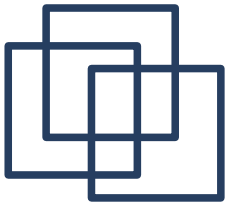
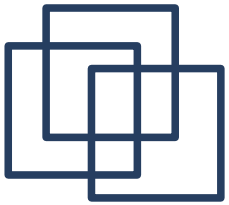


Wprowadzenie do programowania rozproszonego

1. Koncepcja programowania rozproszonego.
2. Model proceduralny.
3. Zdalne wywoływanie procedur.
4. Specyfikacja Sun RPC.

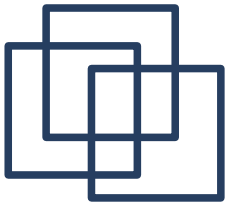


Programowanie rozproszone

Przystępując do tworzenia aplikacji rozproszonej projektant ma do wyboru dwa różne podejścia:

• **projektowanie z punktu widzenia komunikacji między procesami** (*communication-oriented design*) główny nacisk kładzie na protokół komunikacyjny. W drugiej kolejności tworzone są programy klienta i serwera reagujące na zgłoszenia i wysyłające dane zgodnie z protokołem.

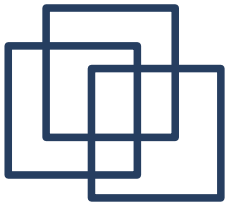
• **projektowanie z punktu widzenia aplikacji** (*application-oriented design*) rozpoczyna się od stworzenia aplikacji rozwiązującej problem. Po przetestowaniu jest ona dzielona na dwie lub więcej części. Na tym etapie dobudowuje się protokoły komunikacyjne.



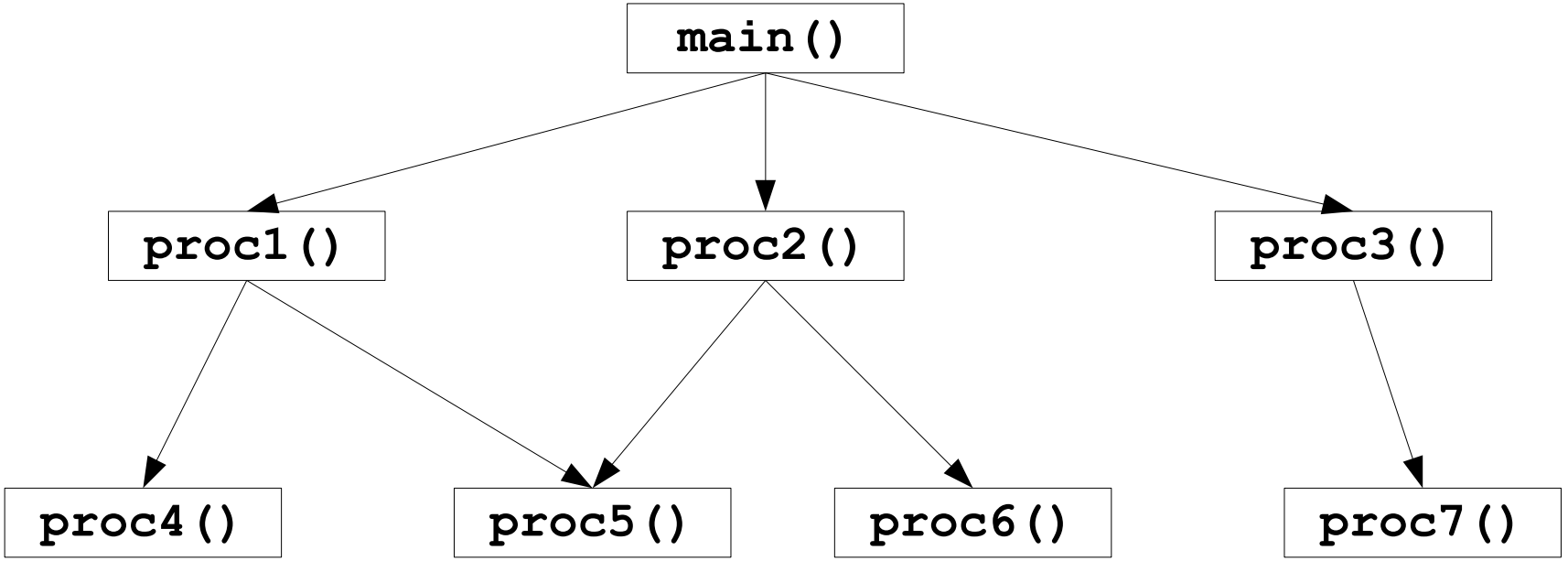
Model zdalnego wywołania procedury

Aby ułatwić projektowanie z punktu widzenia aplikacji opracowano model pojęciowy nazwany **modelem wywołania procedury** – **RPC** (*Remote Procedure Call*)

Model RPC umożliwia projektantowi (programiście) skupienie uwagi na samej aplikacji. Projektant może opracować zwykły, tradycyjny program
rozwiązujący zadany problem, a dopiero potem podjąć próbę rozdzielenia go na części wykonywane na różnych komputerach.

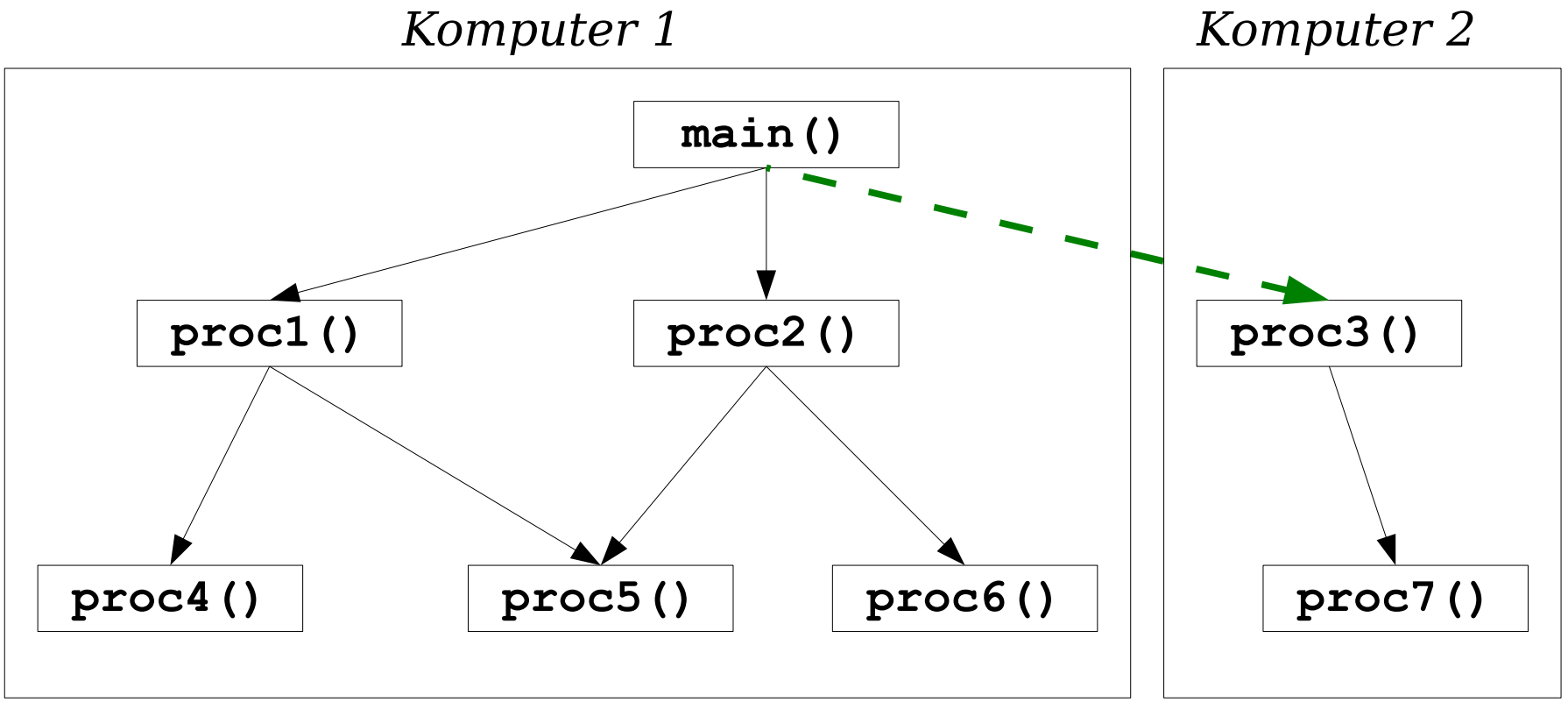


Model wywołań procedur

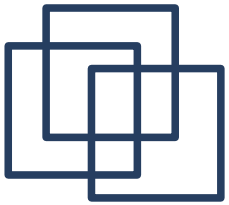


Program składa się z jednej lub wielu procedur, zazwyczaj zorganizowanych w hierarchię wywołań. Strzałka od procedury *n* do procedury *m* oznacza wywołanie *m* z wnętrza procedury *n*.

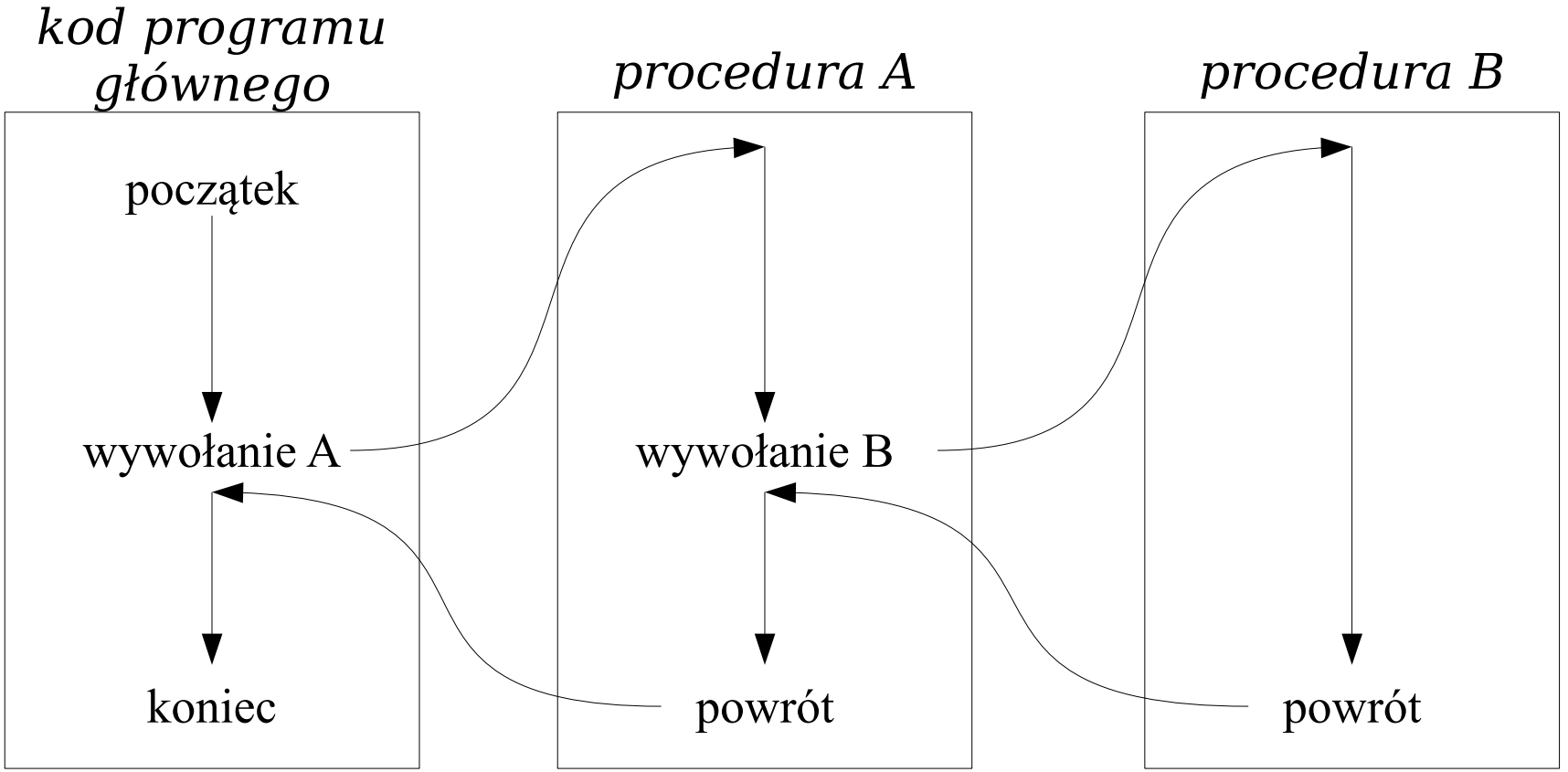
Rozszerzenie modelu proceduralnego



Program rozproszony wykorzystujący model zdalnego wywołania procedury. Linia podziału przebiega pomiędzy programem głównym a procedurą nr 3. Implementacja zdalnego wywołania wymaga użycia protokołu komunikacyjnego.



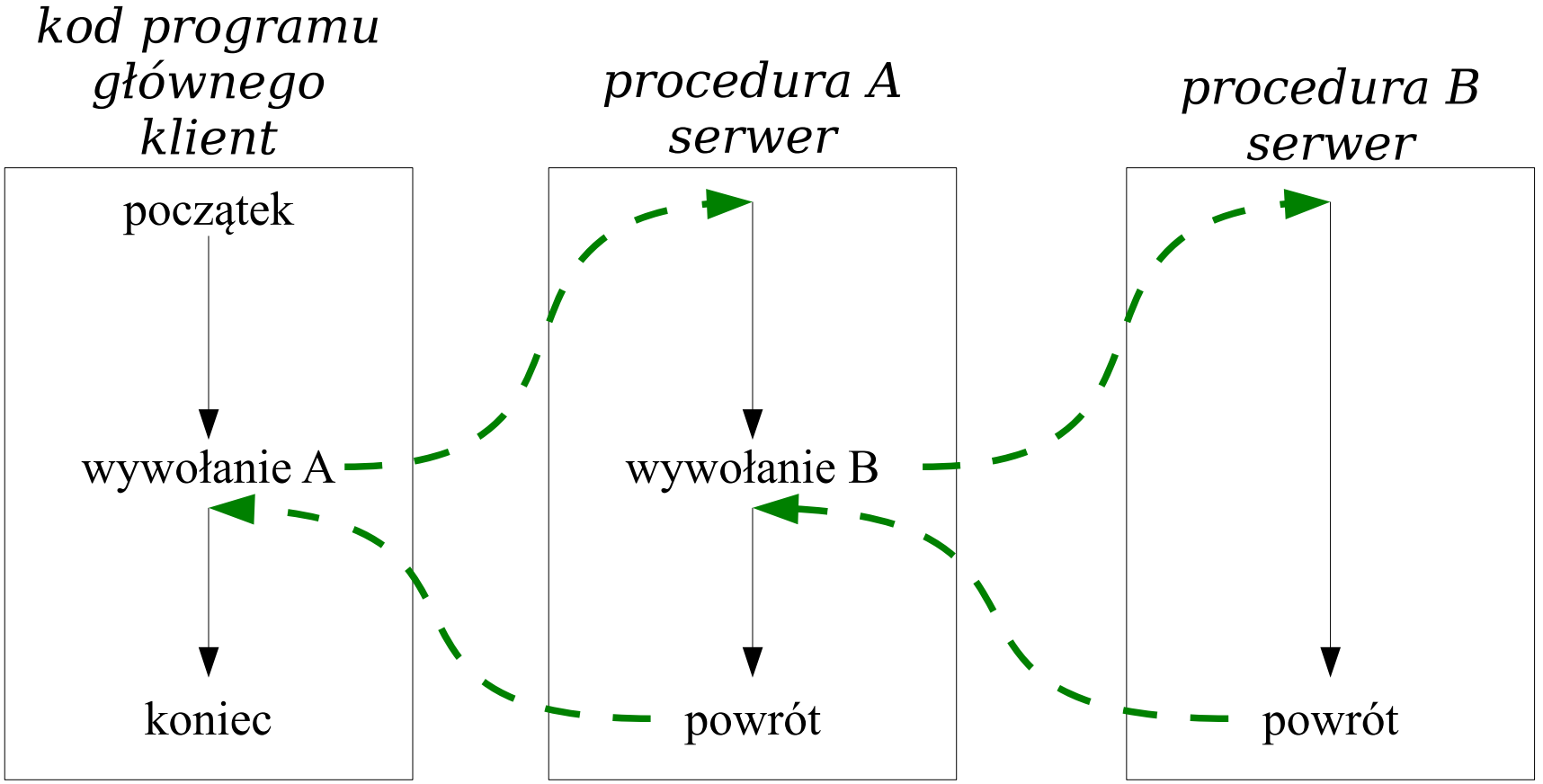
Przebieg wykonania procedury



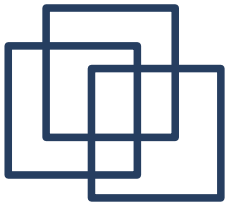
Przepływ sterowania w sytuacji wywołania procedury i powrotu.

Wykonanie programu biegnie przez program główny, następnie przez procedury *A* oraz *B*, po czym wraca do programu głównego.

Przebieg wykonania zdalnej procedury



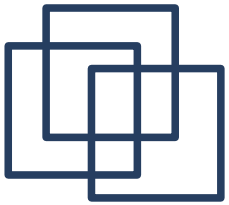
Przepływ sterowania w sytuacji wywołania **zdalnej** procedury. Linie przerywane pokazują przepływ sterowania między klientem i serwerem przy wywołaniu zdalnej procedury i powrocie z niej.



Model klient-serwer a RPC

Wywołanie odległej procedury odpowiada wysłaniu “zapytania“ do serwera. Odpowiedź serwera jest analogiem powrotu z procedury. W warunkach idealnych przebieg zdalnego wywołania procedury byłby identyczny jak wywołania lokalnego. Są jednak przypadki, które to ograniczają:

- duża czasochłonność wywołania zdalnego,
- brak możliwości przekazywania wskaźników,
- brak wspólnego środowiska pracy np. deskryptory wejścia – wyjścia, operacje systemu operacyjnego.

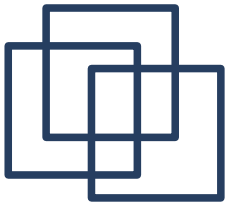


Mechanizm Sun RPC

Sun RPC [RFC 1057] definiuje mechanizm realizujący model zdalnego wywołania procedury. Specyfikacja określa:

- format komunikatów wysyłanych przez program wywołujący (klienta),
- format argumentów wywołania oraz format zwracanych wyników,
- protokół warstwy transportowej – TCP lub UDP,
- kodowanie przesyłanych informacji – XDR,

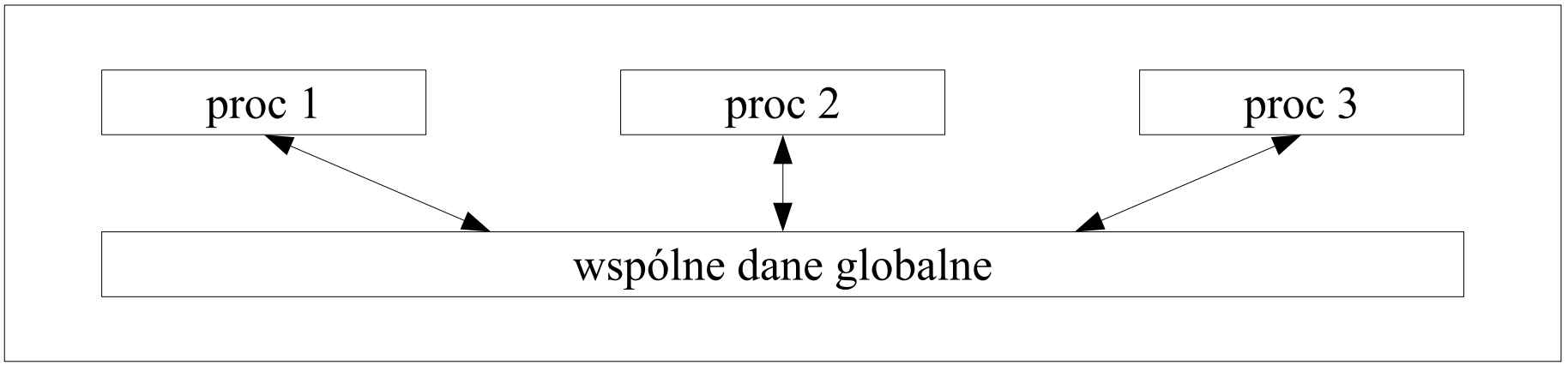
Dla Sun RPC zdefiniowano również system kompilacji, który umożliwia automatyzację konstruowania programów rozproszonych.

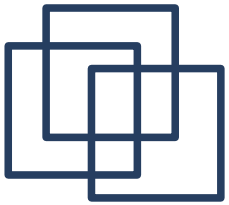


Zdalnie wywoływane programy i procedury

Program odległy jest odpowiednikiem serwera i zawiera jedną lub więcej procedur odległych oraz dane globalne. Do obszaru danych globalnych mają dostęp wszystkie procedury działające w ramach jednego programu.

Jeden zdalnie wywoływany program

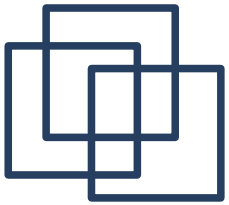




Identyfikacja zdalnie wywoływanych procedur

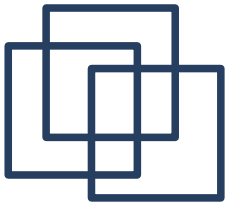
Standard Sun RPC wymaga przypisania każdemu programowi odległemu jednoznacznego 32-bitowego identyfikatora. Ponadto każdej procedurze należącej do danego programu przydziela się liczbę całkowitą. Procedury są numerowane sekwencyjnie: 1, 2, ... , N. Tak więc każdą procedurę odległą można zidentyfikować podając parę liczb.

Specyfikacja przewiduje również możliwość rozróżnienia różnych wersji wywoływanego programu zdalnego.



Przekazywanie argumentów

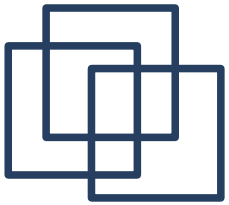
W większości języków programowania argumenty wywołania procedur podaje się w notacji pozycyjnej. Duża liczba argumentów powoduje zmniejszenie czytelności programu. Tę niedogodność można usunąć zbierając wiele argumentów w jedną strukturę Po powrocie do procedury wywołującej z tej struktury można wyodrębnić też wartości wynikowe zwrócone przez procedurę wywoływaną.



Wzajemne wykluczanie procedur

Mechanizm Sun RPC nie pozwala na jednoczesne wywołanie więcej niż jednej procedury w ramach jednego, zdalnie wywoływanego programu. Dopóki trwa wykonanie jednej z nich, system blokuje następne wywołania. Jest to szczególnie istotne w programach wykorzystujących wspólne dane dla kilku różnych procedur.

13



Semantyka wywołań a protokół komunikacyjny

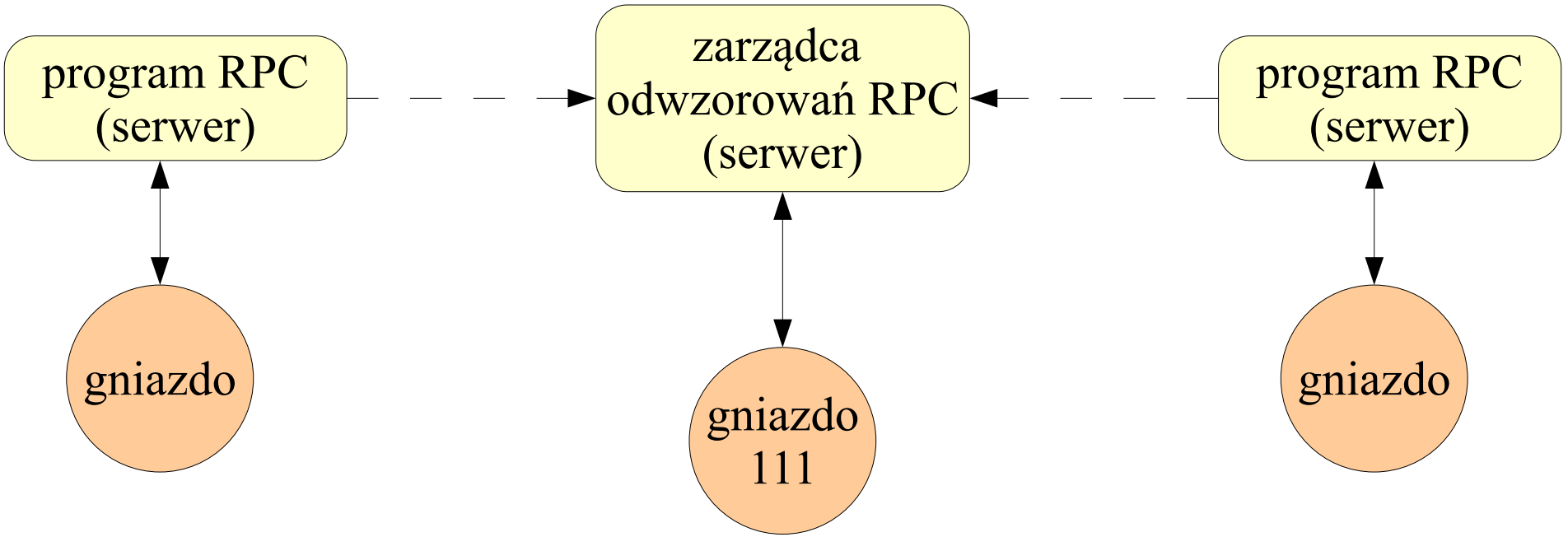
Standard Sun RPC nie gwarantuje niezawodności zdalnego wywołania w przypadku, gdy używanym protokołem warstwy transportowej jest UDP.

Jeśli program wywołujący procedurę nie otrzyma odpowiedzi oznaczającej powrót po jej wykonaniu to procedura wykonała się **zero lub więcej** razy. Nadejście odpowiedzi oznacza natomiast, że procedura została wykonana **co najmniej jeden** raz.

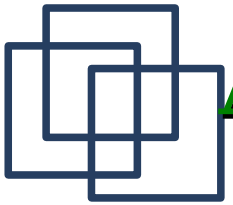
Jeśli aplikacja korzystająca z mechanizmu Sun RPC używa protokołu UDP to trzeba ją zbudować tak, aby była odporna na konsekwencje tych własności.

Przykładowo, wielokrotne i jednokrotne wykonanie zdalnej procedury powinno dać takie same wyniki - **warunek idempotencji**.

Numer programu a numer portu



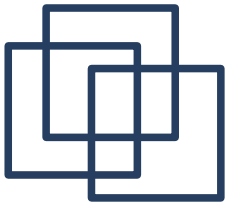
Każdy program RPC rejestruje swój numer i numer używanego portu u zarządcy odwzorowań RPC. Program klienta kontaktuje się z programem zarządcy działającym na porcie 111, aby otrzymać numer portu określonego programu zdalnego.



Algorytm programu zarządcy

1. Utwórz gniazdo bierne, związane z portem 111.
2. Przyjmuj kolejne żądania zarejestrowania numeru portu programu RPC lub odszukania numeru portu na podstawie podanego numeru programu RPC.

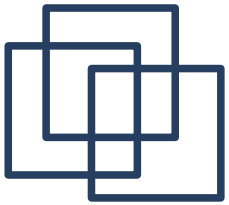
Żądanie rejestracji jest zgłaszane przez programy działające lokalnie. Każde zgłoszenie ma postać pary składającej się z numeru programu RPC i numeru portu. Po otrzymaniu żądania zarządca dopisuje otrzymaną parę do **bazy odwzorowań**.
Żądania odszukania numeru portu mogą być nadsyłane z dowolnego komputera. Każde zgłoszenie zawiera numer programu RPC na podstawie którego zarządca wyszukuje w **bazie odwzorowań** numer portu.



Format komunikatów Sun RPC

Protokół RPC korzysta ze standardu przesyłania danych XDR. Nagłówek RPC zawiera pole typu komunikatu, które służy do odróżnienia komunikatów używanych używanych przez klienta do zainicjowania zdalnego wywołania procedury od komunikatów używanych przez serwer do przesłania odpowiedzi.

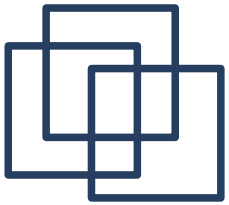
```
enum msg_type{ // typ komunikatu RPC
    CALL = 0;
    REPLY = 1;
};
```



Format komunikatów Sun RPC

Pojedynczy komunikat RPC, w zależności od jego typu ma następującą ogólną postać:

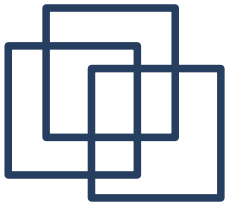
```
struct rpc_msg{
   unsigned int mesgid;
   union switch(msg_type mesgt){
      case CALL:
         call_body cbody;
      case REPLY:
         rply_body rbody;
   }
};
```



Format komunikatów Sun RPC

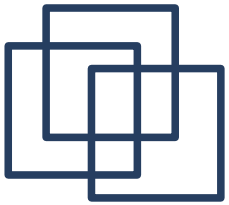
W przypadku komunikatu inicjującego, dane są umieszczane w strukturze **`struct call_body`**:

```
struct call_body {
   unsigned int rpcvers;   /* wersja RPC - zwykle 2 */
   unsigned int prog;      /* numer programu */
   unsigned int vers;      /* wersja programu */
   unsigned int proc;      /* numer procedury */
   opaque_auth cred;       /* dane uwierzytelniające
                              klienta */
   opaque_auth verf;       /* informacja do weryfikacji
                              tożsamości /*
   /* argumenty dla procedury odległej */
};
```



Serializacja argumentów zdalnej procedury

Argumenty dla zdalnie wywoływanych procedur muszą być przekształcone do postaci pozwalającej na przesłanie ich między komputerami. Zabieg kodowania argumentów polega na **szeregowaniu** (*marshal*), **serializacji** (*serialize*) lub **linearyzacji** (*linearize*) argumentów. Pomimo, że protokół RPC umożliwia przesyłanie w ten sposób złożonych struktur w zdalnych wywołaniach procedur to należy pamiętać, że serializacja i deserializacja dużych struktur może wymagać czasochłonnych operacji przetwarzania danych. W związku z tym przy projektowaniu oprogramowania rozproszonego należy unikać przekazywania przez sieć złożonych struktur danych.



Format komunikatów Sun RPC

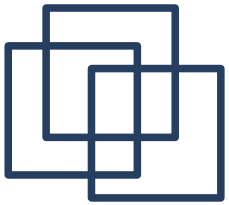
Format odpowiedzi serwera jest uzależniony od tego, czy zgłoszenie zostało zaakceptowane, czy nie. Do opisu tego zdarzenia służy typ.

```
enum reply_stat{
    MSG_ACCEPTED = 0,
    MSG_DENIED   = 1
};
```

Jeśli zgłoszenie zostanie zaakceptowane należy przekazać informację o próbie wywołania zdalnej procedury.

```
enum accept_stat {
    SUCCESS = 0,          /* RPC uruchomiona pomyślnie */
    PROG_UNAVAIL = 1,     /* brak programu zdalnego */
    PROG_MISMATCH = 2,    /* niepoprawna wersja */
    PROC_UNAVAIL  = 3,    /* niedostępna procedura */
    GARBAGE_ARGS  = 4     /* niewłaściwe argumenty */

};
```



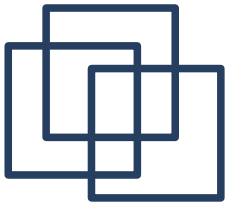
Format komunikatów Sun RPC

Powody odrzucenia zgłoszenia określa typ:

```
enum reject_stat {
   RPC_MISMATCH = 0, /* nieobsługiwana wersja RPC */
   AUTH_ERROR = 1    /* nie można autoryzować klienta */
};
```

Przyczyna niepowodzenia autoryzacji jest opisana typem:

```
enum auth_stat {
   AUTH_BADCRED = 1,        /* zły identyfikator */
   AUTH_REJECTEDCRED = 2,   /* klient musi rozpocząć nową
                               sesję */
   AUTH_BADVERF = 3,        /* zły weryfikator */
   AUTH_REJECTEDVERF = 4,   /* weryfikator nieaktualny */
   AUTH_TOOWEAK = 5         /* odrzucony ze względu na
                               polisę bezpieczeństwa */
};
```

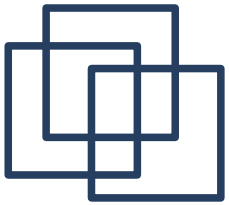


Format komunikatów Sun RPC

Format odpowiedzi serwera jest opisany strukturą:

```
union reply_body switch (reply_stat stat){
   case MSG_ACCEPTED:
      accepted_reply areply;
   case MSG_DENIED:
      rejected_reply rreply;
} reply;
```

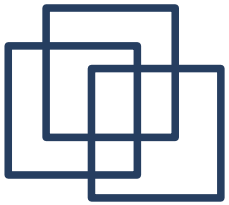
Zwrócenie odpowiedzi pozytywnej **MSG_ACCEPTED** nie jest równoważne z wykonaniem zdalnej procedury. Oznacza jedynie zaakceptowanie zgłoszenia przez serwer.



Format komunikatów Sun RPC

Jeśli żądanie zostało zaakceptowane serwer odsyła informację postaci:

```
struct accepted_reply{
   opaque_auth verf;
   union switch (accept_stat stat){
      case SUCCESS:
         opaque results[0];
         /* wyniki zwracane przez procedurę */
      case PROG_MISMATCH:
         struct{
            unsigned int low;
            unsigned int high;
         } mismatch_info;
      default:
         void; /* przypadek uwzględnia wszystkie pozostałe
                  błędy /*
   } reply_data;
};
```

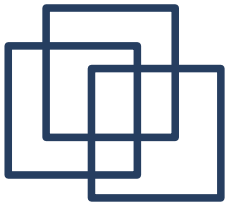


Format komunikatów Sun RPC

Jeśli żądanie nie zostało zaakceptowane serwer odsyła informację postaci:

```
union rejected_reply switch (reject_stat stat){
   case RPC_MISMATCH:
      struct {
         unsigned int low;
         unsigned int high;
      } mismatch_info;
   case AUTH_ERROR:
      auth_stat stat;
};
```

Parametry **low** i **high** oznaczają najniższy i najwyższy numer wersji zdalnego programu dostępnej na serwerze.

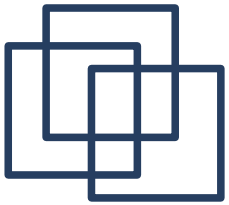


Sprawdzanie tożsamości nadawcy

Standard RPC przewiduje cztery typy kontroli tożsamości nadawcy wywołania. Są one wyszczególnione w poniższej deklaracji:

```
enum auth_flavor{
   AUTH_NULL = 0,          /* brak kontroli tożsamości */
   AUTH_UNIX = 1,          /* identyfikacja UNIX'owa */
   AUTH_SHORT = 2,         /* skrócona kontrola używana
                              w dalszej komunikacji */
   AUTH_DES = 3            /* kontrola oparta na standardzie
                              DES */
};

struct opaque_auth{
   auth_flavor aflavor; /* typ kontroli
   opaque body<400>;    /* dane dla typu kontroli */
};
```



Sprawdzanie tożsamości nadawcy

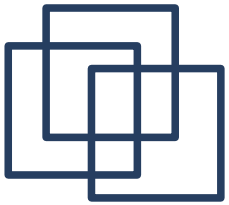
W przypadku kontroli opartej na autoryzacji systemu UNIX pole

body<400> zawiera następującą strukturę:

```
struct auth_unix {
   unsigned int stamp;      /* czas nadania */
   string machinename<255>;/* nazwa komputera - nadawcy
                              */
   unsigned int uid;        /* identyfikator użytkownika */
   unsigned int gid;        /* główna grupa użytkownika */
   unsigned int gids<16>;   /* inne grupy użytkownika */
};
```

Podstawowe wady:

- konwencja zbyt ściśle związana z systemem UNIX,
- brak uniwersalnej przestrzeni nazw oraz identyfikatorów,
- brak możliwości weryfikacji autoryzacji.



Sprawdzanie tożsamości nadawcy - DES

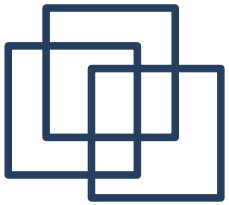
Najbardziej zaawansowany mechanizm kontroli wykorzystuje algorytm DES.

Typ identyfikatora użytkownika określa struktura:

```
enum authdes_namekind {
   ADN_FULLNAME = 0, /* pełna nazwa (max. 255 znaków) */
   ADN_NICKNAME = 1  /* skrót - liczba całkowita */
};
```

natomiast zbiór zaszyfrowanych jest typu:

```
typedef opaque des_block[8];  /* 64-bitowy blok
                                 szyfrowanych danych */
```



Sprawdzanie tożsamości nadawcy - DES

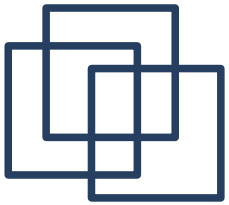
Autoryzacja z wykorzystaniem pełnej nazwy użytkownika

wykorzystuje strukturę:

```
struct authdes_fullname{
   string name<255>;     /* nazwa klienta */
   des_block key;        /* zakodowany klucz */
   opaque window[4];     /* zakodowane okno – określa czas
                            ważności dla autoryzacj */
};
```

Autoryzacja odbywa się za pośrednictwem struktury:

```
union authdes_cred switch (authdes_namekind nkind){
   case ADN_FULLNAME:
      authdes_fullname adc_fullname;
   case ADN_NICKNAME:
      int adc_nickname;
};
```

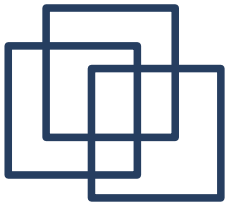


Sprawdzanie tożsamości nadawcy - DES

Serwer odpowiada przesyłając poniższą strukturę:

```
struct authdes_verf_svr{
   des_block adv_timeverf; /* zakodowany weryfikator */
   int adv_nickname;       /* nickname dla dalszej
                              transmisji */
};
```

Weryfikator zawiera **timestamp** otrzymany od klienta pomniejszony o jedną sekundę. Dodatkowo zwracany jest identyfikator, który będzie wykorzystywany do dalszej interakcji pomiędzy klientem i serwerem.



Podsumowanie

Model zdalnego wywołania procedury ułatwia projektowanie programów rozproszonych. Procedury zdalnie wywoływane podobnie jak procedury zwykłe, przyjmują argumenty i zwracają wyniki. Opracowana przez firmę SUN specyfikacja modelu RPC wykorzystuje model XDR do przesyłania danych przez sieć TCP/IP.