ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1997
Seria: INFORMATYKA z. 32 Nr kol. 1356

Piotr W. UMIŃSKI
Politechnika Gdańska, Wydział Elektroniki, Informatyki i Telekomunikacji

NIEZAWODNA KOMUNIKACJA GRUPOWA W SIECIOWYM ŚRODOWISKU ROZPROSZONYM

Streszczenie. W artykule przedstawiono koncepcję usługi serwera o podwyższonej niezawodności, określonej jako serwer grupowy. Usługa takiego serwera jest realizowana na kilku oddzielnych komputerach połącznych siecią. Zapewnia ona poprawną pracę nawet w wypadku awarii niektórych węzłów składowych dzięki bieżącemu porównywaniu wyników oraz zastosowaniu mechanizmów odtwarzania bieżącego. Komunikacja z serwerem grupowym odbywa się za pomocą specjalnego, niezawodnego protokołu grupowego. W artykule omówiono kilka reprezentatywnych protokołów komunikacji grupowej, a następnie przedstawiono oryginalną koncepcję nowego protokołu grupowego GREP. Protokół ten, w odróżnieniu od istniejących protokołów ogólnego przeznaczenia, został zaprojektowany specjalnie do obsługi serwisu grupowego.

RELIABLE MULTICAST GROUP COMMUNICATION IN NETWORKED DISTRIBUTED SYSTEMS

Summary. The new idea of a highly reliable service is presented in this paper for use in large-scale client-server distributed systems. This service, called a grupe server, is performed on several different nodes (computers) at the same time. We will describe some existing reliable multicast protocols (system ISIS, system Newtop, protocol RMP and protocol Totem). Then we will present a new multicast protocol, GREP (Group Reliable Protocol) to be used to access a group server. This protocol allows clients to send a message to the group server and guarantees, that all members of the group will receive the same, correct message. It allows also to generate a common response from the group, based on voted results from each member of the group server.

1. Wstęp

Popularnym modelem przetwarzania dla dużych systemów rozproszonych jest model klient - serwer, realizowany na bazie komputerów połączonych siecią. W modelu tym procesy pełniące role serwerów wykonują operacje, które zlecają im klienci. Dzięki temu możliwe jest rozproszenie operacji wykonywanych przez system, co z kolei pozwala na zwiększenie wydajności systemu. Model przetwarzania typu klient-serwer wydaje się atrakcyjny również ze względów niezawodnościowych - pozwala na wyodrębnienie części systemu o podwyższonych wymaganiach niezawodnościowych i specjalne ich potraktowanie. Jednocześnie duże systemy rozproszone mogą być stosunkowo łatwo tworzone na bazie istniejącego sprzętu (komputery połączone sieciami).

W artykule przedstawiono koncepcję usługi serwera o podwyższonej niezawodności, określanej jako serwer grupowy. Serwer ten, który ma zastąpić tradycyjne procesy - serwery, jest realizowany na kilku oddzielnych, połączonych siecią komputerową węzłach systemu i zapewnia poprawną pracę nawet w wypadku awarii jednego lub kilku komputerów wchodzących w jego skład. Dzięki bieżącemu porównywaniu wyników oraz zastosowaniu mechanizmów odtwarzania bieżącego taki serwer posiada bardzo dużą niezawodność przy zachowanej dużej szybkości działania.

W niniejszej pracy zostanie przedstawiona szczegółowo koncepcja serwera grupowego. Do zapewnienia współpracy pomiędzy procesami - klientami a serwerem grupowym konieczne jest zastosowanie specjalnego protokołu komunikacji grupowej. Protokół taki zapewnia niezawodne dostarczanie wiadomości do wszystkich węzłów wchodzących w skład grupy. Istnieje wiele prac poświęconych komunikacji grupowej. W dalszej części pracy przedstawione zostaną wybrane, istniejące protokoły komunikacji grupowej. Następnie zaprezentowany zostanie nowy protokół komunikacji grupowej opracowany specjalnie na potrzeby tworzenia serwerów grupowych. Protokół umożliwia nie tylko poprawną pracę serwera grupowego, ale umożliwia również diagnozowanie uszkodzonych bądź błędnie działających węzłów wchodzących w jego skład. W podsumowaniu przedstawione zostaną proponowane przez autora kierunki dalszych prac.

2. Pojęcie serwera grupowego

2.1. Odtwarzanie poprawnego stanu w systemach klient - serwer

W środowisku rozproszonym typu klient-serwer możemy wyróżnić dwie klasy procesów: procesy będące klientami oraz procesy spełniające rolę serwerów. Z uwagi na odmienne działanie procesy - klienci i procesy - serwery mają inne wymagania niezawodnościowe. Procesy - klienci są tworzeni *ad hoc* do wykonania określonych zadań. Procesy - klienci korzystają z wyników działania procesów - serwerów, z którymi komunikują się za pomocą komunikatów. Zakładamy, że czas życia (działania) klienta nie jest zbyt długi, a opóźnienie w uzyskaniu wyników działania klienta nie jest katastrofalne dla całego systemu.

W wypadku awarii klienta usunięcie jej skutków musi być dokonane zarówno po stronie klienta, jak i po stronie serwera. Odtwarzanie dokonywane po stronie serwera (ang. *server recovery*) polega na usunięciu zmian w stanie serwera rozpoczętych, ale nie zakończonych i nie potwierdzonych (ang. *non-commited*) przez klienta, który w tym czasie uległ awarii. Odtwarzanie dokonywane po stronie klienta (ang. *client recovery*) polega natomiast na ponownym uruchomieniu procesu klienta.

Jak widać, odtwarzanie po stronie klienta jest stosunkowo proste - przybiera najprostszą postać odtwarzania wstecznego. Odtwarzanie po stronie serwera może być bardziej skomplikowane, ale jest w zasadzie takie samo jak w zwykłych systemach klient - serwer, opartych np. na mechanizmie transakcji.

Proces klienta działa zawsze jako pojedyncza kopia w systemie - nie jest tu dokonywana żadna replikacja. Proces - serwer działa natomiast przez cały czas życia systemu. Oferuje on klientom (i innym serwerom) określonego rodzaju usługi. Musi być dostępny przez cały czas, a czas jego odpowiedzi powinien być minimalny. Jego stan jest związany z całą historią działania tego systemu. Dlatego należy przyjąć inną niż dla klienta strategię odtwarzania poprawnego stanu serwera w wypadku awarii. Przede wszystkim czas odtwarzania poprawnego stanu serwisu po awarii powinien być minimalny. Jednocześnie powinien być odtworzony możliwie najnowszy stan tego serwera - nie jest akceptowalna zwykła inicjalizacja jego pracy od początku. Dlatego w tym wypadku odtwarzanie stanu serwera po awarii powinno się odbywać na bieżąco (ang. *forward recovery*). Wydaje się więc, że najlepsze rezultaty uzyskamy w wypadku zastosowania kilku współbieżnie działających kopii danego serwera.

2.2. Zwiększenie odporności serwerów na błędy

Jedynym z rozwiązań umożliwiających poprawną pracę systemu w wypadku błędu serwera jest zastosowanie kilku serwerów tworzących swego rodzaju hierarchię. Oprócz głównego serwera mamy tu do czynienia z serwerami zapasowymi, działającymi na innych węzłach i przejmującymi funkcje serwera głównego w wypadku jego awarii. Grupę procesów składającą się z serwera głównego i zapasowych nazywać będziemy serwerem hierarchicznym.

Przykład 1

Duplikacja serwerów występuje np. w przypadku serwisu nazw - DNS (ang. Domain Name Service), stosowanym w sieci Internet. Mamy tutaj do czynienia z serwerem głównym (ang. *primary name serwer)* oraz serwerami zapasowymi (ang. *secondary name servers).* Stacje - klienci serwisu znają tylko nazwy symboliczne innych węzłów w sieci. Aby uzyskać adres sieciowy danego węzła, stacja wysyła zapytanie do serwera nazw. Jeśli serwer nazw zna adres sieciowy żądanego węzła, powinien odesłać go w odpowiedzi do tej stacji. Jeśli główny serwer nie jest osiągalny, jego rolę chwilowo przejmuje serwer zapasowy. W DNS wszystkie serwery konfigurowane są lokalnie. Jest to wykonywane ręcznie przez administratora systemu. Każda nowa nazwa w lokalnej domenie, która ma być znana przez serwer, musi zostać ręcznie do niego wprowadzona przez administratora. Serwer główny po zmianie konfiguracji może rozsyłać kopie swojej bazy danych (czyli listę znanych nazw węzłów) do pozostałych serwerów. Działania klientów nie mają natomiast wpływu na zawartość danych (czyli stan wewnętrzny) serwerów. Dlatego każdy klient może komunikować się niezależnie z którymkolwiek z serwerów i od niego uzyskiwać potrzebne informacje.

Serwery hierarchiczne mają zastosowanie tylko w wypadku, gdy ich stan nie zmienia się w wyniku działań klientów, a wszystkie dane są przez cały czas dostępne na serwerze głównym i na węzłach zapasowych. Inaczej ma się sytuacja, gdy klienci mogą wpływać na stan serwera poprzez uaktualnienie jego danych. Operacja ta musiałaby być dokonywana dla wszystkich węzłów wchodzących w skład serwera hierarchicznego, a więc klient musi znać co najmniej pełną aktualną konfigurację serwera.

Przykład 2

Wyobraźmy sobie inny niż omawiany wyżej DNS model serwisu nazw, w którym stacje same rejestrują swoje nazwy w serwerze. Taki system może być przydatny np. w wypadku, gdy mamy do czynienia ze stacjami mobilnymi, tj. takimi, które w różnym okresie czasu są osiągalne pod różnymi adresami w sieci. W tej sytuacji klient musi mieć (przynajmniej pośredni) wpływ na stan wszystkich kopii serwera - musi zgłosić, w jakiej znajduje się sieci, aby można było przyporządkować jego nazwie jego nowy adres. Każda rejestrowana nazwa

musi być odnotowana we wszystkich kopiach bazy nazw. Baza nazw musi być spójna. Jednocześnie szybkość dokonywania zmian musi być duża - aby nadążyć np. za zmianami adresu przenośnego komputera działającego np. w samochodzie i przemieszczającego się z jednej komórki - sieci - do drugiej.

Proponowaną alternatywą dla serwerów hierarchicznych mogą być **serwery grupowe**. Serwer grupowy (w odróżnieniu od serwera pojedynczego, który działa tylko w postaci jednej kopii oraz serwera hierarchicznego, składającego się z serwera nadrzędnego oraz serwerów zapasowych) składa się z kilku równorzędnych procesów serwera działających na oddzielnych węzłach. Procesy te są idealnie zsynchronizowane - otrzymują te same wiadomości i produkują te same wyniki. Komunikacja z tym serwerem odbywa się nie za pomocą zwykłych wiadomości pomiędzy poszczególnymi procesami, ale za pomocą niezawodnej komunikacji grupowej. W dalszej części zostanie omówiony szczegółowo najważniejszy element potrzebny do stworzenia serwera grupowego - mechanizm niezawodnej komunikacji grupowej.

3. Istniejące protokoły niezawodnej komunikacji grupowej

3.1. Wiadomości ogólne

Prace nad niezawodnym protokołami komunikacji grupowej trwają już od dłuższego czasu i są dość zaawansowane. Opracowane do tej pory protokoły komunikacji grupowej umożliwiają wysyłanie wiadomości od pojedynczego nadawcy do wielu odbiorców jednocześnie. Zapewniają one, że wiadomość będzie odebrana przez wszystkich adresatów, bądź też nie zostanie odebrana wcale. Mówimy, że protokół posiada własność zgodności. Protokoły te zapewniają również tę samą kolejność odebrania wiadomości przez wszystkich członków grupy, czyli własność zachowania kolejności.

Większość systemów komunikacji grupowej do przesyłania wiadomości wykorzystuje protokół Multicast IP, będący rozszerzeniem standardowego protokołu IP, stosowanego na przykład w sieci Internet. W środowiskach, w których protokół ten jest niedostępny, stosuje się inne rozwiązaniana w postaci protokołów niestandardowych realizowanych przez niskie warstwy sieci lub zwykłą komunikację typu punkt-punkt.

Wśród istniejących systemów oferujących mechanizmy komunikacji grupowej przedstawiono protokoły stosowane w systemach ISIS, Newtop, RMP i Totem.

3.2. System ISIS

System ISIS opracowany w Cornell University w USA służy do tworzenia odpornych na błędy aplikacji rozproszonych [4]. System ten jest zorientowany na komunikację grupową pomiędzy procesami. Został zaimplementowany w kilku środowiskach, między innymi na bazie systemu UNIX oraz na bazie mikrojądra systemu MACH. i może używać dla celów komunikacji grupowej różnych protokołów transportowych.

Mechanizm komunikacji grupowej w systemie ISIS jest realizowany przez kilka warstw systemu.

Najniższa warstwa (MUTS - ang. *Multicast Transport Service*) jest odpowiedzialna za dostarczenie wiadomości do członków grupy i zapewnia zgodność protokołu. Korzysta ona z protokołu przesuwnego okna. Okno jest przesuwane, gdy nadawca otrzyma potwierdzenia odebrania pakietu od wszystkich członków grupy. Gdy jest to możliwe, protokół przesyła kilka wiadomości w jednym pakiecie, co znacznie zwiększa wydajność systemu.

Z kolei warstwa CBCAST (ang. *Casual Broadcast*) jest odpowiedzialna za zarządzanie członkostwem grupy oraz wymusza zachowanie kolejności wszystkich wiadomości. Prawidłowa kolejność doręczania odebranych wiadomości jest ustalana na podstawie numerów wiadomości. Każdy węzeł posiada wektor VT zawierający tyle elementów, ile węzłów liczy grupa, do której należy ten węzeł. Element VT[k] informuje, ile wiadomości od węzła P_k otrzymał dany do chwili obecnej. Każda wiadomość wysyłana przez dany węzeł jest uzupełniana o wektor VT', który zawiera stan wektora VT węzła nadawczego w momencie wysłania tej wiadomości. Odebrana wiadomość zostaje doręczona oczekującemu na nią procesowi dopiero wtedy, gdy wiemy, że węzeł odbiorczy odebrał te wszystkie wiadomości, które odebrał również proces nadawczy. Jest to sprawdzane poprzez porównanie elementów lokalnego wektora VT z wektorem VT' otrzymanym w odebranej wiadomości.

System ISIS oferuje podstawowe funkcje dla komunikacji grupowej. Bazowy system nie zawiera funkcji związanych z wykrywaniem awarii węzłów oraz procedur rekonfiguracji oraz innych procedur usługowych. Funkcje te zależą w dużej mierze od potrzeb konkretnej aplikacji i są implementowane jako funkcje użytkownika, a nie funkcje systemowe.

3.3. Protokół Newtop

Inne rozwiązanie zostało zaproponowane w protokole Newtop (ang. *NEWcastle Total Order Protocol*), opracowanym w University of Newcastle w Wielkiej Brytanii [1]. Protokół ten pozwala na współistnienie grup opartych na protokole asymetrycznym oraz na protokole symetrycznym i to w obecności błędów.

Protokół symetryczny umożliwia każdej ze stacji niezależne wysyłanie wiadomości zaadresowanych do grupy. Kolejność wiadomości w symetrycznej wersji protokołu ustalana jest na podstawie logicznych zegarów, czyli kolejnych numerów nadawanych wiadomościom [8]. Każdy węzeł posiada swój licznik LC, który jest zwiększany o jeden w momencie, gdy węzeł wysyła wiadomość. W momencie odebrania wiadomości o numerze większym niż aktualny licznik procesu licznik lokalny jest automatycznie zwiększany do wartości zapisanej w wiadomości. Zakłada się, że dwie wiadomości od tego samego węzła przychodzą w tej samej kolejności, w jakiej zostały wysłane przez ten węzeł (zapewnia to warstwa transportowa sieci). Każdy węzeł posiada tablicę liczników, w której zapisywane są numery wiadomości ostatnio otrzymanych od każdego z członków grupy. Wiadomość może być doręczona do procesu, gdy mamy gwarancję, że do naszego węzła nie dotrze żadna wcześniejsza wiadomość. Wiadomości od różnych węzłów doręczane są do procesu w kolejności rosnących numerów. W wypadku gdy dwie wiadomości mają ten sam numer, doręczane są według z góry ustalonej kolejności, takiej samej na wszystkich węzłach (na przykład według numeru nadawcy).

W wersji asymetrycznej protokołu Newtop procesor, który chce rozesłać wiadomość do danej grupy, przesyła ją najpierw do węzła - zarządcy grupy (ang. *sequencer*). Otrzymuje ona numer zgodnie z regułami dla protokołu symetrycznego, a następnie jest rozsyłana przez węzeł zarządzający do pozostałych członków grupy. Węzeł, który żądał rozesłania danej wiadomości, nie może wysyłać innych wiadomości aż do czasu otrzymania kopii swojej wiadomości rozsyłanej przez zarządcę. Obowiązuje to również węzły należące do kilku grup naraz. Dzięki temu proces, który jest członkiem kilku grup, może otrzymywać wszystkie wiadomości we właściwej kolejności. Co więcej, pozwala to również na to, aby węzeł był jednocześnie członkiem wielu grup asymetrycznych i symetrycznych.

Każdy członek grupy zna aktualny skład grupy, określany jako widok grupy (ang. *view*). Wszystkie zmiany składu grupy, związane z awarią węzła, utratą połączeń albo dobrowolnym odłączeniem się węzła od grupy,wpływają na zmianę widoków grupy, na pozostałych węzłach. Wykrycie uszkodzenia węzła jest możliwe na podstawie braku komunikacji z tym węzłem. Protokół zakłada, że uszkodzony węzeł nie wysyła żadnych wiadomości, dlatego też węzły, które przez dłuższy czas nie wysyłają żadnej wiadomości użytkowej, powinny wysyłać specjalne puste wiadomości potwierdzające ich sprawność. W momencie wykrycia uszkodzenia następuje rekonfiguracja grupy. Algorytm rekonfiguracji jest algorytmem rozproszonym i polega na instalacji nowego składu (widoku) grupy na wszystkich sprawnych węzłach w grupie. Przebiega on w dwu fazach: w fazie pierwszej jednostki ustalają pomiędzy sobą nowy skład grupy, po czym w fazie drugiej na wszystkich jednostkach tworzone są nowe

widoki grupy. Jednostki ogłaszają swoje oskarżenia odnośnie do uszkodzeń innych węzłów, przy czym oskarżenia te mogą być korygowane przez inne jednostki, które otrzymały prawidłowe wiadomości od podejrzanego węzła. Gdy jednostki zdołają ustalić nowy skład grupy, instalowany jest nowy widok, w przeciwnym razie następuje samorozwiązanie grupy.

3.4. Protokół RMP

Protokół RMP (ang. *Reliable Multicast Protokol*) został opracowany w USA na West Virginia University we współpracy z NASA i jest proponowany jako standard dla komunikacji grupowej w Internecie [6]. Węzeł nadawczy wykorzystuje mechanizm okna o dynamicznie regulowanej szerokości do uzyskania maksymalnej wydajności transmisji. Wielkość okna jest automatycznie korygowana w zależności od możliwości odbiorców. Rozpoczynając nadawanie, węzeł systematycznie zwiększa swoje okno. W momencie gdy odbiorcy przestają nadążać z odbiorem pakietów, zaczynają oni wysyłać żądania retransmisji, które automatycznie zmniejszają okno nadawcy.

Protokół zapewnia zgodność wiadomości odbieranych przez poszczególne węzły w oparciu o algorytm Changa i Maxemchuka [5]. Potwierdzanie pakietów opiera się na mechaniźmie przekazywania tokena w pierścieniu utworzonym z węzłów - członków grupy. Zgodnie z algorytmem Ch&M tylko węzeł, będący aktualnym posiadaczem tokena, wysyła potwierdzenia odebrania wiadomości. Jeden pakiet może zawierać potwierdzenie odebrania kilku wiadomości, co znacznie redukuje ilość pakietów przesyłanych w sieci. Jeśli węzeł na podstawie odebranych potwierdzeń ustali, że zgubił jakiś pakiet, wysyła do nadawcy tego pakietu żadanie jego retransmisji. Protokół zapewnia w ten sposób, że wszystkie węzły grupy otrzymają identyczne wiadomości.

Transmisja grupowa może się odbywać z różnym stopniem niezawodności. Aplikacja może dla danej grupy żądać odpowiedniego poziomu jakości transmisji (ang. *quality of service - QoS*). W zależności od tego komunikacja może być całkowicie zgodna i w pełni zachowywać kolejność, może także nie wprowadzać żadnego mechanizmu kontroli poprawności i korekcji błędów w przekazywaniu pakietów.

Ważnym elementem RMP są mechanizmy wykrywania uszkodzeń węzłów i rekonfiguracji grupy. Zakłada się, że uszkodzony węzeł nie wysyła żadnych prawidłowych wiadomości ani potwierdzeń. Ewentualne niepoprawne pakiety generowane przez uszkodzoną jednostkę mogą być odrzucane dzięki zastosowaniu na przykład kryptograficznych algorytmów autentyzacji. Wykrywanie uszkodzeń węzła opiera się na braku potwierdzeń od uszkodzonego węzła. Jeśli węzeł nie prześle potwierdzenia pomimo ustalonej ilości retransmisji wiadomości, jest on uznawany za uszkodzony i wykluczany z grupy. Podobnie jak

w innych protokołach rekonfiguracja grupy odbywa się przy wykorzystaniu dwufazowego algorytmu rekonfiguracji (ang. *two-phase commit*). Rekonfiguracji przewodniczy węzeł, który pierwszy zauważy brak aktywności uszkodzonego węzła. W ten sposób wyeliminowano problem istnienia stałego zarządcy grupy, który zawsze jest odpowiedzialny za przeprowadzenie rekonfiguracji.

3.5. System Totem

System Totem został opracowany na University of California w Santa Barbara, USA. [7]. Działa on w oparciu o protokół typu token-ring, działający w sieci Ethernet. Wszystkie węzły systemu podłączone do tej samej sieci lokalnej przekazują między sobą token. W ramach systemu może istnieć wiele grup jednocześnie. Protokół może działać efektywnie w kilku sieciach lokalnych jednocześnie, wykorzystując wyodrębnione węzły pełniące rolę bramy (ang. *gateway*) pomiędzy poszczególnymi sieciami.

W wymianie tokena uczestniczą wszystkie węzły systemu należące do jednego pierścienia (do jednej fizycznej sieci lokalnej), nawet jeśli należą do różnych grup. Token przekazywany jest pomiędzy węzłami za pomocą zwykłych wiadomości unicastowych (protokół UDP). Mechanizm time-outów i żądań retransmisji zapobiega zgubieniu tokena. Tylko węzeł, który aktualnie posiada token, może rozsyłać wiadomości grupowe w postaci pakietów multicastowych.

Każda wiadomość grupowa uzupełniana jest o numer sekwencyjny i logiczny czas wysłania. Numer sekwencyjny wiadomości nadawany jest na podstawie numeru ostatnio wysyłanej wiadomości, który przekazywany jest za pomocą tokena. Każda kolejna wiadomość wysłana w ringu ma numer sekwencyjny o jeden większy od porzedniej wiadomości. Dzięki temu węzły mogą łatwo ustalić prawidłową kolejność odebranych wiadomości oraz określić, jakie wiadomości zostały zgubione i żądać ich retransmisji. Czas logiczny jest wykorzystywany do ustalenia właściwej kolejności wiadomości przekazywanych pomiędzy sąsiednimi ringami. Każdy węzeł posiada swój logiczny zegar [8]. W momencie gdy węzeł otrzyma wiadomość opatrzoną czasem większym niż czas lokalny, logiczny czas lokalny węzła ustawiany jest zgodnie z czasem wpisanym w odebranej wiadomości.

Token krążący w ringu zawiera ponadto specjalne pole informujące o tym, jaki najwyższy numer pakietu został odebrany przez wszystkie węzły w ringu. Tylko wiadomości, co do których mamy gwarancję, że zostały odebrane przez wszystkie węzły systemu, mogą być dostarczone do procesów użytkowych (aplikacji).

Uszkodzenia węzłów oraz przerwy w topologii sieci są wykrywane za pomocą mechanizmu ustalonych czasów oczekiwania (ang. *time-out*). Węzły, które nie odpowiadają przez określony czas, uznawane są za uszkodzone i usuwane z systemu. Nawet jeśli dany węzeł nie jest uszkodzony, a tylko chwilowo spowolniony, to i tak po uzyskaniu gotowości musi wykonać procedurę włączania się do systemu jako nowy węzeł. Jednakże dzięki mechanizmowi krótkich time-outów możliwe jest szybkie wykrycie uszkodzeń występujących w systemie. Specjalny rodzaj wiadomości - *Configuration Change* - jest rozsyłany do wszystkich pozostałych węzłów przez węzeł, który wykrył uszkodzenie sąsiada. Wiadomości te są analizowane przez węzły będące bramami (ang. *gateway*). Na ich podstawie buduje się dynamicznie mapę systemu.

4. Niezawodny protokół komunikacji dla potrzeb serwerów grupowych

4.1. Motywy stworzenia nowego protokołu

Omówione protokoły komunikacji grupowej są protokołami ogólnego przeznaczenia, obsługującymi wysyłanie wiadomości od pojedynczego nadawcy do grupy wielu odbiorców. W wypadku tworzenia serwerów grupowych protokół komunikacyjny powinien zapewnić nie tylko niezawodne dostarczenie wiadomości grupowych do wszystkich węzłów wchodzących w skład serwera, ale także umożliwić synchronizację pracy serwera i pozwalać na lokalizację uszkodzonych węzłów składowych. Dlatego też zaproponowano do tego celu nowy protokół komunikacji grupowej. Został on opracowany przez autora pracy specjalnie na potrzeby tworzenia serwerów grupowych. Protokół ten, o nazwie GREP (Group REliable Protocol) pozwala nie tylko na odbieranie wiadomości do grupy, ale również na uzgadnianie treści i wysyłanie wiadomości od grupy jako całości. Dzięki temu możliwa jest dokładniejsza diagnoza węzłów wchodzących w skład serwera grupowego, pozwalająca na wykrycie nie tylko całkowicie uszkodzonych, ale i błędnie działających węzłów.

4.2. Struktura protokołu GREP

Protokół GREP zapewnia niezawodną komunikację dwustronną pomiędzy grupą procesów rezydujących na różnych węzłach sieci (serwerem grupowym) a pojedynczym procesem (klientem) bądź inną grupą procesów (innym serwerem grupowym). Komunikacja odbywa się poprzez przesyłanie wiadomości. Wiadomości mogą być przesyłane z potwierdzeniem lub bez potwierdzeń.

Klient korzystający z serwera grupowego nie zna dokładnie jego bieżącej konfiguracji. Zamiast komunikowania się kolejno z poszczególnymi węzłami klient komunikuje się z serwerem wysyłając komunikat zaadresowany do całej grupy. Wiadomości zwrotne otrzymuje on od jednego z członków grupy, działającego w imieniu całej grupy i po ustaleniu treści komunikatu ze wszystkimi sprawnymi członkami grupy.

Protokół GREP jest protokołem typu token-rong, w którym token krąży pomiędzy wszystkimi węzłami w grupie. Protokół ma budowę warstwową. W ramach całego protokołu komunikacji grupowej wyróżniamy następujące podprotokoły:

- protokół NMULTI (Non-reliable **Multi**cast Protokol) - protokół najniższej warstwy, zapewnia rozsyłania pakietów zaadresowanych do grupy odbiorców,
- protokół GRING (Group **Ring** Protocol) - protokół komunikacji w ramach grupy. Umożliwia synchronizację pracy grupy: synchronizację odbieranych i wysyłanych wiadomości oraz wykrywanie awarii i przeprowadzanie zmian składu grupy,
- protokół GRREC (Group **R**eliable **Rec**eive Protocol) - zapewnia niezawodne przesyłanie wiadomości od pojedynczego klienta do grupowego serwera,
- protokół GRSEND (Group Reliable **Send** Protocol) - zapewnia zsynchronizowanie węzłów grupy, ustalenie treści komunikatu i przesłanie wiadomości od grupy do klienta.

Można wyróżnić następujące własności protokołu GREP:

W1. Zgodność odebranych wiadomości - wiadomość musi być dostarczona do wszystkich węzłów w grupie bądź do żadnego z nich.

W2. Ogólna kolejność odebranych wiadomości - wiadomości odebrane są dostarczone do wszystkich węzłów w takiej samej kolejności.

W3. Częściowa kolejność odebranych wiadomości - dwie wiadomości wysłane przez jednego nadawcę są odebrane w tej samej kolejności, jak zostały nadane.

W4. Zgodność wysyłanych wiadomości - wiadomość wysyłana przez grupę jest uprzednio potwierdzona przez wszystkie sprawne węzły w grupie.

Zauważmy, że własności W1, W2 i W3 są spełnione przez większość dotychczasowych protokołów komunikacji grupowej. Natomiast wprowadzona własność W4 pozwala na tworzenie w oparciu o ten protokół niezawodnych serwerów grupowych.

4.3. Protokół NMULTI

Protokół NMULTI stanowi najniższą warstwę systemu komunikacji grupowej. Zadaniem protokołu jest rozsyłanie pakietów zaadresowanych do określonej grupy odbiorców. Protokół

nie zapewnia dostarczenia wysłanego pakietu do wszystkich odbiorców ani poprawnej kolejności odebranych pakietów.

Jako NMULTI może być wykorzystany Multicast IP - standardowy protokół rozsyłania pakietów do adresów grupowych IP, będący rozszerzeniem protokołu IP [2], [3]. W wypadku gdy system nie zapewnia korzystania z Multicast IP, warstwę tę można zasymulować używając zwykłej komunikacji typu punkt - punkt.

4.4. Protokół GRING

Protokół GRING bezpośrednio korzysta z NMULTI. Węzły wchodzące w skład grupy tworzą pierścień, w którym przekazują sobie znacznik - token. Jest to więc protokół typu token-ring.

Mechanizm przekazywania tokena w grupie N węzłów działa następująco: Inicjalnie każdy węzeł w grupie otrzymuje kolejno swój numer (1, 2, ...). Węzeł o najniższym w danym momencie numerze (1), który zainicjował powstanie grupy, jest posiadaczem tokena. Inicjuje on pracę ringu. Przekazuje on token do stacji 2, a sam uaktualnia swój numer, tak aby jego numer był najwyższy w grupie. Odbywa się to zgodnie z formułą:

$n'_i = n_i + N;$

gdzie: n'_i - nowy numer węzła i w grupie.

n_i - numer węzła i w grupie przed przesłaniem tokena,

N - aktualna ilość węzłów w grupie.

Token przekazywany jest w postaci pakietu multicastowego, przesyłanego do wszystkich węzłów w grupie. Jest to istotna różnica w stosunku do dotychczasowych protokołów pierścieniowych, które bazowały na komunikacji punkt-punkt. Przekazywanie tokena odbywa się za pomocą protokołu NMULTI.

Token niesie ze sobą informację o stanie grupy, o odebranych komunikatach zaadresowanych do grupy, o komunikatach przygotowywanych do wysłania przez grupę i o komunikatach już wysłanych, których odbiór nie został jeszcze potwierdzony (lista A).

Token ma postać:

$M(G, m, n_i, S_N, R, W, A)$,

gdzie:

- G - nazwa grupy (adres grupowy),
- m - generacja tokena (numer kolejny tokena, który służy do wykrywania zduplikowanych tokenów),
- n_i - numer węzła ostatniego nadawcy tokena,
- S_N - lista, zawierająca status węzłów wchodzących w skład grupy (ang. *nodes state list*),

R - lista wiadomości odebranych i nie potwierdzonych przez wszystkie węzły grupy (ang. *recevid messages list*),

W - lista wiadomości oczekujących na wysłanie (ang. *waiting messages list*),

A - lista wysłanych wiadomości oczekujących na potwierdzenie odbioru (ang. *non-acknowleged messages list*).

Każda lista składa się ze zmiennej liczby rekordów, przy czym rekordy w każdej z list mają ustalone stałe formaty, które będą omówione w dalszej części pracy. Lista R jest obsługiwana przez protokół GREC, a listy W i A przez protokół GRSEND.

Token krąży w pierścieniu z różną częstotliwością. W wypadku gdy do grupy nie dotarł żaden komunikat ani też serwer grupowy nie chce wysłać żadnej wiadomości, token pełni tylko rolę kontrolną. W związku z tym nie musi być on przesyłany zbyt często. Przyjmuje się, że token zostanie odesłany przez węzeł n_i po czasie t_{nor} od odebrania go od węzła n_{i-1}, jeśli lista L w tokienie jest pusta i węzeł n_i nie odebrał w tym czasie żadnego pakietu zaadresowanego do grupy.

W wypadku gdy do grupy dotarł komunikat grupowy do niej zaadresowany, token pozwala na potwierdzenie przez wszystkie węzły grupy odebrania tego pakietu. W celu zwiększenia efektywności działania systemu przyjmuje się, że token z niepustą listą R (tj. taki, który zawiera potwierdzenia odebrania komunikatu) zostaje zmodyfikowany i odesłany przez węzeł n_i, kiedy tylko zostanie odebrany od węzła n_{i-1}. Przyjmuje się, że czas potrzebny na modyfikację i odesłanie tokena zawierającego potwierdzenia przyjęcia pakietów t_{rc} jest krótszy niż normalny czas cyrkulacji tokena ($t_{rc} < t_{nor}$).

Poważną wadą protokołów typu token-ring jest ich wrażliwość na utratę tokena. Token może być utracony w wyniku:

- awarii stacji - posiadacza tokena,
- nieodebrania tokena przez stację przeznaczenia z powodu chwilowego braku zasobów,
- uszkodzenia tokena w czasie transmisji/odbioru (błędy transmisji).

W protokole GRING wykrywanie utraty tokena opiera się na mechanizmie określonych, maksymalnych czasów oczekiwania na token (ang. *time-out*). Każda stacja odbiera wszystkie wiadomości związane z przekazywaniem tokena - zna więc aktualnego posiadacza tokena. Ponieważ przyjmuje się, że token będzie odesłany przez stację - posiadacza tokena najpóźniej po czasie t_{nor}, więc stacja n_i spodziewa się, że najpóźniej po czasie $t_{not} = t_{ns}+t_{delay}$ token zostanie jej przekazany. Wartość t_{delay} powinna zostać tak dobrana, żeby zminimalizować prawdopodobieństwo, że poprawny token zostanie odebrany z opóźnieniem, spowodowanym np. przepełnieniem stacji pośredniczących.

Jeśli stacja n_i nie otrzyma w spodziewanym czasie t_{rt} tokena od stacji n_{i-1}, powinna sama wygenerować nowy token. Jeśli ostatnio usłyszana przez stację n_i wiadomość przekazania tokena miała postać:

M(G, m, n_{i-2}, S_N, R, W, A),

to nowy token powinien mieć postać:

M(G, m+1, n_i, S'_N, R, W, A).

Podobna procedura wytworzenia nowego tokena powinna być zrealizowana, gdy węzeł w czasie $2*t_{rt}$ od odebrania wiadomości M od węzła n_{i-2} nie otrzymał tokena ani nie odebrał wiadomości o przekazaniu tokena z n_{i-2} do n_{i-1}. Zasada ta powinna być prawdziwa dla $k < N/2$ interwałów czasowych t_{rt} od czasu usłyszenia ostatniej wiadomości o wymianie tokena.

Gdy węzeł przez czas $k > N/2$ nie odebrał żadnej wiadomości o wymianie tokena, może domniemywać, że sieć komunikacyjna została przerwana, a on sam stanowi wyizolowany element.

W wypadku podziału sieci na kilka niezależnych fragmentów (ang. *network partition*) przyjmujemy, że tylko część oryginalnej grupy, zawierająca większość węzłów, może działać dalej jako ta grupa (nie jest dopuszczalne, aby działały dwa wyizolowane identyczne serwery grupowe). Dlatego też grupa, której liczebność zmniejsza się jednorazowo o ponad połowę, przestaje istnieć.

Dzięki mechanizmowi numerowania tokena możliwe jest wykrycie istnienia zduplikowanych tokenów w pierścieniu. Sytuacja taka może zaistnieć wtedy, gdy token poprzedniej generacji przybędzie do stacji z opóźnieniem już po wygenerowaniu przez nią nowego tokena. Każdy węzeł, który otrzyma poprzedni token, po odebraniu nowo wygenerowanego tokena nie powinien przesyłać dalej tokena starszego.

Lista statusu S_N ma postać $S_N=\{t, l_1, l_2, l_3, ...l_N\}$. Pole t oznacza czas, po jakim lista powinna być reinicjalizowana. Inicjalnie $l_1, l_2, ..,l_N$ są równe i wynoszą L. Wartość L określa, ile maksymalnie razy dany węzeł może zgubić token. W wypadku gdy węzeł n_i nie otrzyma w przewidzianym czasie tokena od węzła n_{i-1}, podczas generowania nowej wiadomości M modyfikuje listę S_N w ten sposób, że zmniejsza licznik l_{i-1} dla węzła, który nie odesłał tokena.

Jeśli licznik l dla któregoś z węzłów - członków grupy osiąga wartość zero, to zostaje on uznany za uszkodzony - i następuje operacja rekonfiguracji grupy (ustalenia jej nowego składu).

Lista S_N powinna być wypełniania wartościami inicjalnymi L w momencie, gdy pole czasu t zostaje wyzerowane, przy czym okres czyszczenia listy powinien być dość długi w stosunku do okresu obiegu tokena w pierścieniu.

4.5. Protokół GRREC

Protokół GRREC zapewnia niezawodny odbiór wiadomości zaadresowanych do grupy (własności W1 - W3). Każda otrzymywana wiadomość jest opatrzona nagłówkiem zawierającym identyfikator nadawcy (jego adres) oraz numer kolejny wiadomości wysłanej przez tego nadawcę. Dzięki temu możliwe jest ustalenie prawidłowej kolejności wiadomości odebranych od tego samego nadawcy (własność W3). Możliwe jest również wykrycie wiadomości zduplikowanych - wygenerowanych np. gdy nadawca nie otrzymał potwierdzenia odbioru. Wiadomość może zawierać również opcjonalne pole, informujące, czy ma być wygenerowane potwierdzenie. Jako domyślne zachowanie systemu przyjmujemy, że grupa po odebraniu wiadomości generuje potwierdzenie. Gdy jednak w wiadomości zawarte jest jawnie polecenie niepotwierdzania jej, żadne potwierdzenie nie jest odsyłane.

Lista R jest listą potwierdzeń odebranych wiadomości. Dla każdej wiadomości odebranej przez węzeł i tworzony jest rekord postaci

$m_{S(,nr)} = (S, nr, (ack_1, ack_2, ack_3, ...ack_N))$,

w którym S oznacza adres węzła (lub grupy) będącego nadawcą wiadomości, nr - numer kolejny wiadomości wygenerowanej przez S, a lista $(ack_1, ...ack_N)$ informuje, które węzły już potwierdziły odebranie wiadomości S(nr), a które nie. Zauważmy, że rekord listy R nie zawiera treści odebranej wiadomości S(nr) - zakładamy, że wiadomość S(nr) może być odebrana przez stację bądź w poprawnej postaci, bądź nie będzie odebrana w ogóle.

Każdy węzeł, który otrzymał token, porównuje zawartość listy R z lokalną kolejką odebranych wiadomości. Jeśli w kolejce odebranych wiadomości jest wiadomość S(nr), to węzeł potwierdza to poprzez ustawienie odpowiedniego pola ack_i. Jeśli natomiast węzeł nie odebrał wiadomości S(nr), to wysyła żądanie retransmisji zgubionego pakietu do nadawcy pakietu. Nadawca pakietu dokonuje retransmisji komunikatu z nie zmienionym numerem kolejnym wiadomości. Dzięki temu można odróżnić wiadomości retransmitowane od wiadomości nowych - węzeł, który odebrał dwie wiadomości od tego samego adresata i z tym samym numerem po prostu ignoruje drugą wiadomość. Pakiety retransmitowane są wysyłane z adresem grupowym odbiorcy - wobec tego może z nich skorzystać nie tylko ta stacja, która wysłała żądanie retransmisji, ale również inne stacje w grupie, które nie otrzymały jeszcze danej wiadomości.

Gdy odebranie wiadomości zostaje potwierdzone przez wszystkie węzły w grupie, to odpowiedni rekord zostaje usunięty z listy R.

4.6. Protokół GRSEND

Zadaniem protokołuGRSEND jest uzgodnienie treści wiadomości, które mają być wysłane przez grupę, a następnie ich wysłanie i uzyskanie potwierdzenia odbioru wiadomości przez adresatów. Zakładając, że wszystkie węzły w grupie zachowują się tak samo, to znaczy że wykonują ten sam program, możemy przyjąć, że wszystkie one w pewnym momencie będą chciały wysłać tę samą wiadomość do określonego adresata. Szybkość przetwarzania węzłów może być różna, więc konieczne jest zsynchronizowanie pracy tych węzłów. Ponadto, w wyniku pojawienia się błędów niektóre węzły mogą generować złe wiadomości bądź nie generować ich wcale. Zadaniem protokołu GRSEND jest wykrywanie tego typu niezgodności.

Protokół GRSEND korzysta z dwu list przekazywanych pomiędzy węzłami grupy za pomocą tokena. Lista W jest listą wiadomości czekających na wysłanie. Lista A jest listą wiadomości wysłanych, których odbiór przez adresata nie został jeszcze potwierdzony.

Lista W wiadomości przygotowanych do wysłania składa się z rekordów o postaci:

$m_{R(,nr)} = (t, R, nr, (M_1, ack_1), (M_2, ack_2), (M_3, ack_3), ..., (M_N, ack_N))$,

gdzie:
- t - stempel czasowy (dopuszczalny czas uzgadniania wiadomości),
- R - adres węzła (lub grupy), do którego skierowana jest wiadomość,
- nr - numer kolejny wiadomości wygenerowanej do R,
- M_k - propozycja węzła k treści wiadomości, która ma być wysłana do R,
- ack_i - 0 gdy węzeł i nie wyraził zgody na wysłanie wiadomości R(nr),
  1 gdy węzeł i wyraża zgodę na wysłanie wiadomości R(nr) .

Kolejny rekord jest tworzony przez węzeł, który chce wysłać wiadomość R(nr) i jest w danym momencie posiadaczem tokena. Węzeł ten zostaje koordynatorem nadania R(nr).

W odróżnieniu od protokołu GRREC, w rekordzie listy W musi znaleźć się treść M wiadomości R(nr), tak aby każdy z węzłów grupy mógł porównać swoją wersję wiadomości z wiadomością wzorcową, wygenerowaną przez koordynatora. W praktyce zamiast całych wiadomości M_k wystarczy tu umieścić ich sygnatury (np. sumy kontrolne). Wtedy węzły mogą zbadać zgodność sygnatury wiadomości R(nr) przygotowanej przez siebie i wiadomości R(nr) wygenerowanej przez inne węzły w grupie.

Każdy węzeł po osiągnięciu stanu, w którym ma gotową do wysłania wiadomość R(nr), umieszcza jej sygnaturę w rekordzie. Wiadomość zostanie wysłana do adresata po uzgodnieniu jej treści przez większość węzłów w grupie. Nadawcą wiadomości powinien być węzeł, który pierwszy osiągnie większość zgodnych wartości M dla nowej wiadomości. Węzeł ten - koordynator nadawania - powinien też usunąć w odpowiednim momencie rekord wiadomości R(nr) z listy W wiadomości oczekujących na wysłanie.

Pole t rekordu wiadomości R(nr) służy do kontroli czasu uzgadniania wiadomości do wysłania. Ponieważ niektóre węzły mogą być niesprawne bądź też działać zbyt wolno, nie można czekać w nieskończoność, aż wszystkie węzły w grupie potwierdzą gotowość do wysłania wiadomości. Po ustalonej ilości obiegów tokena (parametr t) wiadomość R(nr) powinna być wysłana, jeśli tylko potwierdziła ją większość węzłów w grupie. Pozostałe węzły mogą być w tym wypadku zdiagnozowane jako niesprawne. Jeśli natomiast wiadomość nie została potwierdzona przez większość węzłów w grupie, powinna być odrzucona, a węzły, które ją próbowały wygenerować - zdiagnozowane jako niesprawne.

Jeśli wiadomość R(nr) wymaga potwierdzenia odbioru od adresata, zostaje ona umieszczona na liście wiadomości czekających na potwierdzenie A. Wiadomość tę umieszcza na liście W proces - koordynator nadawania po wysłaniu jej i usunięciu z listy W. Lista A składa się z rekordów odpowiadających wysłanym wiadomościom. Każdy rekord ma postać:

$m_{AckR(,nr)} = (t, R, nr, retr)$,

gdzie: t - stempel czasowy (dopuszczalny czas oczekiwania na potwierdzenie),
R - adres węzła (lub grupy), do którego skierowana jest wiadomość,
nr - numer kolejny wiadomości wygenerowanej do R,
retr - dopuszczalna ilość retransmisji wiadomości.

Stempel czasowy t jest zmniejszany przy każdym obiegu tokena. Pozwala on na zliczanie czasu oczekiwania na odebranie potwierdzenia. Wyzerowanie pola t oznacza, że czas oczekiwania na odebranie potwierdzenia minął i wiadomość powinna być retransmitowana raz jeszcze. Ponieważ treść wiadomości została już uzgodniona uprzednio, więc w przypadku retransmisji wiadomość może być odesłana natychmiast przez dowolny węzeł. Przyjmujemy, że wiadomość zostaje retransmitowana przez ten węzeł, który wyzerował pozycję t w rekordzie wiadomości R(nr). Po powtórnym wysłaniu wiadomości odpowiednio zmniejszana jest wartość pola retr, a czas oczekiwania t zostaje na nowo ustawiony na wartość inicjalną.

Wiadomości od grupy wysyłane są z adresem grupowym nadawcy. Dlatego też potwierdzenie odbioru wysyłane przez odbiorcę jest zaadresowane do grupy. Dowolny węzeł w grupie, który odbierze potwierdzenie komunikatu R(nr), usuwa rekord odpowiadający temu komunikatowi z listy A. Jest to znak dla pozostałych węzłów w grupie, że wiadomość została potwierdzona. Można wówczas usunąć treść wiadomości R(nr) z bufora wiadomości oraz przekazać sygnał poprawnego wysłania wiadomości do procesów, które tę wiadomość wygenerowały.

4.7. Protokół RECRING

Protokół RECRING jest odpowiedzialny za zmiany w składzie grupy. Pozwala na przebudowanie pierścienia w wypadku awarii jednego lub kilku węzłów. Zapewnia, że wszystkie węzły znają ten sam skład grupy. Po rekonfiguracji pierścienia wszystkie węzły muszą mieć ten sam status komunikacji, tj. odebrać te same wiadomości i osiągnąć zgodność przed wysłaniem wszystkich wiadomości. Wiadomości nie odebrane przez wszystkie węzły są odrzucane, a wiadomości, których wysłanie nie zostało potwierdzone przez wszystkie węzły, są tworzone na nowo.

Tworzenie nowego pierścienia odbywa się pod kierunkiem węzła, który pierwszy wykrył awarię. Operacja ta jest realizowana na bazie dwufazowego algorytmu rekonfiguracji (ang. *two-phase commit*). Przebiega w następujących krokach:

KROK 1: Węzeł - inicjator rozsyła propozycje nowego składu grupy,

KROK 2: Sprawne węzły odsyłają swoje odpowiedzi - potwierdzające lub nie nowy skład grupy,

KROK 3: Węzeł - inicjator po otrzymaniu potwierdzeń od wszystkich uznawanych za sprawne węzłów ustanawia nowy skład grupy i rozsyła do pozostałych członków grupy,

KROK 4: Wygenerowanie nowego tokena i rozpoczęcie pracy grupy.

W pierwszym kroku węzeł - posiadacz tokena musi ustalić proponowany nowy skład grupy. Nowa grupa tworzona jest z węzłów uznanych za sprawne. Propozycja nowego składu grupy zostaje kilkakrotnie rozesłana jako pakiet multicastowy do wszystkich potencjalnych członków nowej grupy. Następnie (w drugim kroku) węzeł - koordynator oczekuje na potwierdzenie nowego składu grupy od wszystkich jej nowych członków. W wypadku gdy nie otrzyma on potwierdzenia od któregoś z proponowanych członków grupy - usuwa go z niej. W trzecim kroku nowy stan grupy jest jeszcze raz rozesłany do wszystkich jej członków. W tym też kroku węzłom nadawane są kolejne numery w grupie - tworzona jest struktura logiczna pierścienia. W ostatnim, czwartym kroku tworzony jest nowy token. Wiadomości nie odebrane przez wszystkich uczestników grupy albo nie uzgodnione do wysłania są usuwane. Każdy węzeł po otrzymaniu pierwszego po rekonfiguracji tokena kontroluje swoje bufory i ewentualnie usuwa niepotrzebne (np. odrzucone) wiadomości oraz próbuje jeszcze raz ustalić i wysłać wiadomości gotowe.

4.8. Struktura oprogramowania węzła

Oprogramowanie węzła sieci, który wchodzi w skład grupy (serwera grupowego), składa się z dwu zasadniczych części: komunikacyjnej i użytkowej. W skład części komunikacyjnej wchodzi przede wszystkim proces komunikacyjny. Jest on odpowiedzialny za komunikację z pozostałymi węzłami wchodzącymi w skład grupy. Dla węzłów wchodzących w skład wielu grup musi istnieć wiele procesów komunikacyjnych - każdy dla obsługi jednej grupy. Proces komunikacyjny z jednej strony komunikuje się z interfejsem sieciowym poprzez gniazda (ang. *sockets*), odbierając i nadając komunikaty z adresami grupowymi. Z drugiej strony proces komunikacyjny komunikuje się z procesem (procesami) użytkowym tworzącym oprogramowanie serwera za pomocą specjalnej biblioteki. Oferuje ona funkcje zbliżone do standardowych funkcji komunikacyjnych: wyślij wiadomość, odbierz wiadomość, odczytaj status, a także funkcje do zarządzania grupą: utwórz grupę, dołącz do grupy, odłącz się od grupy, usuń grupę. Zastosowanie biblioteki pozwala na stworzenie różnych interfejsów programistycznych (C, C++, ...) pozwalających na korzystanie z protokołu, przy czym grupowy charakter komunikacji powinien być dla aplikacji możliwie przeźroczysty. Biblioteka powinna oferować zarówno funkcje blokujące (typu wyślij i czekaj na potwierdzenie odbioru), jak i nieblokujące (typu: wyślij i natychmiast wróć).

Komunikaty odebrane i przygotowane do nadania są przechowywane w buforach procesu komunikacyjnego. Komunikaty odebrane z sieci są usuwane z bufora, gdy zostają dostarczone do procesu użytkowego bądź odrzucone. Komunikaty przygotowane do nadania są usuwane, gdy grupa otrzyma potwierdzenie ich odbioru. Obsługa poszczególnych protokołów (GRING, GREC, GSEND) może być wykonywana jako oddzielne wątki (ang. *threads*) procesu komunikacyjnego, co pozwala na szybszą obsługę pakietów dla maszyn wieloprocesorowych. Jako generalną zasadę należy jednak przyjąć, że szczegóły implementacji mechanizmów komunikacyjnych na różnych platformach nie powinny wpływać na sposób działania protokołu.

5. Podsumowanie

Przedstawiona w pracy koncepcja serwera grupowego jest w dalszym ciągu tematem prac projektowych i implementacyjnych. Przewidywane dalsze prace projektowe koncentrować się będą nad doskonaleniem protokołu komunikacji grupowej. Powinny one prowadzić do opracowania lepszego wspomagania obsługi węzłów należących do wielu grup jednocześnie, a

także do usprawnienia procesu diagnozowania uszkodzonych węzłów. W szczególności, diagnoza węzła jako uszkodzonego dokonana w jednej grupie powinna być propagowana do innych grup, do których ten węzeł należy. W innych grupach diagnoza ta może być weryfikowana (np. przez specjalne zapytania testowe), a nie tworzona na nowo, co jest kosztowne i długotrwałe.

Prace implementacyjne koncentrują się obecnie na implementacji pierwszej wersji protokołu komunikacyjnego w heterogenicznym środowisku sieciowym, opartym na komputerach działających pod systemami operacyjnymi HP-UX oraz LINUX. Niezależnie prowadzone są prace związane z opracowaniem mechanizmów odtwarzania poprawnego stanu węzła na podstawie stanu pozostałych węzłów wchodzących w skład grupy.

LITERATURA

[1] Ezjilchelvan P.D., Macedo R.A., Shrivastava S. K.: Newtop: A Fault-Tolerant Group Communication Protocol, Technical Raport, University of Newcastle, 1991.

[2] Waitzman D., Partridge C., Deering S.: Distance Vector Multicast Routing Protocol, RFC 1075, November 1988.

[3] Deering S.: Host Extensions for IP Multicasting, RFC1112, August 1989.

[4] van Renesse R., Birman K.: Fault-Tolerant Programming Using Process Groups, IEEE 1994.

[5] Chang J. M., Maxemchuk N. F.: Reliable Broadcast Protocols, ACM Transactions on Computer Systems, August 1984.

[6] Montgomery T., Callahan J. R., Whetten B.: Fault Recovery in the Reliable Multicast Protocol, International Conference on Distributed Computing, September 1996.

[7] Moser L.E., Melliar-Smith P.M., Agarwal D.A., Budhia R.K., Lingley-Papadopulosc C.A.: Totem: A Fault-Tolerant Multicast Group Communication System, Communications of the ACM, Vol. 39 No. 4, April 1996.

[8] Lamport L.: Time, clocks and the ordering of the events in a distributed system, Communications of the ACM, vol. 21 no. 7, July 1978.

[9] Krawczyk H., Umiński P. W,: Węzły wirtualne jako metoda zwiększenia odporności na błędy aplikacji działających w systemach rozproszonych, Informatyka 10/95.

Recenzent: Dr inż. Katarzyna Stąpor

Wpłynęło do Redakcji 21 listopada 1996 r.

Abstract

The client-server programming model is a very popular model for constructing large - scale distributed systems. This model allows to specify and fulfil different reliability requirements for different parts of the system. The new idea of a highly reliable service is presented in this paper. This service, called a group server, is performed on several different nodes (computers) at the same time. Clients send messages to the server using reliable multicast protocol. The messages are addressed to the group rather then to the particular node. Due to the use of this protocol all nodes in the group can receive and generate the same messages. Therefore, all results obtained by all replicas should be the same - and a failure of one of the replicas does not lead to the failure of the whole service.

There are many reliable multicast protocols described in the literature, which use different techniques for obtaining the requested level of quality and reliability. We will describe some of them in a little more detail (system ISIS, system Newtop, protocol RMP and protocol Totem). However, those protocols are general-purpose multicast protocols and are not best suited for the construction of group servers. Because of this we will create a new multicast protocol, GREP (Group Reliable Protocol) to be used to access our group server.

This protocol allows clients to send a message to the group and guarantees, that all members of the group will receive the correct message. It allows also to generate a common response from the group, based on results obtained from each member of the group server. GREP is a token-ring based protocol, where the token is used to synchronise all members of the group. This protocol is a multilevel protocol; different subprotocols are responsible for passing token and group synchronisation, receiving messages, voting and sending messages, and detecting and diagnosing faulty nodes. The node, which is producing results differing from the results of the other nodes is diagnosed as faulty and removed from the group until a successful recovery action is performed.