PA160
Distribuované aplikace – základní protokoly
(Programování distribuovaných systémů)

Materiál dle `http://www.cee.hw.ac.uk/courses/5nm1/`

RPC

- RPC – Remote Procedure Call
  - Tváří se jako „obyčejná“ procedura
  - Tělo je vykonáno v jiném procesu (na jiném stroji]
- Důvody zavedení
  - Vyšší úroveň abstrakce než sokety
  - Nezávislé na architektuře či operačním systému
  - Předávání jednoduchých i složitých datových typů
- Vyžaduje podporu (jmenné služby, bezpečnost, . . .)

RPC Standardy

- Tři základní
  - ONC (Open Network Computing)
  - DCE (Distributed Computing Environment)
  - Microsoftí COM/DCOM „standard“

RPC Standardy – popis

- ONC (definováno v RFC 1831)
  - vyvinuto SUNem
  - nejrozšířenější, dostupné prakticky na všech operačních systémech
- DCE (`http://www,opengroup.org/dce/`)
  - Rovněž široce dostupné
  - Zahrnuje i adresářové služby
- COM (`http://www.microsoft.com/com/`
  - Proprietární
  - V současné době rozšíření DCE, původně componentní model (jeden počítač)

ONC RPC

- RPC jedinečně definován:
  - *Číslem programu*: skupina procedur
  - *Číslem verze*: verze
  - *Číslem procedury*: jedinečný identifikátor konkrétní procedury
  - Uživatel může přiřazovat čísla programů v rozsahu `0x20000000--0x3FFFFFFF` (20 milionů čísel)

ONC – části

- Clientská část (příklad)
  - `clnt_create()`: de facto napojení na skupinu procedur (server)
  - `clnt_call()`: zavolání konkrétní procedury
  - `clnt_destroy()`: odpojení se
- Serverová část (příklad)
  - `svc_register()`: registruj skupinu u `portmapperu`
  - `svc_getargs()`: převezmi argumenty (XDR zakódované)
  - `svc_sendreply()`: zašli výsledek volání (XDR)
  - `svc_unregister()`: zruš registraci

Portmapper

- Zprostředkuje přístup ke službám
  - Naslouchá na portu 111
  - Udržuje mapu programů a odpovídajících čísel portů
  - Použití
    - ∗ 1. Server se zaregistruje na portmapperu
    - ∗ 2. Client získá aktuální číslo portu konkrétní služby od portmapperu (dotaz na portu 111)
    - ∗ 3. Client se spojí se serverem

ONC API – tvorba klienta

```
CLIENT *
clnt_create(host, prog, vers, prot)
     char *host;      -- hostname
     u_long prog;     -- program number
     u_ong vers;      -- version number
     char *prot;      -- protocol (udp, tcp, unix)
```

- Vrací *handle* na skupinu procedur

ONC API – registrace serveru

```
extern bool_t
svc_register(xprt, prog, vers, dispatch, protocol)
     SVCXPRT *xprt;
     u_long prog;
     u_long vers;
     void (*dispatch)();
     u_long protocol; //tcp or udp, zero means do not register
```

- Registruje *program* a jeho *verzi*
- Vrací `true` při úspěchu

ONC API – volání klienta

```
enum clnt_stat

CLNT_CALL(rh, proc, xargs, argsp, xres, resp, timeout)

     CLIENT *rh;  -- handle from clnt_create

     u_long proc; -- registered procedure

     xdrproc_t xargs; -- XDR to encode input

     caddr_t argsp;   -- address of input agr

     xdrproc_t xres;  -- XDR to decode output

     caddr_t resp;    -- address of result buffer

     struct timeval timeout; -- timeout
```

XDR

- Problém s odlišnou architekturou komunikujících počítačů
  - Pořadí bytů, typy čísel, řetězce, enumerované typy
- XDR: eXchange Data Representation funkce
  - Kódují a dekódují data
  - Konkrétní implementace je závislá na architektuře
  - Zajišťují strojovou nezávislost při výměně dat

Vestavěné XDR konverzní funkce

Používány v `clnt_call()`
Jednoduché funkce (příklady)

```
xdr_int()   xdr_char()  xdr_u_short     xdr_bool()
xdr_long()  xdr_u_int() xdr_wrapstring()
xdr_short() xdr_enum()  xdr_void()
```

K dispozici jsou i agregační funkce (příklady)

```
xdr_array()  xdr_string()  xdr_union()
xdr_vector() xdr_opaque()
```

Rovněž je možno definovat uživatelské funkce umožňující manipulovat s nepředdefinovanými typy

Generování kódu pro použití RPC

- Nevýhody nativního RPC
  - Složité API, těžko laditelné
  - Použití XDR složité

  Většinou se používá stále stejným způsobem

- Řešení: použití generátoru kódu (`rpcgen`).
  Generuje:
  - Hlavičkový soubor
  - Potřebné XDR funkce
  - *stubs* na straně klienta i serveru
- `rpcgen` pro C/C++, `jrpcgen` pro Javu

Chování RPC

- Neúspěchy
  - Nelze nalézt server
  - Ztracený požadavek
  - Ztracený výsledek
  - Zhroucení serveru (po přijetí požadavku)
  - Zhroucení klienta

Ztracený požadavek

- Používání timeoutů
- Používání retransmise
  - at least once: alespoň jednou
  - at most once: nejvýše jednou
  - exactly once: právě jednou
  - no guarantees: bez záruky o počtu retransmisí

Ztracená odpověď

- Rozlišení typu požadavků
  - Idempotentní: je možno zopakovat (bez vedlejších efektů)
    - ∗ Ztráta řešena prostou retransmisí
  - Neidempotentní: má vedlejší efekty
    - ∗ Retransmise musí být zpracována serverem
      - · Rozezná retransmisi
      - · Neopakuje tělo
      - · Vydá výsledek, pouze pokud jej má v cache

Zhroucení serveru

- Pořadí požadavek versus výpadek
- Výpadek před zpracováním požadavku
  - Retransmise
- Výpadek před odesláním odpovědi
  - Vrátí chybu
- Client není schopen zjistit příčinu

Zhroucení klienta

- Procedura zůstává na serveru
  - zmařený výkon CPU
  - drží zdroje (např. zamčené soubory)
  - může odpovídat „podruhé“ po rebootu klienta
- Možné řešení: *soft služby* (nikoliv součást RPC)

RPC – literatura

- Douglas E. Comer: Computer Networks and Internets, Prentice Hall, 2001 (3. vydání); kapitola 33
- SUN developers guide: `http://docs.sun.com/?q=ONC+RPC`
- Přednášky:

  `http://www.cee.hw.ac.uk/courses/5nm1/index.htm`

Distribuované objekty

- DO – distribuce enkapsulovaných dat na počítačích v síti
  - Pouze data (funkce nejsou přenositelné) – distribuovaná enkapsulovaná data
  - Skutečně přenositelné funkce (Java)
- Složené dokumenty
  - Data plus prohlížeč či editor
  - Původně pouze pro výměnu dat mezi jednotlivými aplikacemi
  - Později přidána „distribuce“
- *Distribuované objekty* nebo *distribuované komponenty* (data plus funkce)

Remote Method Invocation

- Objektová varianta RPC
  - Data patří *instanci* objektu
  - Implementace závisí na třídě objektu
- Pro připomenutí: RPC volá konkrétní proceduru
  - privátní data sdílena
  - vždy stejná procedura

Klient server technologie

- Klasické technologie *klient-server* na aplikační úrovni:
  - HTTP a CGI
    - pomalé (oddělené procesy)
    - bezstavové (problém se seancemi)
  - COM
    - Proprietární Microsoft
    - Nemá persistentní objekty ani distribuované jmenné služby
    - Nefunguje mimo MS Windows
  - OO přístupy