# System komunikacji międzywęzłowej *ICS*

Projekt z przedmiotu RSO

Maciej Remiszewski
M.Remiszewski@elka.pw.edu.pl

17 czerwca 2005

## Spis treści

## 1 Wstęp

Klaster OpenSSI jest, jak sama nazwa wskazuje, rozwiązaniem typu *Single System Image*. Użytkownik widzi jeden, spójny obraz systemu, mimo tego, że każdy węzeł jest obsługiwany przez osobne jądro. Dzieję się tak dlatego, że w tajemnicy przed wszystkimi użytkownikami, jądra potajemnie komunikują się ze sobą. Aby zapewnić komunikację na odpowiednio niskim poziomie, powstał dedykowany system komunikacji międzywęzłowej ICS. Dzieli się on na dwie warstwy. Wysoka warstwa udostępnia interfejsy usług, podczas gdy warstwa niska zajmuje się fizyczną realizacją komunikacji.

ICS jest częścią projketu *Cluster Infrastructure (CI)*, który ma na celu stworzenie powrzechnej infrastruktury klastrującej dla rozwiązań opartych na systemie Linux. Bliższe informacje, jak również kompletną specyfikację techniczą podsystemu ICS można znaleźć na stronie *http://ci-linux.sourceforge.net/*.

## 2 Paradygmaty komunikacji

System *ICS* można najogólniej scharakteryzować trzema cechami:

- Realizowany jest model klient/serwer.
- Na wysokim poziomie, *klient* ma do dyspozycji mechanizm wiadomość/odpowiedź, bądź też niezawodny mechanizm przekazywania wiadomości.
- Na wysokim poziomie, *serwer* odbiera wiadomości od *klienta* i ewentualnie wysyła odpowiedź.

Taki model umożliwia efektywną implementację obydwu bazowych typów komunikacji: przekazywania wiadomości i zdalnego wywoływania procedur.

Zarówno wiadomości jak i odpowiedzi ICS składają się z:

- stałej wielkości bufora *in-line* (*il*)
- żadnego, bądź wielu buforów *out-of-line* (*ool*) o nieograniczonej długości

Wielkość bufora *il* ustalona jest stałą *ICS_MAX_INLINE_DATA_SIZE*, która w przypadku OpenSSI ma wartość 300 (bajtów). Z koleji liczebność buforów *ool* ograniczona jest do 7, bez limitu długości poszczególnych buforów. Budowa interfejsów umożliwia efektywną implementację transferu danych *ool* przez niższą warstwę. Może on być realizowany przy pomocy różnych protokołów. W OpenSSI dane *ool* są po prostu załączane za buforem *il*.

Argumenty danych (zarówno *il* jak i *ool*) mogą być dwojakiego rodzaju: *Argumenty wejściowe* onoszą się do danych wysyłanych do serwera (wiadomości, lub wywołań RPC), natomiast *Argumenty wyjściowe* do odpowiedzi wysyłanych do klienta (dotyczy tylko RPC).

Aby klient mógł wysłać wiadomość (lub czekać na odpowiedź) oraz by serwer mógł obsłużyć rządanie klienta, potrzebne są *uchwyt*. Są to struktyry danych, przezroczyste dla szkieletu generowanego przez *icsgen*, które mają następujące własności:

- zawierają stan transakcji (długość wiadomości w bajtach, czy została wysłana etc.)
- zawierają stałej wielkości bufor *il* opisany wyżej.
- zawierają informację o buforach *ool* będących aktualnie w użyciu.

Wiele z metod w interfejsie korzysta z *callback'ów*. Są to metody wywoływane przez niższy poziom ICS, w reakcji na określone zdarzenia. Dokładny opis można znaleźć w odpowiednich specyfikacjach. Warto jednak zwróć uwagę na argument *callarg*. Jest to argument interfejsu, który istnieje jedynie po to, by być użytym jako argument callback'u.

# 3 Generacja kodu przy użyciu *icsgen*

Na najwyższym poziomie, interfejs pomiędzy ICS a innymi komponentami CI/SSI z niego korzystającymi jest zbiorem usług, z których każda oferuje zestaw dobrze określonych operacji. To właśnie te usługi składają się na interfejs podsystememu ICS. Definiuje się je przy użyciu specjalnego języka, który w czasie kompilacji tłumaczony jest na kod wiążący lokalny komponent z ICS, klienta ICS i serwerem ICS, a w ten sposób komponent ze zdalną usługą. Procesem odpowiedzialnym za generację kodu jest *icsgen*.

*icsgen* analizuje pliki z definicją usługi i na ich podstawie generuje dalsze pliki źródłowe:

- nagłówki z prototypami funkcji dla ICS
- nagłówki z makrami wywołującymi usługi obsługiwane przez ICS
- szkielet kodu mapujący operacje pomiędzy warstwami ICS i pomiedzy węzłami (chodzi tu głównie o reprezentację danych)
- tablice wywołań serwera, używane wewnętrznie przez ICS

## 3.1 Usługi

Każda usługa musi zostać zadeklarowana w podsystemie ICS za pomocą pliku (lub plików) identyfikujących usługę w systemie oraz zawierających definicje operacji dostępnych w ramach tej usługi. Zgodnie z konwencją, pliki z definicjami powinno się nazywać *nazwa_usługi.svc*.

Definicje serwisów zapisane są w pliku nagłówkowym *cluster/ics.h* który mapuje usługi na kanały komunikacyjne podsystemu ICS. Wiele usług można zgrupować i udostępnić poprzez współdzielony kanał jako podusługi. Wówczas będą one miały wspólne zarządzanie przepływu.

## 3.2 Operacje

Każda operacja dostępna w ramach usługi ICS musi być dokładnie zdefiniowana. Dla każdej z nich trzeba zadeklarować paramtery i ich atrybuty. Opis zarówno definiuje sam interfejs, jak również ustala w jaki sposób ICS obsługiwał będzie dane przekazywane pomiędzy węzłami.

*icsgen* przetwarza wstępnie definicje usług za pomocą preprocesora C. W definicjach można zatem korzystać z wbudowanych dyrektyw preprocesora, definiować makra oraz wstawiać komentarze, zgodnie ze składnią C.

## 3.3 Kodowanie danych

Przekazywanie danych wymaga dokładnego zdefiniowania typu danych. Na poczet transmisji danych pomiędzy warstwami ICS oraz węzłami klastra, wszystkie dane są kodowane, by mogły być przekazywane poprzez bufory *il* i *ool*. Dla każdego używanego typu danych muszą zostać zdefiniowane metody do:

- zakodowania danych po stronie klienta (*cli_encode_typ()*)
- zdekodowania danych po stronie klienta (*cli_decode_typ()*)
- zakodowania danych po stronie serwera (*srv_encode_typ()*)
- zdekodowania danych po stronie serwera (*srv_decode_typ()*)

# 4 Kanały komunikacji ICS

Komunikacja komponentów CI/SSI z sytemem ICS następuje w oparciu o usługi. Wiadomości i odpowiedzi wysyłane są poprzez dwukierunkowe kanały komunikacyjne zaimplementowanych poprzez TCP. Każda wiadomość i odpowiedź jest mapowana na określony kanał. Z jednym kanałem moze być związanych wiele usług (mapowanie usług na kanały zapisane jest w pliku *cluster/ics/ics_conf.c*). Usługi grupowane są na kanałach, ponieważ to kanały komunikacyjne są bytami na których działa *kontrola przepływu* (*flow control* lub *throttling*). Kontrola przepływu realizowana jest przez zatrzymymanie komunikacji przez dany kanał. Dzieje się tak wtedy, gdy po stronie serwer bądź klienta kończą się zasoby, a systemowi ICS, lub komponentowi CI/SSI korzystającego z usługi, zależy by tych zasobów nie wyczerpać. Po stronie serwera limity zajętości zasobów określone są dla każdego kanału komunikacyjnego z osobna, dlatego też istotnym jest grupowanie usług na kanałach. Okazuje

się jednak, że gdyby kontrola przepływu była realizowana arbitralnie, mogły by wystąpić zakleszczenia. Przykładowo, niektóre wywołania RPC muszą się zakończyć aby zwolnić zajęte zasoby. Rozważmy następujący scenariusz:

- węzeł A wysyła wywołanie RPC do węzła B
- kanał komunikacyjny zostaje zablokowany
- B próbuje wysłać RPC do A, by zwolnić zasoby
- zwolnienie zasobów odblokowałoby kanał, lecz nie może się zakończyć, gdyż kanał jest zablokowany

W celu uniknięcia tego typu problemów wprowadzono kanały priorytetowe, które nigdy nie są blokowane. Serwer ICS automatycznie ustawia priorytet wychodzącej wiadomości lub wywołanie RPC tak by było na wyższym poziomi niż RPC które wywołało wychodzące wywołanie/wiadomość (w przypadku przychodzącej wiadomości, nie dzieje się to automatycznie). Kanały na których przesyłane są odpowiedzi nigdy nie są blokowane.

# 5 ICS wysokiego poziomu

Na wysokopoziomowych usługach ICS opiera się cała komunikacja międzywęzłowa. Ogólnie można podzielić je na:

- ogólne procedury inicjalizacyjne, informujące o podniesieniu lub opuszczeniu węzła etc.
- procedury klienckie, wywoływane przez szkielet klienta wygenerowany przez *icsgen*
- kod kontrolujący serwer, obsługujący tworzenie i niszczenie uchwytów oraz demonów serwera, jak również wywołujący szkielet serwera wygenerowany przez *icsgen*
- procedury serwera, wywoływane przez szkielet serwera wygenerowany przez *icsgen*

# 6 ICS niskiego poziomu

Niski poziom ICS odpowiada za transport danych. Procedury tej części systemu wywoływane są wyłącznie przez ICS wysokiego poziomu. Nie są wywoływane przez szkielet generowany przez *icsgen* ani przez żadne inne komponenty CI/SSI.

Do komunikacji pomiędzy wysokim a niskim poziomem ICS wykorzystuje się koncepcję uchwytów. Występują one w dwóch odmianach: *Uchwyty klienta* - używane po stronie klienta oraz *uchwyty serwera* wykorzystywane przez serwer. Każdy uchwyt podzielony jest na dwie częsci: wysoko- i niskopoziomową. Niskopoziomowa część jest w całości przezroczysta dla wysokiego poziomu ICS. Z koleji część wysokopoziomowa zawiera pewne informacje wykorzystywane do komunikacji pomiędzy warstwami ICS. Jej reszta jest z koleji przezroczysta dla niskiego poziomu. Niski poziom ICS możee swobodnie manipulować swoją częścią uchwytu. Typowymi

przypdakami użycia są nadzór procesu kodowania/dekodowania argumentów wejściowych i wyjściowych, zapis całości bufora *il*, czy choćby obsługa blokad niezbędnych do obsługi uchwytów.

# 7 Podsumowanie

Pomimo możliwości definiowania własnych ułsug, podstawową funkcją systemu ICS w rozwiązaniu OpenSSI jest realizowanie komunikacji pomiędzy jądrami w celu zapewnienia SSI. Działanie tych mechanizmów jest dla użytkownika i jego procesów zupełnie przezroczyste. Wszelkie mechanizmy komunikacji, czy synchronizacji międzyprocesowej realizuje się (z punktu widzenia użytkownika) dokładnie jak w klasycznym systemie Linux. W tym własnie twki siła rozwiązań typu SSI. Dodatkowo, w dobrych rozwiązaniach, nie jest to kwestia jedynie wygody, ale i wydajności. To system decyduje jak zrealizować rozproszenie aplikacji i robi to najlepiej jak potrafi.

# 8 Uczta migrujących filozofów

Poniższy przykład powstał w celu ilustrowania przezroczystości działania ICS z punktu widzenia procesów uzytkownika. Do ilustracji migracji korzystamy z biblioteki *cluster.h*. Kompilacja z opcją *-lcluster*.

## 8.1 Kod źródłowy