ROZDZIAŁ XXXIII

KOMUNIKACJA W CZASIE RZECZYWISTYM W SIECI INTERNET

Rozdział prezentuje problemy i rozwiązania zastosowane podczas realizacji złożonego systemu komunikacyjnego, który może być wykorzystany nie tylko w sieci wewnętrznej ale również w sieci Internet. System został zrealizowany na Politechnice Wrocławskiej w ramach projektu e-Informatyka. Główne problemy i zaproponowane rozwiązania dotyczą synchronizacji strumieni danych i głosu, transmisji slajdów wchodzących w skład prezentacji, ograniczeń nakładanych przez model bezpieczeństwa na platformie Java oraz miksowania cyfrowych strumieni audio. System pozwala na równoległą komunikację głosową, tekstową (chat) jak i pokaz slajdów wielu użytkownikom jednocześnie. Wszystko to jest możliwe przy minimalnych wymaganiach sprzętowych. Dzięki zastosowanym rozwiązaniom powyższa funkcjonalność jest dostępna nawet dla użytkowników korzystających z wolnych połączeń modemowych.

1. POWODY STWORZENIA SYSTEMU

Rzeczywiste potrzeby posiadania systemu do przeprowadzania konferencji oraz nauczania z wykorzystaniem sieci Internet i technik multimedialnych a także pracy w rozproszonych zespołach, zmusiły autorów do stworzenia platformy komunikacyjnej jako części portalu e-Informatyka. Jeden z autorów miał okazje wcześniej prowadzić wykłady dla studentów wykorzystując sieć Internet. Tą drogą studenci otrzymywali wszystkie niezbędne materiały oraz treści wykładów. Musieli samodzielnie przygotować się do zaliczenia egzaminu końcowego. Najważniejszą wadą powyższego podejścia był brak bezpośredniego kontaktu pomiędzy studentami a wykładowcą.

Przedstawiony system zapewnia ciągły kontakt głosowy wykładowcy i studentów, pozwalając słuchaczom na aktywny udział w dyskusji. Zalety takiego rozwiązania są oczywiste. Pozwala ominąć wiele ograniczeń czasowych, logistycznych i finansowych (np. związanych z rezerwacją sal konferencyjnych czy kosztami dojazdów). Tego typu problemy występowały

Lech Madeyski, Adam Mrozowski, Sebastian Gil: Politechnika Wrocławska, Wydział Informatyki i Zarządzania; Wybrzeże Wyspiańskiego 27, 50-370 Wrocław; lech.madeyski@pwr.wroc.pl, adam.mrozowski@pwr.wroc.pl, create@klub.chip.pl

między innymi podczas seminariów organizowanych na Politechnice Wrocławskiej przez firmy Microsoft czy Sun Microsystems. Wymagania sprzętowe nakładane na użytkowników naszego systemu są bardzo niewielkie – wystarczy komputer z kartą dźwiękową i dostępem do Internetu. Początkowo rozwiązaniem wydawały się komercyjne produkty do transmisji strumieni audio i wideo. Jednak analiza rozwiązań Microsoft i RealNetworks wykazała, że możliwość ich praktycznego zastosowania do prowadzenia wykładów on-line w polskich realiach jest niewielka. Systemy te umożliwiają komunikację w Internecie jednak e-nauczanie (ang. e-Learning) kładzie również nacisk na inne aspekty, takie jak możliwość transmisji slajdów czy możliwość zarządzania użytkownikami (przyznawania głosu poszczególnym użytkownikom). Należy przy tym zapewnić minimalne koszty udziału studentów w takiej formie nauki. Nie muszą oni posiadać kamer internetowych a co najważniejsze powinni mieć możliwość dostępu do prezentacji z różnych miejsc bez konieczności korzystania z szybkich łącz sieciowych. MG/KRC Poland Media w styczniu 2003 opublikowało dane mówiące, że w Polsce ok. 5 milionów ludzi używa Internetu i 70% z nich stosuje modemy wolniejsze niż 115Kbps. Taka przepustowość jest za niska aby, korzystając z istniejących rozwiązań, brać udział w wykładach na odległość. Powyższe ograniczenia zmusiły autorów do stworzenia nowego systemu umożliwiającego efektywne nauczanie w czasie rzeczywistym z wykorzystaniem sieci Internet i połączeń sieciowych o niskich przepustowościach. Kluczowym zadaniem okazało się zapewnienie synchronizacji pomiędzy wykładowcą a studentami.

2. CHARAKTERYSTYKA SYSTEMU

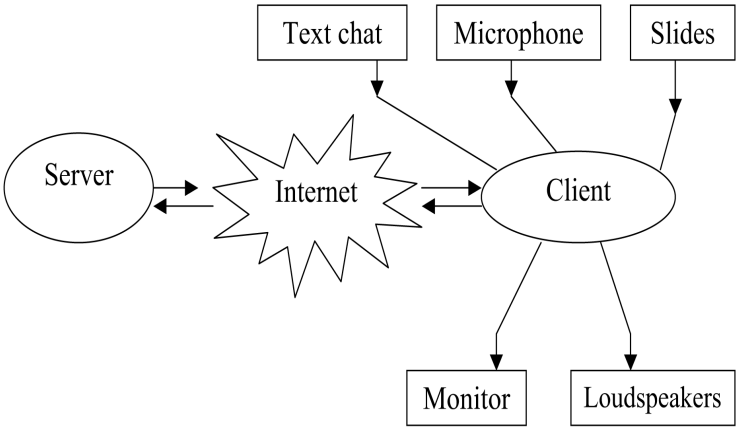
Zdecydowano się na użycie protokołu RTP (ang. *Real Time Transport Protocol*) jako protokołu bazowego. Na rys. 33.1 przedstawiono elementy systemu. Pierwszym z nich jest serwer. Jest on odpowiedzialny za:

- odbieranie informacji pochodzących od wykładowcy i od studentów,
- synchronizację danych,
- wysyłanie informacji do studentów.

Praktycznie wiąże się to z odbieraniem i łączeniem strumieni wejściowych po stronie serwera. Zanim zaczniemy używać systemu dla odbierania i wysyłania danych musimy zapewnić strumienie komunikacyjne. W tym celu wykorzystamy technikę multiemisji (ang. *multicast*) [2] pozwalającą na wysyłanie jednego strumienia danych od razu do wszystkich klientów zarówno w intranecie jak i Internecie. W przeciwnym wypadku należałoby tworzyć osobny strumień dla każdego klienta, co powodowałoby niepotrzebny ruch. Strumienie wysyłane są poprzez protokół RTP, który zapewnia efektywną transmisję. Drugi element systemu, jakim jest klient odpowiada za:

- odbieranie strumieni danych od serwera,
- przetwarzanie strumieni danych na efekty dźwiękowe i wizualne,
- odczyt slajdów w twardego dysku i nagrywanie dźwięku z mikrofonu,
- zamianę dźwięku i obrazu na strumienie danych RTP,
- wysyłanie strumieni danych do serwera.

Jako medium komunikacyjne możemy wykorzystać Internet lub intranet. Warto zauważyć, że nie ma potrzeby posiadania kamer internetowych. Zamiast tego można przesyłać slajdy, co obniża wymagania sprzętowe i koszty. Użytkownik nie musi widzieć wykładowcy. Wystarczy, że słyszy prowadzącego, widzi omawiane slajdy i ma możliwość zadawania pytań na głos i za pomocą tekstu. Dyskusja pomiędzy wykładowcą i studentem jest słyszana poprzez pozostałych uczestników prezentacji.



Rys. 33.1. Schemat systemu

Budowa systemu przedstawionego na rys. 33.1 wydaje się być prosta, ale są tam elementy wymagające głębokiej analizy i unikalnych rozwiązań. Następne podrozdziały przedstawiają problemy, które autorzy napotkali w trakcie projektowania i implementacji systemu oraz sposoby ich rozwiązania.

3. PROBLEMY ZWIĄZANE Z REALIZACJĄ SYSTEMU

Biorąc pod uwagę niewątpliwe zalety platformy Java (w szczególności przenośność tworzonego oprogramowania) autorzy zdecydowali się wybrać J2SE (ang. *Java 2 Standard Edition*) i JMF (ang. *Java Media Framework*) [1, 7] firmy Sun Microsystems jako platformę programową. Najważniejsze problemy napotkane podczas realizacji systemu to:

- synchronizacja w czasie rzeczywistym głosu i danych,
- transmisja slajdów,
- ograniczenia modelu bezpieczeństwa Javy,
- miksowanie cyfrowego dźwięku pochodzącego od wykładowcy i aktywnych słuchaczy.

3.1. SYNCHRONIZACJA

Charakterystyka problemu

Komercyjne produkty często bazują na protokołach, w których strumienie audio i wideo są wysyłane razem. W takim przypadku nie ma potrzeby synchronizacji. Proponowane rozwiązanie bazuje na protokole RTP i szkielecie JMF, więc synchronizacja jest niezbędna.

Slajdy transmitowane są w oddzielnym strumieniu niż głos. Gdyby nie było synchronizacji, mogłoby się okazać, że studenci słuchają opisu slajdu, który się jeszcze nie pojawił, albo że słyszą opis poprzedniego slajdu. Taki rezultat jest niedopuszczalny.

Rozwiązanie

Ponieważ każdy strumień może pochodzić z innego źródła, ale musi być ciągle zsynchronizowany ze wszystkimi pozostałymi strumieniami autorzy zdecydowali się zastosować dwa poziomy synchronizacji:

- synchronizacja na poziomie transmisji,
- synchronizacja na poziomie prezentacji po stronie klienta.

JMF zapewnia synchronizację zgodnie ze specyfikacją RTP [6]. Każdy strumień powiązany jest z odpowiadającym mu protokołem kontrolnym RTCP, zapewniającym poprawne stemplowanie czasem.

Ta informacja jest również używana do synchronizacji wszystkich strumieni pochodzących od jednego użytkownika identyfikowanych przez tę samą nazwę kanoniczną CNAME. Nawet jeśli strumienie wysłano w osobnych sesjach, każdy będzie identyfikowany przez ten sam identyfikator CNAME.

Z punktu widzenia programisty każdy strumień musi posiadać ten sam CNAME załączony do deskryptora strumienia. Problemem jest synchronizacja strumieni pochodzących od różnych użytkowników. Wiemy jednak, że wszystkie strumienie pochodzące od jednego użytkownika posiadają ten sam identyfikator CNAME i są z definicji zsynchronizowane. Pozostaje więc zapewnić, że wszystkie strumienie wychodzące do wszystkich użytkowników pochodzą od jednego użytkownika. Nadawcą tym jest serwer, który odbiera wszystkie strumienie pochodzące od wykładowcy i wszystkich aktywnych słuchaczy, a następnie przesyła je dalej do wszystkich jako własne. W ten sposób mamy pewność, że strumienie są zsynchronizowane na całym etapie transmisji od nadawcy poprzez serwer, aż po odbiorcę. Dlatego konstrukcja serwera podobna jest do klienta. Serwer jest klientem, który umożliwia synchronizację wszystkich strumieni przez niego przechodzących.

Do synchronizacji strumieni po stronie klienta użyto interfejsu zegara dostarczanego przez szkielet JMF. Oferuje on dwie reprezentacje: czas trwania mediów i znacznik czasu. Czas trwania mediów jest opisywany przez rozpoczęcie i zakończenie prezentacji, podczas gdy znacznik czasu to licznik, który startuje wraz z rozpoczęciem prezentacji. Jeśli dwa lub więcej zegarów używa tego samego znacznika mają one ten sam czas, który można wykorzystać do celów synchronizacji. Wykorzystując interfejs zegara, możemy ustawić ten sam znacznik dla wielu strumieni danych. Zanim to jednak uczynimy, należy określić zegar wzorcowy, względem którego będziemy synchronizować strumienie. Odtwarzacz taki nazywany jest MASTER, a pozostałe SLAVE. Biorąc pod uwagę, że pierwszy strumień jaki odbieramy, jest zawsze strumieniem wykładowcy, wybrano go jako MASTER. Wykorzystując serwer oraz znaczniki czasu, rozwiązano jeden z głównych problemów, jakim była synchronizacja strumieni.

3.2. TRANSMISJA SLAJDÓW

Charakterystyka problemu

Kolejnym problemem okazała się transmisja slajdów. Protokół RTP pozwala na transmisje jedynie dwóch typów strumieni: audio i wideo. Jak zatem włączyć transmisje slajdów JPEG? Należy pamiętać, że rozwiązanie musi zapewniać efektywną pracę nawet przy niskiej prędkości połączenia internetowego.

Rozwiązanie

Użycie innego niż RTP protokołu (np. FTP) dla transmisji pliku uniemożliwiłoby płynne wykłady, ponieważ studenci musieliby pobierać potrzebne pliki korzystając z protokołu FTP w określonych momentach wykładu. Z drugiej strony użycie RTP do ciągłej transmisji slajdów nie jest dobrym rozwiązaniem (odpowiada to transmisji obrazu wideo, więc jest nieprzydatne ze względów wydajnościowych). Wiadomo, że JMF pozwala na od-

czyt i prezentacje JPEG, nie ma jednak mechanizmów pozwalających transmitować sekwencje slajdów, poprzez RTP. Pomimo to JPEG jest poprawnym formatem JMF więc możemy go wysłać strumieniem jako pakiet JPEG/RTP. Rozmiary obrazów JPEG ograniczane są do wielokrotności obszaru 8x8. Ograniczenie to wynika z zastosowania protokołu H.261 [8] do wysyłania strumienia wideo w warstwie transportowej RTP. Jak widać istnieją mechanizmy do transmisji obrazów JPEG poprzez RTP, ale nie są one odpowiednio wydajne. Brak też źródła danych pozwalającego na odczyt obrazów z dysku twardego i pakowanie ich do strumienia RTP.

Autorzy zdecydowali się na użycie mechanizmów rozszerzeń RTP. W JMF istnieją dwa rodzaje klas: *PullBufferStream* i *PullBufferDataSource*, *PushBufferStream* i *PushBufferDataSource*. Pierwszy zwykle stosuje się do implementacji statycznych źródeł danych jak np. pliki, podczas gdy drugi implementuje dynamiczne dane takie jak media czasu rzeczywistego. Klasy *PullBufferStream* i *PullBufferDataSource* mogą być wykorzystane do implementacji usług takich jak wideo na żądanie, gdyż pliki takie są stałe w czasie, ale w naszym przypadku mamy dane dynamiczne. Dlatego też należy użyć klas *PushBufferStream* i *PushBufferDataSource*.

Do ustalenia pozostały jeszcze obsługiwane rozdzielczości i częstotliwość wysyłania slajdów. Jak wcześniej wspomniano z użyciem protokołu H.261 związane są pewne ograniczenia. Możemy wybrać jedną z obsługiwanych rozdzielczości: 176x144, 352x240 i 704x144 i minimalną częstotliwość zmian obrazu 1 klatka na sekundę. Rozdzielczość 176x144 wydaje się być niewystarczająca, gdyż slajdy są niewyraźne i nieczytelne dla użytkownika. Należy więc wybrać pomiędzy 352x240 a 704x576 biorąc pod uwagę zarówno wymaganą jakość obrazu jak i wymóg, aby system był dostępny dla wszystkich użytkowników (nawet tych z łączem internetowym o niskiej przepustowości). Sekwencje slajdów wysyłane co sekundę mogą spowodować zablokowanie łącza i sytuację, kiedy żaden slajd nie będzie dostarczony w całości i nie będzie widoczny na monitorze studenta. To sytuacja bardzo prawdopodobna, ponieważ RTP nie gwarantuje jakości połączenia. Z tego względu nie można być pewnym, że wszystkie pakiety zostaną dostarczone do odbiorcy.

Należy także pamiętać, że wykładowca zmienia slajdy znacznie rzadziej niż co sekundę. W związku z tym powinno się je wysyłać z mniejszą częstotliwością. Przesyłanie slajdów tylko raz nie jest najlepszym rozwiązaniem z dwóch powodów. Po pierwsze student może przyłączyć się do zajęć w dowolnej chwili i powinien widzieć slajd, który aktualnie jest omawiany. Drugą przeszkodą jest zawodność protokołu RTP. Nie można być pewnym, że slajd, który został wysłany poprawnie dotarł do odbiorcy. Dlatego proces wysyłania musi być powtarzany co pewien czas. Pozostaje zatem ustalenie częstotliwości wysyłania slajdów. Parametr ten określa ilość ramek wysyłanych na sekundę w strumieniu RTP. Minimalna wartość to 1 ramka na sekundę. Niestety ta częstotliwość jest zbyt wysoka biorąc pod uwagę nasze potrzeby. Rozwiązaniem problemu jest wysyłanie pakietów JPEG/RTP razem

z pakietami pustymi. Pakiety JPEG ustawiane są jako ramki kluczowe a pozostałe puste pakiety oznaczane są jako nieistotne. W ten sposób plik JPEG wysyłany jest tylko raz na 10 sekund a w pozostałym czasie transmitowane są puste ramki. Jak widać rozwiązanie problemu transmisji slajdów polega na pakowaniu obrazów JPEG w kluczowe ramki strumienia wideo i ustawieniu pozostałych ramek jako nieznaczących w celu zwiększenia efektywności. Puste ramki są niezbędne jedynie do podtrzymania ciągłości strumienia wideo. Takie rozwiązanie zapewnia, że każdy slajd będzie dostarczony, czyni aplikacje wyjątkowo wydajną i rozwiązuje problem łącz internetowych o niskiej przepustowości.

3.3. OGRANICZENIA MODELU BEZPIECZEŃSTWA JAVY

Charakterystyka problemu

Aby ułatwić dostęp do systemu zdecydowano się wykorzystać aplety Javy. Niestety model bezpieczeństwa Javy narzuca następujące ograniczenia:

- aplety nie zezwalają na zapis i odczyt z dysku, wiec nie możemy odczytać slajdów JPEG,
- w Javie nie ma standardowych klas pozwalających na dostęp do zewnętrznych urządzeń multimedialnych, jak np. karta dźwiękowa; mechanizmy te dostarczane są wraz z JMF jako natywne metody, więc nie można ich użyć w aplecie,
- model RTP wymaga do swojego działania sesji i połączenia z serwerem, na co nie pozwala model bezpieczeństwa.

Rozwiązanie

Zaproponowany przez Sun Microsystems model bezpieczeństwa ma wiele zabezpieczeń, które ograniczają funkcjonalność apletów. Z tego powodu pojawiła się idea podpisywania apletów [3, 4, 5]. Zweryfikowany, podpisany aplet udostępnia całą funkcjonalność Javy bez żadnych ograniczeń. Aby wykorzystać tą cechę należy spakować wszystkie klasy i dodać podpisany certyfikat. Certyfikat z kluczem publicznym zdefiniowany jest przez Suna jako cyfrowy podpis pochodzący od określonego dostawcy. Wszystkie systemy bazujące na kluczach publicznych wymagają dostępu do centralnej bazy danych zawierającej wszystkie klucze publiczne. Dostęp do takich danych kontrolują organizacje certyfikujące takie jak VeriSign, Entrust, Thawte, Netscape i Microsoft. Komercyjne certyfikaty są jednak kosztowne a ich ważność ograniczona czasowo. Z tego względu zdecydowano się na inne rozwiązanie – wygenerowano własny certyfikat. Takie rozwiązanie okazało się zupełnie wystarczające, pozwalając na tworzenie sesji i przesyłanie plików na komputer klienta. Użycie RTP do ciągłej transmisji pozwoliło na przesyłanie wielu plików w jednym strumieniu bez potrzeby potwierdzania certyfikatu dla każdego wczytywanego obrazu JPEG. W tym momencie system był gotowy do testów w Internecie.

3.4. MIKSOWANIE DŹWIĘKU

Charakterystyka problemu

Ostatni problem, jaki pozostał do rozwiązania, polegał na dodaniu możliwości transmisji dźwięku w czasie rzeczywistym. Problem ten związany był z miksowaniem cyfrowego dźwięku.

Rozwiązanie

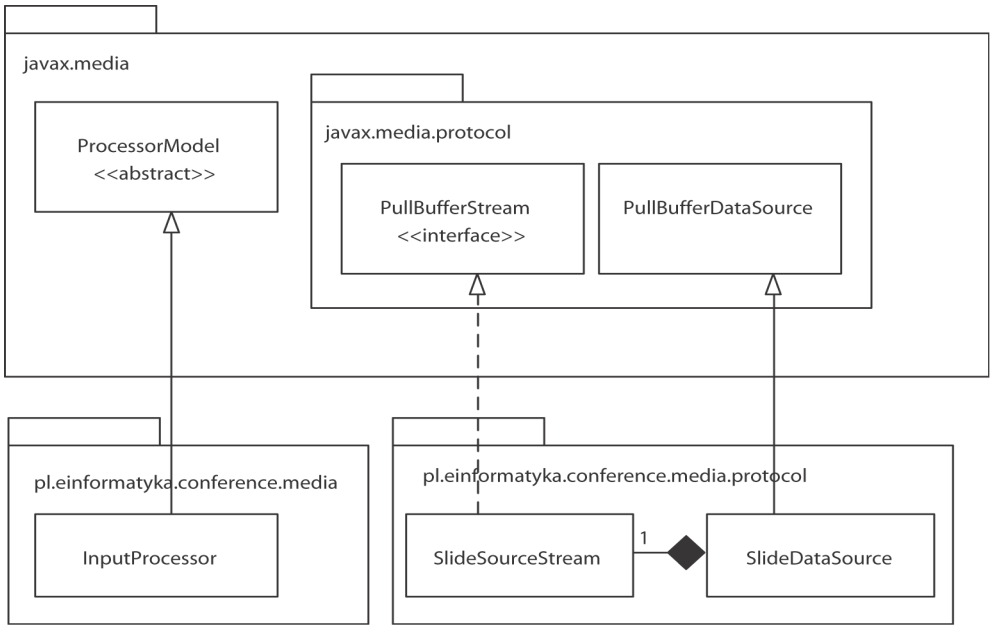
W JMF klasa *MixedDataSource* jest odpowiedzialna za łączenie różnych źródeł danych i stworzenie jednego źródła wyjściowego. Musimy jednak pamiętać, że mamy jedno źródło danych, ale wiele strumieni. Jeśli spróbujemy wysłać taki złożony strumień odbierzemy jedynie pierwszy strumień *MixedDataSource.* Wszystkie pozostałe zostaną stracone, ponieważ klasa *MixedDataSource* łączy jedynie źródła danych, a nie strumienie wejściowe. W związku z tym stworzono jedno źródło danych z wieloma strumieniami. Następnie spróbowano wysłać wszystkie strumienie ze źródła wyjściowego w jednej sesji w taki sposób, aby otrzymać je jako jedno źródło po stronie klienta. Dźwięk okazał się składową wszystkich strumieni, co nie było miłe dla ludzkiego ucha. Dlaczego tak się stało? Wszystkie odebrane strumienie zostały zidentyfikowane jako jeden strumień wejściowy. Dokumentacja JMF mówi, że każdy strumień powinien być transmitowany w osobnej sesji. Postąpiono więc w ten sposób. otrzymując wszystkie strumienie połączone w jedno wejściowe źródło danych. Niestety próba odtworzenia dźwięku zakończyła się zgłoszeniem wyjątku, gdyż odtwarzacz nie obsługiwał połączonych źródeł danych. Brak bowiem klas JMF obsługujących takie źródła danych. Problem wydawał się trudny do rozwiązania bez implementacji klas odpowiedzialnych za odtwarzanie połączonych źródeł danych. Autorzy dostrzegli jednak rozwiązanie natury sprzętowej. Bodaj wszystkie dostępne na rynku karty dźwiękowe obsługują miksowanie cyfrowego dźwięku w czasie rzeczywistym. W związku z tym nie trzeba było implementować osobnej klasy miksującej czy nawet implementować połączonych źródeł danych. Jedyne co należało zrobić, to wysyłać wszystkie strumienie w osobnych sesjach a następnie stworzyć osobny odtwarzacz dla każdego strumienia. Resztą zajęła się karta dźwiękowa. Efekt końcowy okazał się doskonały.

Rozważmy teraz, czy proponowane rozwiązanie jest optymalne. Jakość dźwięku jest idealna, jednak wymaga nieco większej przepustowości sieci. Dwa lub więcej strumieni jest wysyłane zamiast jednego. W idealnym przypadku proces miksowania powinien odbywać się po stronie serwera a nie na karcie dźwiękowej klienta. Implementacja JMF dostarczona przez Suna nie udostępnia takiej możliwości. Musi ona być dostarczona jako rozszerzenie JMF. Nowy procesor sygnału musi być implementacją algorytmu miksowania strumieni audio. Nie jest to proste zadanie, gdyż wymagane są dwa procesory. Pierwszy wykonujący przetwarzanie strumienia zanim, ten zostanie wysłany do *MixedDataSource.* Drugi powinien realizować

końcowe przetwarzanie i łączyć strumienie wejściowe w jeden strumień wynikowy. Testy dowiodły jednak, że rozwiązanie z kartą dźwiękową jest wystarczająco wydajne nawet w polskiej rzeczywistości i nie ma potrzeby implementacji osobnych klas miksujących. Rozwiązanie problemu miksowania dźwięku cyfrowego pozwoliło użytkownikom na prowadzenie dyskusji poprzez Internet w czasie rzeczywistym.

4. PREZENTACJA SYSTEMU

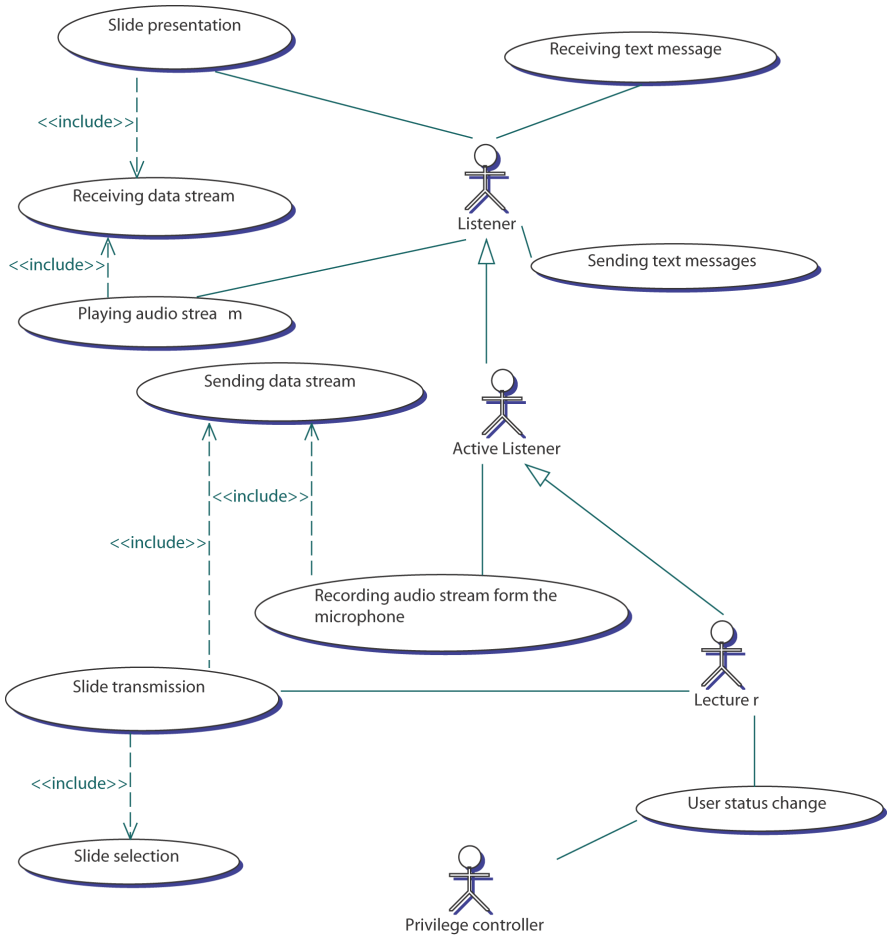
Wszystkie istotne problemy napotkane w trakcie projektowania i implementacji (synchronizacja, transmisja slajdów, ograniczenia modelu bezpieczeństwa, miksowanie cyfrowego dźwięku) zostały rozwiązane i system jest gotowy do pracy. Jak widać na rys. 33.2 jest on rozszerzeniem istniejącego szkieletu JMF na platformie Java.



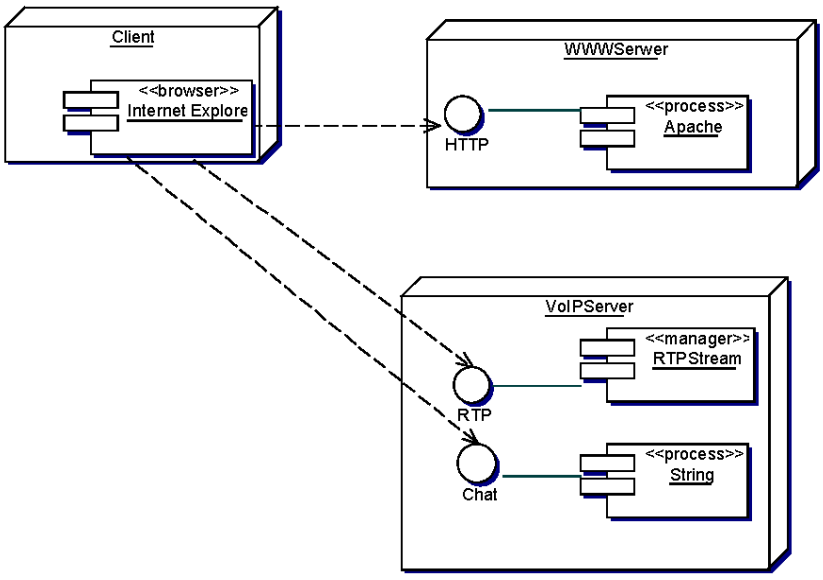
Rys. 33.2. System jako rozszerzenie Java Media Framework

Opisane rozwiązanie udostępnia funkcjonalność pozwalającą na realizację elektronicznego nauczania i konferencji utrzymując wymagania sprzętowe na bardzo niskim poziomie. Użytkownicy nie potrzebują szybkich łącz sieciowych i kamer, aby mieć dostęp do przypadków użycia przedstawionych na rys. 33.3

Posiadanie mikrofonu jest zalecane, ale nie obowiązkowe, gdyż dyskusja może toczyć się także za pomocą konwersacji tekstowych (ang. *chat*). Funkcjonalność ta jest również udostępniana przez system. Jedynym wymaganiem jest komputer z kartą dźwiękową i dostępem do sieci Internet.



Rys. 33.3. Diagram przypadków użycia



Rys. 33.4. Diagram rozmieszczenia

Diagram rozmieszczenia na rys. 33.4. przedstawia elementy systemu z perspektywy konfiguracji uruchomieniowej. Na diagramie możemy wyróżnić następujące węzły: Klient, Serwer głosowy, Serwer WWW. Klient łączy się z serwerem Apache poprzez protokół HTTP. Najważniejszym elementem jest połączenie Klient-Serwer głosowy. Węzły te połączone są poprzez RTP do transmisji głosu i slajdów oraz osobno do przesyłania wiadomości tekstowych (chat).

5. PODSUMOWANIE

Została przeprowadzona eksperymentalna weryfikacja wydajności systemu. Szczegółowa analiza i prezentacja rezultatów testów wydajnościowych systemu wykracza poza ramy tego rozdziału. Warto jednak podkreślić, że testy przeprowadzono zarówno dla łącz ISDN jak i z wykorzystaniem modemu analogowego. Serwer pracował na komputerze Politechniki Wrocławskiej, a klient łączył się poprzez linię abonencką sieci Dialog. Odległość pomiędzy systemami wynosiła 10 hopów. Pomiary wykonywano dla różnej rozdzielczości slajdów i przy wyłączonej ich transmisji. Dowiodły one, że system doskonale sprawdza się podczas transmisji głosu w czasie rzeczywistym. W przypadku transmisji strumienia głosowego bez slajdów nawet modem analogowy 56K okazał się wystarczający. Jakość dźwięku była bardzo

dobra, a opóźnienie strumienia nie przekraczało 3-4 sekund. Wysyłanie slajdów razem ze strumieniem dźwiękowym znacznie zwiększyło wymaganą przepustowość sieci. Nie było to problemem dla 128K ISDN czy sieci uczelnianej, ale modem 56K mógł mieć trudności. Małe obrazy docierały do odbiorcy po 3-4 sekundach. Duże pliki transmitowane przy połączeniu o niskiej przepustowości wymagały do 30 sekund na dotarcie do celu i jakość dźwięku obniżała się. Takie zachowanie nie było obserwowane przy szybszych połączeniach. Można zatem stwierdzić, że dzięki zaprezentowanym rozwiązaniom system oferuje możliwość praktycznego wykorzystania zaawansowanych funkcji komunikacyjnych przy minimalnych nakładach na infrastrukturę sprzętową i komunikacyjną. W szczególności pozwala korzystać z systemu użytkownikom łącz o ograniczonej przepustowości.

LITERATURA DO ROZDZIAŁU

[1] DeCarmo L.: Core Java Media Framework, Prentice Hall PTR, 1999.

[2] Deering S.: „Host Extensions for IP Multicasting", RFC1112, Network Working Group, September 1989.

[3] Microsoft, HOWTO: Make Your Java Code Trusted in Internet Explorer, http://support.microsoft.com, 2000.

[4] Netscape, Netscape Security Documents, http://wp.netscape.com/eng/ security, 2000.

[5] Oaks S.: Java Security, 2nd Edition, O'Reilly, May 2001.

[6] Schulzrinne H., Casner S., Frederick R., Jacobson V.: RTP: A Transport Protocol for Real-Time Applications, RFC1889,NetworkWorkingGroup, January 1986.

[7] Sun Microsystems, JMF 2.0 Beta release notes, http://java.sun.com/products/java- media/jmf/2.1.1/specdownload.html.

[8] Turletti T., Huitema C.: RTP Payload Format for H.261 Video Streams, http://www.faqs.org/rfcs/rfc2032.html, October 1986.