik zdalnych referencji. Protokół RMI zapewnia odpowiednią jego aktualizację. Obiekty zdalne mogą być zniszczone dopiero, kiedy nie ma do nich ani zdalnych, ani lokalnych referencji.
- Aby serwer np. kończy sam pracę, kiedy nie jest już potrzebny, przydatna może być informacja o zniknięciu ostatniej zdalnej referencji. W takiej sytuacji serwer powinien implementować interfejs Unreferenced, zawierający tylko jedną, bezparametrową metodę unreferenced() a która jest wywoływana przez RMI w momencie stwierdzenia opróżnienia listy

RMI – Java może więcej



RMI Activation

- Uruchamianie serwisów na żądanie już od wersji 1.2 Javy dostępny jest RemoteObject także serwer **java.rmi.activation.Activatable.**
- Referencje zdalne mogą zachowywać ważność po zniszczeniu obiektu, do którego się odwoływały, a nawet po restarcie maszyny wirtualnej, na której się znajdowała.

http://java.sun.com/javase/6/docs/technotes/guides/rmi/activation/overview.html

RMI - ewolucja

- Ciągle upraszczanie sposobu tworzenia aplikacji
  - Dynamiczne generowanie stubów oraz skeletonow (od java 1.5)
    - Jesli klient uzywac javy < 1.5 strona serwerowa musi dostarczac stubów
- Wsparcie dla SSL/TSL
  - Standard SSL/TSL Socket Factory Classes