RPC – Remote Procedural Call

Materiały do prezentacji można znaleźć na stronie:

`http://www.houp.info/rpc`

Wprowadzenie

Podstawowe założenia RPC:

- Program uruchamiany na maszynie A może wywołać procedurę innego programu działającego na maszynie B (oczywiście w szczególnych przypadkach może zachodzić $A = B$).
- Z punktu widzenia programisty, wywoływanie procedur zdalnych i lokalnych powinno odbywać się identycznie. Procedury zdalne i lokalne mogą dzielić tą samą przestrzeń nazw i ogólnie mogą być traktowane „tak samo".

- Programista nie musi w ogóle troszczyć się o sprawy związane z siecią (gniazdka, protokoły itp.).

Zalety idei RPC

Pełna realizacja tych idei dała by wiele korzyści:

- Łatwość pisania przenośnych programów rozproszonych / sieciowych. Implementacja komunikacji client–server sprowadzałaby się do prostego wywoływania odpowiednich funkcji.
- Można by dość prosto pisać systemy działające w tzw. środowiskach *heterogenicznych* – używających różne systemy sprzętowe i programowe.
- Przenoszenie starszych aplikacji na nowe standardy sieciowe stałoby się trywialne.

Podstawowe problemy

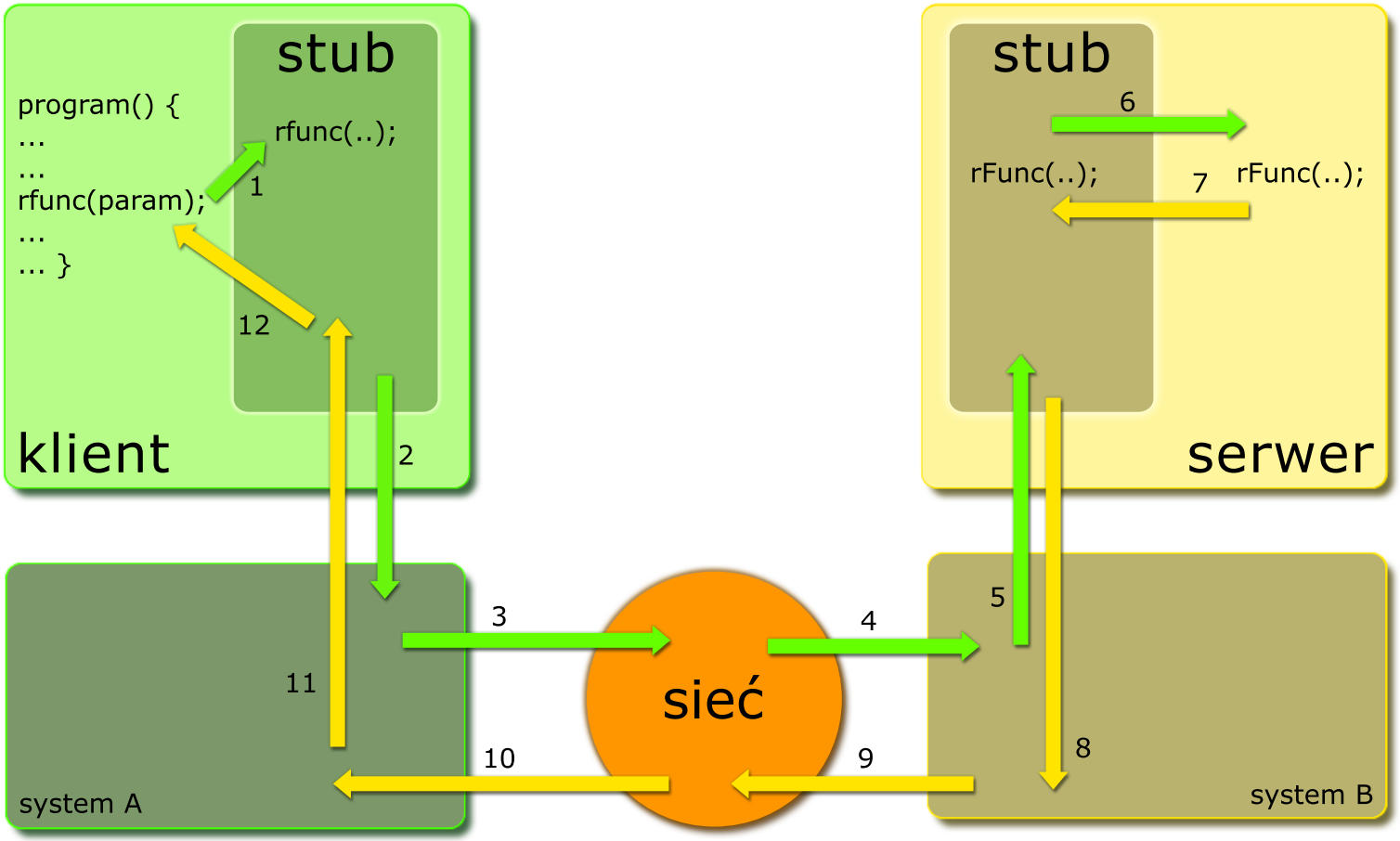
Jak to zwykle bywa, czysta teoria jest zbyt piękna aby była możliwa.

- Istnieje szereg problemów z przekazywaniem argumentów do zdalnych funkcji (chociażby przekazywanie wskaźników, a w szczególności złożonych (dynamicznych) typów danych). Procedury nie mogą również korzystać ze zmiennych globalnych.
- Różne systemy, języki, kompilatory, architektury sprzętowe wymuszają określony format wbudowanych typów danych (zakres liczb, kolejność bajtów) a także wprowadzają wprowadzają różnorakie ograniczenia

ograniczenia na rzutowanie jednego typu na drugi itp.

- Trudno zrealizować zdalne funkcje o zmiennej liczbie argumentów, oraz takie w których zmieniają się typy tych argumentów (np. standardowa funkcja `printf`).
- Odpowiednie zabezpieczenie systemu obsługującego RPC staje się wręcz krytyczne.
- Trudno kwestią jest zagwarantowanie wywołania procedury na serwerze, sprawdzenie czy procedura została rzeczywiście wykonana oraz czy została wykonana tylko raz.
- Oczywiście technologia RPC generuje spore narzuty.

Idea zdalnego wywołania



Przykładowe implementacje/standardy RPC

Tradycyjne implementacje w systemach Unix i Windows:

SUN RPC (znane również jako ONC RPC)

Implementacja wywodzi się z wersji systemu Unix firmowanej przez firmę Sun. Jest to tradycyjny standard RPC w Unixach. Obecnie większość współczesnych systemów unixopodobnych korzysta z niego. (Istnieją również implementacje tego standardu dla systemów Windows.) Najpowszechniejszym zastosowaniem tego standardu jest sieciowy system plików NFS.

DCE/RPC czyli Distributed Computing Environment / Remote Procedure Calls. Standard wprowadzony przez wiele firm (m.in. Apollo – obecnie część HP, IBM, DEC) jako część większego projektu DCE.

MSRPC Jedną z powszechnie używanych implementacji/rozszerzeni standardu DCE/RPC można znaleźć w produktach firmy Microsoft pod nazwą MSRPC. Komponenty działające w technologii DCOM / ActiveX oraz sieciowy system plików SMB, działają na bazie MSRPC. Oczywiście najsłynniejszym „zastosowaniem" MSRPC jest bardzo przydatny program MS-Blaster który pomaga użytkownikom w

szybkim kończeniu swojej pracy. (Istnieją również wolne i przenośne implementacje tego standardu. Jedną z nich jest FreeDCE. Projekty takie jak Samba czy Wine zawierają również swoje implementacje MSRPC.)

Przykładowe implementacje/standardy RPC c.d.

Współczesne (nowoczesne) rozwiązania:

- Na bazie MSRPC działają bardziej złożone rozwiązania, szeroko stosowane na platformie Windows – głównie DCOM.
- Obecnie szeroko stosowne są również różne odmiany technologi CORBA będącej rozszerzeniem i rozwinięciem pomysłu RPC dla języków obiektowych.
- W języku Java dostępny jest mechanizm Java Remote Method Invoke, działający na zasadach podobnych do

RPC.

- W językach działających na platformie .NET dostępna jest podobna technologia .NET Remoting.
- W świecie aplikacji działających w oparciu o sieć Web zastosowanie znajduje standard XML–RPC (który bazuje na przesyłaniu spakowanych w XMLu pakietów poprzez HTTP) oraz SOAP (który w istocie jest rozwinięciem pomysłu z XML–RPC). XML–RPC zostanie omówiony za chwile.

Wszystkie wyżej wymienione technologie są obecnie szeroko stosowane w różnych aplikacjach i systemach.

Sun RPC / ONC RPC

System Sun RPC składa się z:

- Standardu XDR (eXternal Data Representation) który jest dla Sun RPC językiem opisu interfejsu (ang. interface description language – IDL). Służy on do definiowania nagłówków (*stub*) dla zdalnie wywoływanych procedur.
- Programu `rpcgen` który automatycznie generuje nagłówki *stub* dla serwera i klienta. Wynikiem działania programu są odpowiednie pliki źródłowe w języku C. W przypadku server automatycznie generowana jest funkcja `main`.

- Programu (demona) `portmap` który umożliwia klientom „znajdowanie” serwerów (a dokładniej odnajdywanie portów na których słucha serwer).

Założenia Sun RPC

- Wiadomości wysyłane są protokołem TCP lub UDP. (Klient decyduje jakim protokołem chce się komunikować.)
- Klient musi znać adres serwera. Port ustalany jest poprzez kontakt klienta z demonem `portmap` który musi działać na maszynie klienta. Aplikacja serwera „udostępniająca" swoje funkcje zdalnie, również kontaktuje się z lokalnym procesem `portmap` i „rejestruje się".
- Wszystkie dane (wejściowe i wyjściowe) i w ogóle całe wiadomości kodowane są przy użyciu XDR.

Standard XDR

Język XDR służy do definiowania nagłówków funkcji przy użyciu uniwersalnych (zdefiniowanych niezależnie od języka, implementacji i architektury systemu) typów danych. Oto niektóre z typów danych XDR:

- boolean,
- int, hyper (32 i 64 bitowa liczba całkowita big-endian),
- float i double (liczby zmienno–przecinkowe w standardzie IEEE),
- enumeration,

- structure,
- string,
- fixed length array,
- variable length array,
- union,
- void,
- opaque data.

XDR zakłada,że wszystkie dane są przekazywane w blokach o długości będącej wielokrotnością 4 bajtów. Jeśli coś jest mniejsze, na końcu zostają „doklejone” zera. Ponadto w XDR nie ma pojęcia wskaźników jako taki. Są jednak tablice od zmiennym rozmiarze, oraz istnieje pewna ograniczona możliwość definiowania struktur wiązanych[a].

[a] Więcej o tym w RFC1832.

Przykłady

- `01rdate` – serwer udostępnia jedna procedurę, która zwraca wartość `time(NULL)`. Klient na jej podstawie podaje aktualną datę serwera.
- `02ravg` – serwer przyjmuje do 200 liczb zmienno-przecinkowych i wylicza ich średnią arytmetyczną, którą zwraca.

Przykłady są dostępne na stronie `http://www.houp.info/rpc`. Do ich poprawnego funkcjonowania potrzebny jest proces `portmap` działający na maszynie serwera.

XML–RPC

Protokół XML–RPC jest prostym protokołem przeznaczonym głównie dla aplikacji bazujących na Web. Jako „protokół transportu" używany jest standardowy protokół HTTP. Zapytanie XML–RPC jest po prostu odpowiednio sformułowanym wywołaniem metody POST. Treść zapytania klienta zakodowana jest w języku XML. Aplikacja serwera w jakiś sposób musi współpracować z serwerem http (może to być skrypt cgi, lub jakiś moduł serwera, lub nawet odrębny dedykowany serwer). Wynik zwracany przez serwer również jest dokumentem XML.

XML–RPC

Pokażemy jeszcze przykład komunikatów protokołu komunikacyjnego XML–RPC. Oto przykład żądania wywołania metody:

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>
<methodCall>
    <methodName>examples.getStateName</methodName>
        <params>
            <param>
            <value><i4>41</i4></value>
            </param>
        </params>
</methodCall>
```

XML–RPC

Oto przykład możliwej odpowiedzi serwera:

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: UserLand Frontier/5.1.2-WinNT

<?xml version="1.0"?>
<methodResponse>
    <params>
        <param>
            <value><string>South Dakota</string></value>
         </param>
    </params>
</methodResponse>
```

XML–RPC

Oto przykład odpowiedzi serwera, gdy pojawił się błąd:

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 426
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:02 GMT
Server: UserLand Frontier/5.1.2-WinNT

<?xml version="1.0"?>
<methodResponse><fault><value>
            <struct><member>
                    <name>faultCode</name>
                    <value><int>4</int></value>
                </member>
                <member>
                    <name>faultString</name>
                    <value><string>Too many parameters.
                    </string></value>
                </member>
            </struct>
</value></fault></methodResponse>
```

Koniec

Dziękuję za uwagę.