INNOWACYJNA GOSPODARKA
NARODOWA STRATEGIA SPÓJNOŚCI

Inwestujemy w Waszą przyszłość

INSTEPRO
Zintegrowane Sterowanie Produkcją

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ ROZWOJU REGIONALNEGO

Raport wewnętrzny projektu InStePro

Nr 4.1: Projekt platformy koordynacji i nadzoru modułów

Data

30.09.2010

Przygotował zespół :

T. Pełech-Pilichowski
J.T. Duda

Platforma koordynacji i nadzoru modułów **jest podstawowym komponentem systemu InStePro**.

Do głównych jej zadań zalicza się pełnienie roli:

1. **Zarządzającego poszczególnymi modułami systemu**;
2. Przekaźnika informacji pomiędzy modułami a systemem ERP oraz komponentami nadzoru nad produkcją. Zadania realizowane na tym etapie są istotne ze względu na liczne możliwości uzyskiwania sygnałów diagnostycznych ze środowiska produkcyjnego (PLC, SCADA, MES, detektory, sygnalizatory, mierniki) oraz przekazywania ich do systemu ERP.

ZARZĄDZANIE MODUŁAMI SYSTEMU

Spis treści

1. Zadania i zlecenia

Zadania realizowane w systemie InStePro są podstawowymi składowymi scentralizowanego systemu obliczeniowo-decyzyjnego. W zależności od wykonywanych funkcji, wyróżnia się:

- Procesy bezpośrednio związane z systemem operacyjnym. Umożliwiają koordynacje zadań, wymianę informacji pomiędzy zadaniami oraz zarządzają działaniem całego systemu. Do tego typu należy:
    - moduł główny zarządzający wszystkimi operacjami systemowymi;
    - biblioteki dedykowane, które mają na celu izolację użytkownika od właściwości systemu operacyjnego i wewnętrznych protokołów komunikacji.
- Procesy specyficzne dla automatyki, tworzące podsystemy, które realizują poszczególne funkcje merytoryczne (przetwarzanie danych, sterowanie, monitoring, identyfikacja, analiza stanów procesu, obsługa sytuacji alarmowych, archiwizacja danych i obsługa poleceń operatorskich). Funkcje merytoryczne (funkcjonalności) określane są mianem **zleceń systemowych**. Jedno zadanie możne zawierać jedno lub wiele zleceń systemowych.

Zleceniu odpowiada procedura zlecenia, która realizuje lub nadzoruje poszczególne operacje (funkcje systemu). Jest ona na ogół realizowana w stosunkowo krótkim czasie, bez wewnętrznej kontroli reżimu czasowego (np. jednorazowe obliczenie wartości wejść sterujących przez algorytm regulacji, jednorazowa aktualizacja zawartości raportu, rozwiązanie zadania optymalizacji lub identyfikacji, jednorazowa analiza stanu alarmów itp.) przez autonomiczną procedurę („procedura zlecenia”, której przypisuje się unikalny numer).

Procedury te pogrupowane są w zadania. Możliwe jest dodanie nowego zlecenia (a tym samym rozszerzenie funkcji systemu) przez dołączenie do kodu zadania procedury zlecenia (ewentualnie z procedurami pomocniczymi) oraz kodu wywołującego tę procedurę.

Każde zlecenie może występować w roli:

- **zlecenia doraźnego** (wykonywanego jednorazowo; natychmiastowa egzekucja)
- lub **zlecenia okresowego** tj. stanowiącego obsługę zdarzenia czasowego, buforowanego.

Zlecenie może być zgłoszone przez dowolną procedurę biblioteki merytorycznej lub przez operatora.

Zadania często aktywowane rezydują w pamięci operacyjnej komputera, natomiast te, które wywoływane są rzadko, na okres realizacji wczytywane są do pamięci z dysku, a po wykonaniu są z niej usuwane. Zlecenia wykorzystują biblioteki procedur przeznaczone do komunikacji z innymi zadaniami oraz ułatwiające programowanie złożonych funkcji merytorycznych.

Zadania wykonujące poszczególne zlecenia powiązane są ze sobą tylko poprzez wspólną bazę danych (modularność systemu), jakkolwiek zlecenia zawarte w tym samym zadaniu mogą także używać wspólnych struktur danych lokalnych. Procedury realizujące zlecenia współdziałają ze sobą w analogiczny sposób jak wątki. Jednakże specyfika danych wykorzystywanych w procedurach merytorycznych umożliwia znaczne usprawnienie procesów obliczeniowych przez współbieżną realizację różnych procedur w ramach tego samego zadania, czyli w trybie wielowątkowym.

Wynikiem przetwarzania może być modyfikacja danych dynamicznych zlecenia, modyfikacja danych dynamicznych innego zlecenia i modyfikacja danych w bazach globalnych. Parametry są to dane wykorzystywane w sposób bierny przez wszystkie procedury danego zadania. Zajmują one statyczny obszar pamięci, wypełniony kopią zbioru parametrów przechowywanego w pamięci masowej.

2. Zadania główne

Do głównych zadań systemu InStePro należą:

- **Moduł główny** – zadanie o najwyższym priorytecie, odgrywa zasadniczą rolę w systemie jako administrator zleceń oraz administrator globalnej bazy danych. Przechowuje oraz uaktualnia w swej pamięci globalną bazę danych oraz agendę systemową, kontroluje wymianę danych pomiędzy modułami systemowymi oraz zajmuję się aktywacją zleceń. Moduł ten kontroluje zegar czasu rzeczywistego i rozpoznaje zdefiniowane przez użytkownika zdarzenia okresowe. Jest serwerem dla wszystkich zadań całej aplikacji.
- **Cykliczne przetwarzanie zmiennych procesowych** – przeliczanie sygnałów procesowych na zmienne procesowe, weryfikacja poprawności danych oraz sprawdzaniem czy zmienne procesowe spełniają ograniczenia. W systemie przyjmuje się 7 stref kontroli zmiennych: normalna, dwie awaryjne, dwie alarmowe i dwie wysokiego alarmu. Strefy te, za wyjątkiem ostatniej, definiowane są względem zadanej wartości nominalnej. Zadanie to zawiera dwa zlecenia:
    - przetwarzania podstawowego, które aktywowane jest synchronicznie, co 1 sekundę (obliczanie wartości chwilowych zmiennych, aktualizacja wartości zadanych regulatorów);
    - przetwarzania wtórnego, które aktywowane jest co ustaloną liczbę sekund, przeznaczone do wyliczania średnich, trendów, sygnalizacji alarmów itp.

  Aktywacja jest zsynchronizowana z procesem transmisji danych procesowych z warstwy sterowania bezpośredniego. Możliwa jest indywidualizacja okresu aktualizacji zmiennych. Jeżeli wszystkie dane nie są skompletowane, to zlecenie nie jest uruchamiane, natomiast dane aktualne interpolowane są w oparciu o dane historyczne. Wartości średnie wykorzystywane są przez zaawansowane algorytmy sterujące.
- **Interpretacja poleceń operatorskich** – zarządzanie komendami operatorskimi, zadanie ciągle aktywne. W zależności od komendy głównej, generuje ono zadania pomocnicze interpretujące dalsze komendy. Wyróżnia się tutaj zadania:
    - realizujące zgłaszanie raportów zmiennych procesowych i obsługę komend operatorskich na tych raportach;
    - zgłaszające raporty układów regulacji i umożliwiające interakcję operatora z raportami;
    - przeznaczone do realizacji zaawansowanych komend operatorskich, dotyczących stanu systemu, stanu zleceń, interakcyjnej modyfikacji parametrów systemu.
- Zadanie odpowiedzialne za realizację wszystkich **operacji dyskowych** systemu w czasie rzeczywistym, zawierające zlecenie aktualizacji bazy archiwalnej, inicjowane co określoną liczbę sekund (liczba ta jest maksymalnym czasem utraty zarejestrowanej informacji). Zadanie zawiera 3 zlecenia realizujące odpowiednio: zapis i odczyt danych zleceń oraz zapis przebiegów czasowych do plików tekstowych (np. do dalszego przetwarzania z wykorzystaniem pakietu MATLAB).
- **Niskopoziomowe zadania dedykowane**, inicjujące i nadzorujące przebieg transmisji między systemem InStePro i systemem sterowania bezpośredniego;
- **Wyświetlanie** – prezentacja wartości zmiennych procesowych;
- **Wizualizacja** stanu pętli regulacyjnych (funkcjonalność opcjonalna) oraz do zmiany wartości zadanych i parametrów regulatorów.

3. Realizacja zleceń

Realizacja zleceń jest nadzorowana przez **moduł główny**. Są one rejestrowane i kolejkowane w agendzie systemowej (przechowywanej w pamięci zadania) oraz aktywowane zależnie od wyspecyfikowanych uzależnień czasowych. Zlecenie może być zgłoszone do wpisania do agendy systemowej oraz do usunięcia z niej przez inne zlecenie lub przez komendę operatora.

Zgłoszenie nowego zlecenia sprowadza się do wywołania funkcji, dla której specyfikuje się odpowiednie dane:

- identyfikator zlecenia (moduł zlecenia, wersja, węzeł, na którym ma być uruchamiane),
- tryb aktywacji (zlecenie jednorazowe, okresowe),
- czas aktywacji oraz maksymalne opóźnienie aktywacji,
- specyfikacje warunków dostępu do globalnej bazy danych,
- rozmiar struktury danych,
- dane dynamiczne, które będą przesłane do procedury zlecenia w czasie aktywacji,
- W przypadku zleceń cyklicznych – krotność oraz czas następnej aktywacji.

Globalna baza danych rezydująca w pamięci zadania zawiera:

- agendę systemową wraz z wartościami argumentów dla poszczególnych zleceń systemowych,
- pełną specyfikację poszczególnych procesów,
- często używane parametry systemu pomiarowego,
- wartości nominalne wszystkich zmiennych procesowych oraz ograniczenia technologiczne,
- aktualne oraz średnie wartości zmiennych procesowych, a także ich relacje w stosunku do ograniczeń,
- wartości zadane regulatorów,
- jakościową charakterystykę stanów instalacji oraz listę zleceń wpisanych do agendy systemowej ,
- dane historyczne.

Dane przechowywane są na dysku (dane archiwalne) oraz w pamięci operacyjnej. Dane przechowywane w pamięci operacyjnej są okresowo (co *k* sekund) zapisywane na dysk, co zwiększa niezawodność systemu. Globalna baza danych rezydująca na dysku jest kopią danych, uaktualnianą co określoną liczbę sekund, przechowywaną przez zadany okres czasu w postaci zbiorów godzinowych. Wszystkie dane z bazy globalnej, wymagane do realizacji poszczególnych zleceń, są pobierane i modyfikowane za pośrednictwem zadania systemu, które spełnia zatem funkcję administratora zleceń oraz globalnej bazy danych. Dedykowane biblioteki systemowe umożliwiają dokonywanie wszystkich wymaganych operacji na bazie globalnej, bez względu na miejsce przechowywania danych, co pozwala na łatwą synchronizację dostępu z wyeliminowaniem możliwości powstania deadlock'u, ułatwia modyfikację systemu oraz ogranicza propagację błędów przetwarzania.

Zaprojektowano modyfikację struktur komunikacyjnych w celu stworzenia możliwości bezpośredniej komunikacji w układzie klient-serwer pomiędzy poszczególnymi modułami systemu.

4. Wymiana danych, koordynacja i nadzór modułów

4.1. Transmisja danych i komunikaty

Wymiana danych między dowolnym zadaniem i administratorem bazy danych odbywa się tylko z wykorzystaniem systemowego mechanizmu *send_data-receive_data*. Zadanie użytkowe występuje w roli nadawcy (pobranie danych) lub odbiorcy (przesłanie danych).

Transmisja danych przebiega wg następującego schematu :

a) zadnie użytkowe wysyła przy pomocy procedury systemowej *send*() komunikat do administratora zawierający informacje o swoich zadaniach (parametry sterujące komunikatu) oraz (jeśli są) dane przesyłane, po czym zawiesza się w oczekiwaniu na odpowiedź administratora;

b) administrator jest zawieszany na wywołaniu procedury systemowej *receive*() w oczekiwaniu na komunikaty zadań użytkowych lub sygnał budzika systemowego;

c) po otrzymaniu komunikatu interpretuje jego zawartość i przesyła przy pomocy systemowej funkcji *reply*() odpowiedź, która między innymi może zawierać żądane dane;

d) po przesłaniu odpowiedzi administrator wykonuje standardową pętlę, która kończy się na wywołaniu funkcji *receive*().

Zadanie wysyłające komunikat, po otrzymaniu odpowiedzi sprawdza jej poprawność, sygnalizuje ewentualne nieprawidłowości i kontynuuje przetwarzanie.

Wszystkie szczegóły techniczne związane z formowaniem komunikatów, ich wysłaniem i odbiorem zamknięto w wymienionych uprzednio procedurach własnej biblioteki procedur pomocniczych (plik *sysp.c*). W efekcie, w procedurach merytorycznych systemu operacje związane z transmisją danych ograniczają się do specyfikacji wymaganych danych i wywołania odpowiednich procedur.

4.2. Komunikacja dla potrzeb przetwarzania danych

Projektowany system cechuje się łatwością modyfikacji funkcji merytorycznych. Dzięki mechanizmom archiwizacji danych na dysku oraz interpolacji przebiegu procesu w oparciu o te dane, zapewnione są bardzo korzystne warunki restartu całego systemu sterowania. Możliwe jest przetrzymywanie bazy danych na dysku komputera, na którym nie są uruchomione zadania posiadające silne wymagania dotyczące reżimu czasowego.

Oddziaływanie operacji wymiany danych pomiędzy buforami plików dyskowych (rezydującymi w pamięci komputera), a dyskiem, dotyczy komputera, który realizuje te operacje. Dzięki temu możliwe jest zredukowanie obciążenia systemu sterowania, jakie wprowadza *File Manager* systemu QNX.

System będzie mógł współpracować z warstwą sterowania bezpośredniego o dowolnej architekturze (sterowniki cyfrowe, karty pomiarowe, komputer dedykowany).

Hierarchiczność zadań operatorskich i oddzielenie obsługi klawiatury od zadań realizujących odpowiednie raporty, umożliwi filtrację zleceń operatorskich pozwalającą usunąć oddziaływanie błędnych zleceń na poziomy wyższe (w szczególności na zadanie główne InStePro).

W systemie spełnione są wymagania dotyczące zabezpieczenia przed powstawaniem blokad dotyczące dostępu do globalnej bazy danych oraz komunikacji międzyprocesowej. System wykorzystuje tylko dwa

mechanizmy komunikacji (szczególnie mechanizm *Send-Receive-Reply*, bardzo efektywnie realizowany w systemie QNX[1]), co ma swoje uzasadnienie i daje szereg zalet wymienionych powyżej.

4.3. Realizacja zleceń w trybie KLIENT-SERWER

4.3.1. Komunikacja między zadaniami

Projektując system InStePro wyspecyfikowano wymagania związane ze sprawną komunikacją między zadaniami, tj.:

a) współbieżną i równoległą realizację usług;
b) odciążenie AZS od częstej transmisji danych;
c) możliwość synchronizacji usług;
d) możliwość realizacji usług niesynchronizowanych;
e) uproszczenie mechanizmów koordynacji przetwarzania dla najczęściej wykonywanych usług;
f) minimalizacja interakcji między oprogramowaniem merytorycznym, a trybem realizacji usług.

Jak wspomniano w Rozdziale 3, komunikacja pomiędzy zadaniami merytorycznymi systemu InStePro jest możliwa tylko za pośrednictwem modułu głównego. Do systemu wprowadzono **mechanizm bezpośredniej komunikacji między zadaniami** w celu uniknięcia przeciążenia modułu głównego i niedopuszczalnych w systemie czasu rzeczywistego opóźnień. Z drugiej strony wprowadzenie możliwości bezpośredniej komunikacji międzyzadaniowej może być źródłem blokad i problemów z synchronizacją. Należy również pamiętać, że system ma działać w środowisku rozproszonym, trzeba więc uwzględnić potencjalne problemy związane z awarią sieci i poszczególnych jej węzłów.

Problemy komunikacji międzyzadaniowej są z reguły trudne do rozwiązania i zawsze trzeba się liczyć z przyjęciem kompromisu między niezawodnością systemu, a jego elastycznością i prędkością działania. Należy dążyć do maksymalnego uproszczenia schematów komunikacyjnych – zarówno z uwagi na ograniczenia czasowe systemu, jak i konieczność wyeliminowania możliwości pojawienia się blokad.

Analiza wymagań protokołu bezpośredniej komunikacji międzyzadaniowej:

a) **Złożoność przetwarzania** – konieczna funkcjonalność umożliwiająca wykonanie złożonych i czasochłonnych funkcji.
b) **Rozproszenie przetwarzania** – rozróżnienie zadań zasadniczych na główne (zlecające przez moduł główny zadaniom pomocniczym czasochłonne czynności, dając możliwość przyjęcia innego zlecenia od administratora) i pomocnicze. W tym celu wyróżniono 4 stany, w których mogą znajdować się zadania:
    - SLEEP (w stanie uśpienia tzn. zawiesił wykonywanie na określony czas),
    - READY (zadanie jest gotowe do wykonania, ale nie jest aktualnie wykonywane),
    - ACTIVE,
    - WAIT (wątek[2] nie wykonuje w danej chwili żadnych operacji, nie jest również zakończony, ale oczekuje na komunikat z innego zadania).
c) **Komunikacja skrośna** – stworzenie możliwości komunikacji zadań bez pośrednictwa administratora (cel: uniknięcie przeciążenie modułu głównego).

[1] QNX cechuje wielozadaniowość, szybka komunikacja międzyprocesowa, mechanizmy RT
[2] Kontekst wątku – bufory pamięci operacyjnej przetrzymywane na stałe lub przez okres realizacji zlecenia

d) **Prostota protokołu komunikacyjnego** – zgłaszania zleceń przez zadania główne do modułu głównego, który zapisuje je w agendzie i szuka wykonawcy. Wykonawca natomiast, żądania danych oraz informacje o zakończeniu etapu pracy będzie przekazywał już bezpośrednio do zadania zlecającego usługę.

- Zadanie wykonujące usługę – **SERWER** (wątek realizujący usługę – WRU)
- Zadanie (wątek) zlecające usługę — **KLIENT**.

Sposób przekazywania danych do serwera i rezultatów do klienta musi być dokładnie zdefiniowany i pozbawiony możliwości wystąpienia zakleszczeń. Z punktu widzenia programisty zadań merytorycznych, zlecanie usług powinno być możliwie proste, a problemy związane z komunikacją międzyzadaniową niewidoczne. Konieczne jest więc stworzenie biblioteki funkcji użytkownika, ukrywających te aspekty zlecania usług i komunikacji międzyzadaniowej, które są nieistotne z punktu widzenia merytorycznych funkcji systemu.

W realizacji procedur w układzie klient-serwer systemu InStePro kluczową rolę pełnią:

- Zlecenia autonomiczne procedury o statusie wątków, realizowane bez wewnętrznej kontroli reżimu czasowego (operacje merytoryczne sterowania):
  - **Zlecenia główne** (podstawowe funkcje systemu), uruchamiane cyklicznie (automatycznie) i kontrolowane przez AZS (nazywane w niniejszym raporcie jako ***Zlecenie***). Realizowane przez wyodrębnione moduły odpowiednich zadań spełniające rolę bardzo zbliżoną do wątków, bez wewnętrznej kontroli reżimu czasu rzeczywistego (operacje merytoryczne sterowania).
  - Zlecenia doraźne – **usługi**: złożone operacje obliczeniowe wykorzystujące duże pakiety danych.
- Proces główny – Administrator Zleceń Systemowych (AZS) – zarządzanie zleceniami akwizycja i koordynacja zleceń w czasie rzeczywistym.

4.3.2. Administrator Zleceń Systemowych (AZS)

Jak sygnalizowano w Raporcie wewnętrznym 2.4 oraz Rozdziale 2 niniejszego opracowania, od strony merytorycznej realizowane w systemie przetwarzanie danych dzieli się na przetwarzanie synchroniczne, inicjowane co jedną sekundę, oraz przetwarzanie asynchroniczne – realizujące funkcje zlecane bądź przez procedury przetwarzania synchronicznego bądź przez operatora.

Wszystkie moduły przetwarzania synchronicznego zostaną zgrupowane w jednym procesie. Będą one wykorzystywały wspólne zasoby danych procesowych pobieranych za pośrednictwem modułów komunikacyjnych z systemu sterowania (SCADA lub inne systemy automatyki procesowej). Dane te będą buforowane w pamięci procesu w postaci spójnych szeregów czasowych o zadanej długości i na bieżąco przesuwanych w pamięci. Rozwiązanie takie pozwoli uniknąć opóźnień związanych z wymianą komunikatów z innymi procesami.

Moduły przetwarzania asynchronicznego zostaną zgrupowane w kilku procesach grupujących procedury przetwarzające dane podobnego typu, koordynowanych przez proces zarządzający zwany **Administratorem Zleceń Systemowych** (AZS), którego rolą jest:

- obsługa startu i restartu systemu (w tym interpolacja brakujących danych sygnałów diagnostycznych oraz inne operacje związane z obsługą bazy danych czasu rzeczywistego),
- ładowanie do pamięci operacyjnej zadań, odczyt parametrów, inicjalizacja zmiennych,

- buforowanie danych wszystkich zleceń (w tym obsługa agendy systemowej), kontrola i harmonogramowanie zleceń,
- uruchamianie zleceń zgodnie z zadanymi uwarunkowaniami czasowymi (kontrola zegara czasu rzeczywistego),
- pośrednictwo w kontaktach wszystkich pozostałych procesów z GBD, zarządzanie bazami danych sygnałów diagnostycznych: zmiennych pomiarowych, procesowych itp.,
- kontakt z systemami zewnętrznymi poprzez co najmniej dwa procesy komunikacyjne, realizujące odpowiednio (1) pobieranie danych z otoczenia procesu (systemy SCADA lub inne systemy automatyki przemysłowej, systemy ERP/MRP) oraz (2) odsyłanie komunikatów systemów zewnętrznych.

Globalna baza danych (GBD) będzie przechowywała dane wykorzystywane lub modyfikowane przez różne podsystemy. W jej skład będą wchodziły następujące zbiory danych:

a) agenda systemowa zawierająca zbuforowane dane zleceń;
b) dane konfigurujące system sterowania procesem, tj. zmienne procesowe, sygnały procesowe, pętle regulacyjne, atrybuty zdefiniowanych procesów technologicznych, tworzących zhierarchizowaną strukturę, umożliwiającą interpretację zmiennych procesowych w kategoriach wejść/wyjść stanu, sprzężeń zwrotnych itp.;
c) baza czasu rzeczywistego wartości zmiennych procesowych mierzonych synchronicznie wraz z zakodowaną informacją o wiarygodności tych zmiennych;
d) baza czasu rzeczywistego stanu pętli regulacyjnych – wartości zadane i zakodowany stan pracy regulatora;
e) baza czasu rzeczywistego zdarzeń asynchronicznych i stanu procesu.

Działanie AZS prowadzone jest w oparciu o harmonogram zadań (**agendę systemową**), zawierającą zbuforowane dane zleceń (m.in. parametry uruchomieniowe zapisane w postaci odpowiednich struktur, cykliczność uruchamiania zadań, możliwość wykonywania zleceń w innych zadaniach).

AZS będzie kontaktował się ze wszystkimi procesami poprzez standaryzowane komunikaty (wewnętrzne protokoły komunikacyjne). Zasadniczo wszystkie procesy będą pobierały i modyfikowały dane w bazie globalnej poprzez AZS. W dalszej perspektywie możliwe będzie również tworzenie kanałów komunikacyjnych pomiędzy poszczególnymi procesami w dynamicznym układzie klient-serwer.

4.3.3. Zasady ogólne funkcjonowania układu klient-serwer.

Układu klient-serwer funkcjonuje w oparciu o następujące zasady, umożliwiające autonomię modułów systemu oraz skrośną komunikację pomiędzy:

- Przydzielenie serwera, rejestracja stanu usługi i zainicjowania jej realizacji (w wątku serwera) – realizowane przez AZS.
- Dla każdej usługi procesy klienta i serwera tworzą dynamiczny kanał komunikacyjny (dane konfigurujące i bufory) dla bezpośredniej transmisje danych i wyników pomiędzy tymi procesami.
- Wykorzystane mechanizmy transmisji danych między procesami i węzłami sieci (dostępne w systemie QNX):
    - nieblokujące stałe komunikaty do wybranego procesu, zwane depozytami (PROXY[3]),

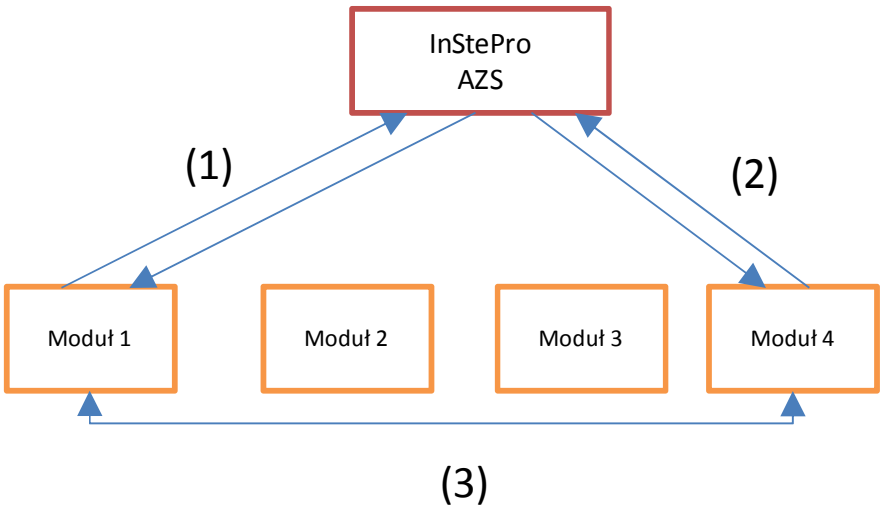
[3] PROXY – depozyt jest typem nieblokującej, stałej wiadomości skierowanej do konkretnego procesu. Depozyty działają w sieci, komunikacja ograniczona jest jednak do sieci lokalnych. Po utworzeniu depozytu nie jest możliwa zmiana treści wiadomości. Depozyty realizują nieblokujące przesyłanie wiadomości. A więc proces, który wywołał funkcje Trigger() nie czeka na odebranie wiadomości przez proces, do którego była ona

- synchronizowane komunikaty o dowolnej zawartości (SEND()-RECEIVE()-REPLY());
- Wszystkie komunikaty wysyłane przez AZS są typu nieblokującego (PROXY i REPLY), co eliminuje możliwość jego zablokowania.

Planowane jest stworzenie biblioteki procedur umożliwiających łatwe zgłaszanie usług, z ukryciem wszystkich operacji związanych z ich uruchamianiem, kontrolą realizacji i transmisjami danych.

Rysunek 1 przedstawia ogólny schemat komunikacji pomiędzy modułami 1 oraz 4 w proponowanej architekturze klient-serwer. Zakładając, że Moduł 1 potrzebuje danych generowanych w trakcie działania Modułu 4 (oznaczonych jako *dataset)*, komunikacja będzie przebiegała w następujących etapach:

(1) Moduł 1 staje się klientem, wysyła do AZS żądanie udostępnienia zestawu *dataset*.
(2) AZS komunikuje się z Modułem 4 (żądanie wygenerowania *dataset*).
(3) Moduł 4 tworzy kanał komunikacyjny z Modułem 1 (przesyłanie zestawu *dataset*)



Rysunek 1. Schemat komunikacji pomiędzy modułami za pośrednictwem AZS

4.3.4. Koordynacja modułów/procesów

Przyjęto założenie, że jak najwięcej szczegółów powinno być ukrytych przed użytkownikiem - programistą zadań merytorycznych. Założono możliwość rozproszonej realizacji złożonych zleceń głównych poprzez zgłaszanie usług obliczeniowych do innych procesów systemu, w tym również na innych węzłach sieci komputerowej.

Usługi mają status zleceń systemowych, z tym że wymiana danych pomiędzy serwerem a klientem może się odbywać bez pośrednictwa AZS. Ma to na celu odciążenie AZS od pośrednictwa w przesyłaniu dużych komunikatów i stanowi istotne odstępstwo od dotychczasowych zasad funkcjonowania omawianego systemu. Zgłoszenie usługi może być dokonane przez dowolny podsystem, za pośrednictwem AZS.

Zgłoszenie następuje z poziomu wątku realizującego zlecenie (WRZ; wątek zawarty w procesie realizującym zadanie merytoryczne – procesie realizującym zlecenie PRZ) realizujący zadanie

skierowana. Natomiast odbieranie wiadomości możliwe jest zarówno w trybie blokującym, jak i nieblokującym. W pierwszym przypadku proces zostaje zawieszony do momentu nadejścia pobudzenia (funkcja jest także odblokowywana w wyniku dostarczenia sygnału lub usunięcia z systemu depozytu, do którego adresowana była funkcja odbioru wiadomości)

merytoryczne nazywamy wątkiem realizującym zlecenie WRZ) i jest równoznaczne z nadaniem WRZ statusu klienta. Dany WRZ może zgłosić równocześnie wiele usług. Po zgłoszeniu usług WRZ przekazuje do AZS informacje, że PRZ jest w stanie WAIT (czeka na zakończenie zleconych usług).

AZS jest serwerem wszystkich zleceń niezwiązanych z obsługą operatorską, przy czym wszystkie wysyłane przez niego komunikaty są typu nieblokującego (PROXY i REPLY). Eliminuje to możliwość zablokowania AZS.

4.3.5. Cykl inicjalizacji zleceń

Wszystkie zlecenia są inicjowane przez centralnego administratora w następującym cyklu:

1. AZS umieszcza we własnej kolejce zleceń gotowych, zlecenie które wymaga realizacji (ze względu na jego uwarunkowania czasowe) i nadaje mu status READY. Zlecenie to może być zleceniem cyklicznym (głównym) lub doraźnym zleconym przez dowolny podsystem, przeznaczonym do zrealizowania w trybie natychmiastowym. Zlecenie może wymagać realizacji w trybie klient-serwer, w takim przypadku wątek który je zleca przesyła odpowiednią informację do AZS.

2. AZS poszukuje wykonawcy zlecenia w sieci komputerowej według algorytmu określonego przez klienta przy pomocy parametru *node*.

3. Po przydzieleniu procesora, AZS wysyła komunikat nieblokujący typu *task_proxy* do wybranego zadania. Komunikat ten zostanie odebrany z chwilą gdy pobudzany proces znajdzie się w stanie WAIT. W przypadku gdy proces nie istnieje, AZS otrzymuje odpowiedź negatywną.

4. Jeżeli wskazany proces nie istnieje, to jest ładowany jako potomek AZS. Po załadowaniu wykonuje procedurę *open_task*(), która dokonuje alokacji buforów ogólnego przeznaczenia procesu, odczytuje przydzielony do zadania identyfikator sieciowy *my_task_id*, oraz otwiera kanał komunikacyjny typu PROXY, przydzielając mu specyficzny dla zadania identyfikator o nazwie *task_proxy*, po czym przesyła do AZS komunikat informujący o aktywności zadania, z podaniem *task_proxy* i węzła sieci, na którym zadanie zostało zaalokowane. AZS rejestruje przysłany *task_proxy*, oraz przydzielony przez system identyfikator sieciowy (*task_id*).

5. Proces realizujący usługę zawiesza się w oczekiwaniu na komunikat od AZS (funkcja systemowa *receive*()). Po odebraniu go przesyła do AZS komunikat typu SEND, żądając przesłania specyfikacji zlecenia.

6. AZS, po uzyskaniu komunikatu (punkt 5), odsyła do zadania *task_id* komunikat typu REPLY, zawierający wymaganą specyfikację zlecenia i nadaje zleceniu w kolejce status ACTIVE. Jeżeli zlecenie ma być realizowane w trybie klient-serwer, to AZS przesyła do wątku realizującego usługę (WRU) informację, że konieczne jest nawiązanie bezpośredniej komunikacji z klientem. W takim przypadku WRU, po otrzymaniu specyfikacji usługi, nadaje sobie status serwera. Dalsza komunikacja przebiega zgodnie z jednym z trybów opisanych w punkcie 5.4.

7. Zlecenia główne są do końca nadzorowane przez AZS. Dla pozostałych zleceń AZS rejestruje ich stan na podstawie komunikatów wysyłanych przez PRZ. Sposób realizacji zlecenia w PRZ jest zależny od oprogramowania zlecenia. W szczególności wątek realizujący zlecenie (WRZ) może komunikować się z AZS lub innymi procesami systemu. Narzuca się jedynie konieczność informowania AZS o stanie realizacji zlecenia po chwili, w której programista chce uzależnić dalszą obsługę zlecenia od potrzeby i priorytetów realizacji innych wątków procesu. W tym celu WRZ oddaje sterowanie do PRZ z wartością jednoznacznie sygnalizującą stan realizacji zlecenia. PRZ wywołuje wtedy standardową funkcję *toInStePro*(), która przesyła odpowiedni komunikat typu SEND do AZS.

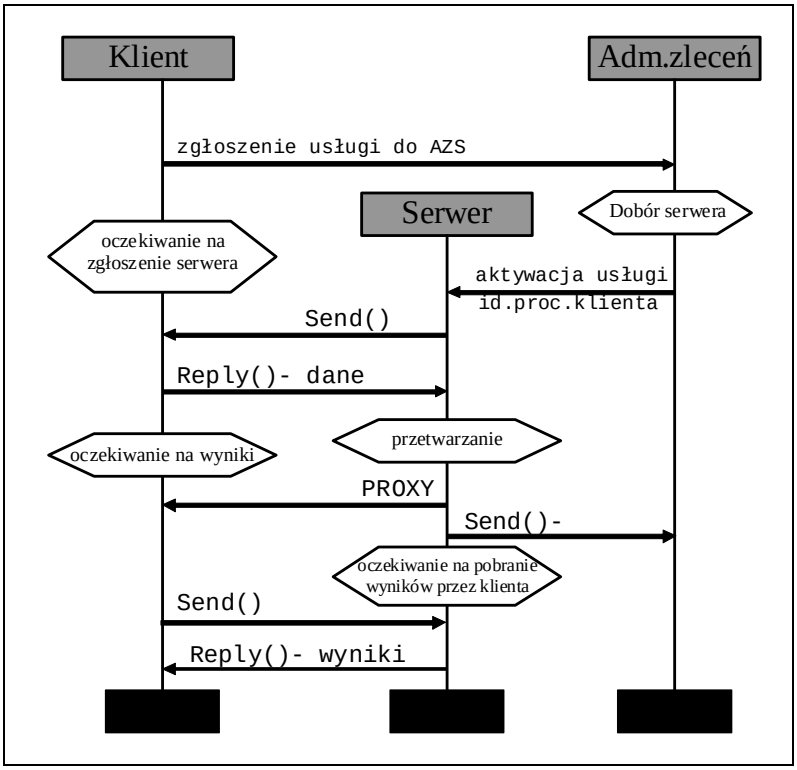
8. AZS po odebraniu komunikatu jw. zmienia status zlecenia w kolejce: jeśli zlecenie zostało zakończone usuwa go z kolejki, jeśli zostało zawieszone dla umożliwienia realizacji pilniejszych wątków – nadaje mu status READY, jeśli zostało przerwane w oczekiwaniu na dane z innych podsystemów, nadaje mu status WAIT.

4.3.6. Typy usług

Aktywizacja usług wymaga dołączenia do danych dynamicznych bufora zawierającego specyfikację parametrów kanału komunikacyjnego klienta.

Usługa zwykła - typowe doraźne operacje numeryczne (rozwiązanie układu równań, symulacja, wykonanie operacji macierzowych). Klient zleca usługę, przesyła dane niezbędne do jej wykonania i po zakończeniu obliczeń odbiera ich rezultaty. Usługa zwykła wymaga przejścia klienta w stan oczekiwania na jej zakończenie.

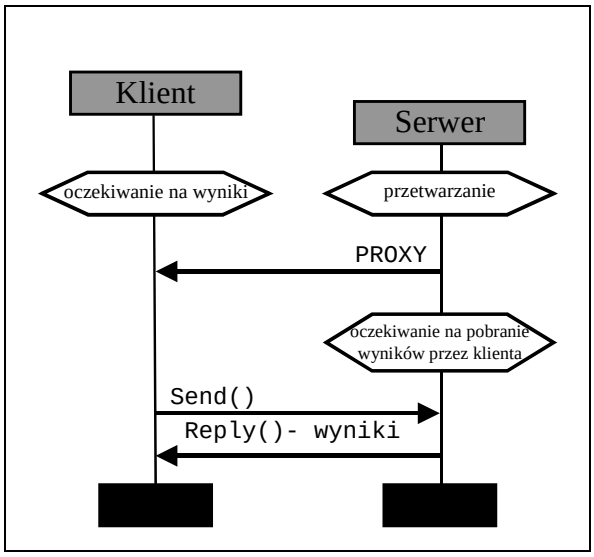
Przykładem takiej usługi jest zlecenie symulacji zgłaszane przez zadanie sterowania predykcyjnego do zadania zawierającego procedury symulacji, zlecenie obliczenia modelu matematycznego dla procedur optymalizacji, zlecenie wykonania operacji macierzowych dla potrzeb identyfikacji i regulacji.



Rysunek 2. Usługa zwykła

Usługa stała (permanentna, cykliczna) – klient zleca usługę, która jest ponawiana automatycznie w zadaniu serwera w trakcie każdorazowej realizacji określonego wątku. Ten typ usługi można zlecić tylko zleceniom stałym wywoływanym okresowo przez AZS. Dany serwer może wykonywać różne usługi permanentne w ramach tego samego wątku. Wynika to stąd, że ten typ usługi jest adresowany do zlecenia stałego i inicjowany poprzez odpowiednią zmianę parametrów tego zlecenia. Oczywiście tego typu zlecenie może być zlecone przez wielu klientów np. z wielu konsol operatorskich można zażądać od regulatora danych do rysunku.

Dobrym przykładem takiej usługi jest przygotowania i wysłania wyników regulacji do ich prezentacji graficznej, klientem jest tu zlecenie obsługi grafiki, serwerem wątek realizujący zadania regulacji.



Rysunek 3. Cykl usługi stałej

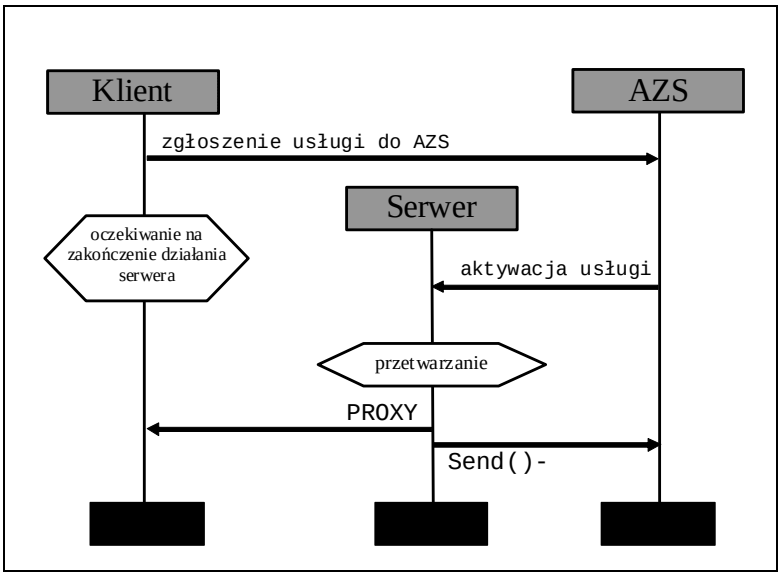
Po zainicjowaniu usługi przez klienta, jest ona realizowana jako usługa nadzwyczajna zlecenia okresowego (tryb zmiana parametrów zlecenia, bez jego wykonania). W każdym następnym wywołaniu wątku realizuje się tylko etapy 6 i 7 działania usługi zwykłej serwera, tzn. serwer po zakończeniu działań, wysyła komunikat typu PROXY, czeka na komunikat SEND od klienta i wysyła wyniki komunikatem REPLY. Usługa permanentna kończy się, gdy serwer dostaje negatywną odpowiedź na komunikat PROXY, co oznacza, że klient zrezygnował z usługi (usunął PROXY). A więc klient chcąc zakończyć usługę permanentną może po prostu usunąć PROXY lub zakończyć działanie. Klient może także specjalnym komunikatem zgłosić zakończenie usługi (*kill_service()*) lub jej modyfikacje (*modyf_service()*). W drugim przypadku poprzednia wersja usługi jest unieważniona, a na jej miejsce wchodzi nowa.

Usługi stałe zgłaszane są w taki sam sposób jak zwykłe, z tym że wymagają podania identyfikatora zlecenia systemowego, do którego mają być przypisane.

Usługa w tle – dotyczy to takich operacji, jak adaptacja modeli w zadaniu regulacji, które nie muszą być wykonywane w trybie pilnym, a więc nie muszą blokować realizacji wątku. W tym przypadku działanie serwera jest takie, jak przy usłudze zwykłej, natomiast odmienne jest działanie wątku będącego klientem. Klient nie czeka na zakończenie zleconej przez siebie usługi. Bufor przeznaczony na wyniki działania serwera nie może być zwalniany w momencie usuwania kanału komunikacyjnego, ani też po zakończeniu wątku klienta, ponieważ przechowywane w nim dane są przeznaczone do wykorzystania przez inne wątki procesu, bądź kolejne wywołania tego samego wątku klienta. Oczywiste jest, że adres tego bufora musi być znany wątkom, które będą chciały z niego korzystać. Po odebraniu wyników (dzieje się to automatycznie, bez ingerencji programisty wątku klienta) kanał komunikacyjny jest usuwany, a pozostawiony zostaje jedynie bufor z wynikami. Usługa w tle może być zgłoszona w dowolnym miejscu programu klienta bez wpływu na reżim dalszego przetwarzania.

Usługa synchronizacyjna – może posłużyć do synchronizacji sprzężonych ze sobą wątków obsługujących polecenia operatorskie (np. zadania raportowania, wyświetlania). Żadne dane nie są przekazywane do serwera, a klient nie oczekuje na wyniki. Istotny jest tylko to, kiedy zakończyła się usługa. W związku z tym, tryb komunikacji jest maksymalnie uproszczony – klient zleca usługę za pośrednictwem AZS, a następnie czeka na komunikat typu PROXY od serwera, który oznacza

zakończenie usługi. Usługa synchronizacyjna jest zgłaszana jednoetapowo, gdyż nie wymaga ona przesyłania danych ani odsyłania rezultatów.



Rysunek 4. Usługa synchronizacyjna

4.3.7. Operacje wykonywane przez klienta

1. Otwarcie kanału komunikacyjnego przez dodanie do listy kanałów struktury danych (*wait_list*) i stałej długości. Struktura ta przechowuje wszystkie informacje potrzebne do przeprowadzenia komunikacji.
2. Wysłanie do centralnego administratora zleceń (AZS) zgłoszenia usługi, zawierającego informacje o rozmiarze buforów przechowujących dane zlecenia oraz przeznaczonego na wyniki.
3. Przejście w stan oczekiwania na komunikat typu SEND od serwera, informujący o gotowości do przyjęcia danych. Komunikat taki ma stałą długość, zawiera adres struktury danych przechowującej dane zlecenia, identyfikator procesu serwera, oraz adres struktury danych odpowiedzialnej za zlecenie po stronie serwera. Klient otrzymuje w ten sposób informacje niezbędne do dalszej komunikacji.
4. Odebranie komunikatu i zachowanie identyfikatora serwera i adres jego struktury odpowiedzialnej za zlecenie.
5. Korzystając z otrzymanych informacji o serwerze, klient wysyła mu za pomocą komunikatu typu REPLY dane potrzebne do wykonani usługi.
6. Przejście w stan oczekiwania na komunikat typu PROXY, którego nadejście informuje o zakończeniu obliczeń przez serwer i gotowości do przesłania ich wyników.
7. Odebranie komunikatu typu PROXY od serwera.
8 Wysłanie komunikatu typu SEND, i oczekiwanie w odpowiedzi wyników obliczeń. W komunikacie zawarty jest adres i rozmiar bufora przeznaczonego na wyniki. Komunikat SEND jest blokujący, działanie PRZ jest więc całkowicie zawieszone – nie może w tym czasie wykonywać innych wątków.
9. Po uzyskaniu odpowiedzi (REPLY) dekrementacja licznika aktywnych zleceń synchronizowanych. Jeżeli licznik jest równy 0, klient podejmuje działanie wątku korzystając z wyników zleconych usług.

4.3.8. Operacje wykonywane przez serwer

1. Działanie serwera rozpoczyna komunikat od AZS, zawierający informacje o zleceniu, identyfikator procesu klienta, adres struktury odpowiedzialnej za zlecenie u klienta, adres PROXY i rozmiary buforów przeznaczonych na dane zlecenia i wyniki obliczeń. Dane te tworzą strukturę o nazwie *cl_data* i stałej długości, która jest dołączana przez klienta do danych dynamicznych zlecenia i odesłana przez AZS do serwera. Oprócz tego podane są informacje o maksymalnych czasach oczekiwania na kolejne etapy komunikacji (*timeout*).
2. Alokacja struktury *wait_list*, na liście kanałów komunikacyjnych.
3. Alokacja buforów przeznaczonych na dane niezbędne do wykonania zlecenia.
4. Sprawdzenie na własnej liście kanałów komunikacyjnych, czy klient zgłaszający usługę nie może być w stanie oczekiwania na komunikat danego serwera, co może mieć miejsce w przypadku próby zlecenia przez jednego klienta kolejnej usługi dla tego samego serwera. Jeżeli tak nie jest, to serwer wysyła do klienta komunikat typu SEND, oznaczający żądanie danych. Podany w nim jest identyfikator serwera i adres kanału komunikacyjnego w procesie serwera. Do czasu wysłania przez klienta komunikatu REPLY serwer jest zawieszony (wynika to z blokującego charakteru funkcji SEND). Taki tryb transmisji gwarantuje, że kanał zostanie otwarty prawidłowo w dowolnym układzie klient-serwer.
5. Rozpoczęcie obliczeń po uzyskaniu danych komunikatem REPLY.
6. Po wykonaniu usługi, wysłanie do klienta komunikat typu PROXY, informującego o zakończeniu działania. Następnie serwer:
    - pozostaje w stałym nasłuchu kolejki FIFO przychodzących komunikatów, nie wysyłając informacji do AZS o zakończeniu zlecenia. Dzieje się tak w przypadku, gdy klient ustawił flagę *wait_cl_send* (tzn. zażądał biernego czekania).
    - zgłasza do AZS zakończenie zlecenia i przechodzi w stan oczekiwania (może podjąć inne wątki zlecone przez AZS).
7. Komunikat SEND od serwera powoduje wysłanie REPLY z wynikami.
8. Zwolnienie bufora opisującego kanał komunikacyjny, bufora z wynikami obliczeń i zakończenie działania wątku.

5. Uwagi końcowe

Projekt środowiska klient-serwer jest podstawą do budowy platformy do realizacji czasochłonnych zleceń symulacji dynamiki procesów w trybie rozproszonym. Dzięki bardzo efektywnym mechanizmom komunikacji międzyprocesowej i alokacji pamięci QNX opóźnienia komunikacyjne i organizacja przetwarzania nie będą zauważalnie wpływać na czas przetwarzania. Zastosowana autonomia modułów i komunikacja skrośna zapewnie koordynację wykonywania zadań/zleceń oraz nie odciąża cały system, co z kolei umożliwia spełnienie przyjętego reżimu czasu rzeczywistego (domyślnie – 1 sekunda). • Omawiane środowisko można także wykorzystać do implementacji systemów wieloagentowych. Procedury realizujące usługi zwykłe i w tle mogą być widziane jako zbiór M-agentów konkurujących o podjęcie zleceń zgłaszanych do administratora zleceń.