УДК 683.1

ПРОГРАМНА ПІДТРИМКА СИНХРОНІЗАЦІЇ МУЛЬТИМЕДІЙНИХ МАТЕРІАЛІВ ЛЕКЦІЇ

А.М. Глибовець, Д.С. Тихоновський

Національний університет "Києво-Могилянська академія",
04070, Київ, вул.Сковороди, 2.
Тел.: (044) 463 6985, andriy@glybovets.com.ua

Розглядаються можливості використання технологій мультимедіа в електронному навчанні. Особливу увагу звернуто до проблеми синхронізації мультимедійних метеріалів різних форматів в електронних лекціях.

This speech covers the possibilities for the application of multimedia technologies to e-learning. A special attention is paied to the problem of how to synchronize multiformat media content of electronic lectures. Application support of the synchronization is also considered.

Вступ

Основним елементом покращення програмних систем підтримки електронного навчання є якісне використання сучасних технологій в області мультимедіа [1 – 4].

Під мульмедіа зазвичай розуміють: текст, графіку, аудіо, відео та їхні комбінації в електронному вигляді. Сьогодні майже кожний персональний комп'ютер здатний відтворювати мультимедійні презентації для дозвілля, роботи або освіти. В англомовному світі з'явився навіть спеціальний термін - "edutainment" [4], який можна було б перекласти як "розвага заради освіти". Під ним розуміють сукупність інтерактивних мультимедійних технологій, які використовуються в електронній освіті.

Передача мультимедійних документів великого розміру і досі є проблематичною, враховуючи обмеженість пропускної здатності комунікаційних каналів та якість передачі даних. Хоча впровадження високошвидкісних оптоволоконних (optical fiber) та бездротових (wireless) мереж покращило ці показники. Але найбільш істотних змін процес передачі аудіо та відео через Інтернет зазнав протягом останніх років, із появою поняття "потокового мультимедіа" (streaming multimedia) – це засіб передачі великих файлів в "потоці" (stream). Різниця полягає в тому, що потокове аудіо/відео може починати програватись вже після отримання перших пакетів даних від постачальника, і не треба чекати завантаження всього файлу, на відміну від звичайного методу скачування.

Ключовим тут є поняття формату даних, адже він по-перше відповідає за компресію мультимедійних файлів, тобто зменшення розміру за умов максимального збереження якості; а по-друге дозволяє передавати великий файл у вигляді неперервного потоку бітів, який розпізнається програмним забезпеченням на боці клієнта (отримувача потоку) та програється з мінімальною затримкою у кілька секунд. Поки програється одна частина потокового мультимедіа, наступні вже підвантажуються і відтворюються "безшовним" способом (seamlessly).

Таким чином, потокове мультимедіа дозволяє перегляд та прослуховування значних обсягів даних, навіть при відносно повільному модемному підключенні до Інтернет.

Програмна підтримка синхронізації мультимедійних матеріалів лекції

Наразі, електронне навчання істотно поступається очним формам за показниками якості навчання та представленню матеріалів. Однією з найважливіших причин є те, що в умовах електронного навчання важко забезпечити контакт лектора з аудиторією і передати авторське бачення наданих матеріалів.

Шляхом розв'язку цих проблем може бути використання систем підтримки мультимедійних електронних лекцій, в яких аудіо та відеозапис виступу лектора узгоджувався б із синхронізованим представленням демонстраційних матеріалів (документів, схем, малюнків тощо). Але тут постає складність організаційного характеру: якщо лектор має оперувати електронними матеріалами лекції під час виступу перед аудиторією, то всі підготовчі роботи мають бути виконані ним заздалегідь. В ідеальному варіанті, слід забезпечити викладача нескладним але ефективним інструментарієм для внесення/завантаження в систему всіх демонстраційних матеріалів за найкоротший час. Разом із тим, аудиторія, тобто всі користувачі локальної мережі або мережі Інтернет, потребують якісного представлення навчальних матеріалів: як електронних документів, так і аудіовізуального супроводження, коректно синхронізованих між собою, із необхідним інструментарієм для навігації та зворотного зв'язку із сервером – тобто постачальником електронної лекції.

Поширеною практикою стає супроводження статичних навчальних матеріалів електронної лекції відеозаписом промови та коментарів викладача, що посилює інтенсивність та зрозумілість подачі матеріалів. Зазначимо, що тут є певні обмеження стосовно якості подання відеозапису або трансляції, бо від цього залежить розмір відеофайла. Стандартним є подання відео розміром 320x240 пікселів, із частотою кадрів 30 fps.

Можна навести багато вдалих прикладів застосування мультимедіа у електронних лекціях [5]. Зазначені системи підтримують підготовку, запис, пряму трансляцію та відтворення електронних лекцій, які складаються з двох основних частин: демонстраційної (статичної) – тобто всіх електронних документів, які пропонуються на розгляд аудиторії; мультимедійної – аудіовізуальний потік, що передає промову викладача, його вказівки та коментарі.

Визначення форматів та джерел лекційних матеріалів

Статичні електронні документи

Статичні демонстраційні матеріали поки не були чітко визначеними. Ними можуть бути різні формати подання тексту, графіки, схем, формул і т.п. Найвірогідніше це – електронні документи, які можуть бути відкриті веб-браузером, можливо, за допомогою встановлених плагінів. На цьому варто зробити акцент, бо це цілком задовольняє одну з необхідних вимог до електронних лекцій, а саме: відтворення лекції безпосередньо у веб-браузері – тонкому клієнті. Ця обставина робить лекцію доступною для перегляду через мережеве з'єднання без необхідності встановлення додаткового програмного забезпечення. Електронні документи можуть мати різний формат [6, 7] з використанням спеціалізованих додаткових плагінів [8, 9]. Наприклад, плагіни для перегляду DjVu наявні для всіх основних браузерів, а для PDF достатньо встановити Acrobat Reader.

Отже, існує широкий вибір форматів демонстраційних даних лекції, і всіх їх можна переглядати у вікні браузера. Що наводить на думку: чому б не ідентифікувати всі ці ресурси просто за їхніми URL (Uniform Resource Locator). Це найпростіший і найефективніший спосіб, який, до речі, і використовується в більшості існуючих систем. Таким чином, всі демонстраційні матеріали однозначно визначаються за своїми URL-адресами, причому знаходитись вони можуть як в локальній папці, так і на віддаленому інтернет-сервері. Це додатково дає можливість лектору посилатись на сторонні ресурси, не порушуючи чиїхось авторських прав несанкціонованим копіюванням.

Для підготовки таких лекційних матеріалів від викладача вимагається зібрати всі потрібні електронні документи локально та зберегти посилання на віддалені ресурси, впорядкувати їх за логічною послідовністю. Далі, можна приступати до запису аудіо/відео презентації, після редагування якої лектор зможе в певному зручному інтерфейсі визначити більш менш точні часові мітки для підвантаження кожного демонстраційного документа. Зберегти співвідношення типу “часова мітка – URL” не являє складності. Це можна реалізувати як параметричний файл (найпростіше), документ XML, записи в таблиці бази даних певної СУБД.

Останній варіант можливий у випадку більш повнофункціональної системи електронної освіти, коли СУБД вже існує, зв'язок із нею встановлюється під час кожного сеансу роботи користувачів, і тоді можна говорити про централізоване зберігання всіх даних системи. Але якщо ціллю є лише забезпечення синхронізації матеріалів лекції, то використання документів XML – найоптимальніше. Звичайно, можна обмежитись і параметричним файлом, який буде складатись із рядків типу “timestamp=url\n”, але це ускладнює ситуацію, якщо доведеться зберігати ще якісь описові та метадані стосовно матеріалів лекції, а таке цілком ймовірно.

Ситуація ускладнюється, у випадку синхронного подання лекції. Тоді, викладач повинен мати можливість додавати кожний новий документ прямо під час трансляції. Але якщо він вже підготував і відсортував за послідовністю всі документи (посилання), то достатньо надати йому можливість бачити цей перелік перед очима і лише клікати за необхідним лінком. Цей перелік може бути згенерований автоматично за існуючим XML-документом, який під час трансляції/запису лекції буде доповнюватись відповідними часовими мітками.

Потокові аудіовізуальні матеріали

Друга складова частина лекції – це аудіо або відеопотік, який підвантажується із певного джерела, за визначеним протоколом, у певному визначеному форматі. Відтворення потоку здійснюється відповідним програвачем-плагіном у браузері.

Відео може передаватись у прямій трансляції, або воно було вже записаним та закодованим для подальшої обробки та перегляду. У контексті електронних лекцій актуальні обидва варіанти, адже лекція може відбуватись як у прямій трансляції, так і багаторазово відтворюватись із запису – video on demand (VOD). Передача відео у прямій трансляції одразу створює необхідність забезпечення кодування потоку із записуючого пристрою у певний визначений формат. Це досить складна програмна складова будь-якої системи, тому тут найчастіше застосовують готове спеціалізоване програмне забезпечення, наприклад, Microsoft Windows Media Encoder, Helix від RealNetworks або Flash Media Server від Adobe.

Найчастіше, відео представлено в одному із наступних форматів: AVI (Audio Video Interleave, Microsoft) [10]; MPEG (Moving Picture Experts Group); MOV (QuickTime Movie, Adobe) [11]; SWF, FLV (Adobe Flash, ex-Macromedia Flash) [12]; RM (RealNetworks RealMedia) [13] та інші поширенні формати.

Способи подання відеоматеріалів

Завантаження файлу з сервера

Це найбільш прямий і простий спосіб передачі відео через Інтернет. Скачування (download) відеозапису нічим не відрізняється від скачування будь-якого іншого формату даних, хіба що значним розміром файла. В цих цілях використовують протокол TCP на транспортному рівні, та FTP, HTTP на більш високих рівнях.

Очевидний недолік – необхідність закачування всього вмісту великого файла перш ніж його можна буде переглянути, а це суттєва затримка в часі. До того ж, потрібне місце для зберігання файла на локальній машині. Все це створює незручності при застосуванні в електронному навчанні.

Потокове відео

Цей спосіб вирішує проблеми, пов'язані із закачуванням великих відео файлів, а також надає інші переваги. Ідея полягає у тому, щоб розбити великий медіа файл на велику кількість часток незначного розміру, які можна передавати один за одним клієнтові, на боці якого здійснюється декодування потоку та відтворення його програвачем. Передавання відео контенту в потоці дозволяє вирішити проблему синхронізації передачі даних і їх відтворення програвачем.

Недоліки в цьому випадку пов'язані з тим, що UDP (User Datagram Protocol) та його надбудови, на відміну від надійного TCP, не гарантують доставку всіх пакетів, оскільки у випадку їхньої втрати, повторно вони надсилатись не будуть. Інша проблема – фаірволи часто блокують передачу даних за протоколами побудованими на UDP.

Розбиття відео файла на послідовність фрагментів

Це компроміс між першим і другим варіантами. Якщо на стороні сервера розбити великий файл на N часток незначного розміру Δ, то можна уникнути значних затримок при відтворенні на стороні клієнта, навіть при передачі по HTTP. Можливі проблеми тут – це втрата пакетів при передачі і пов'язані із цим затримки. Для того щоб цей метод можна було застосовувати програмне забезпечення має чітко відповідати вищезазначеним умовам.

Комунікація між постачальником та отримувачем мультимедіа

Існують різні способи комунікації та подання потокового мультимедіа.

Наприклад, передача даних може здійснюватись як point-to-point (“один-до-одного”), multicast (“один-до-багатьох”) або broadcast (“один-до-всіх”). Джерелом відеопотоку може бути як вже записаний файл, так і пряма трансляція відеозапису у певному форматі кодування. Специфічні властивості конкретного застосування для передачі відеопотоку визначають архітектуру системи. Докладніше про це можна знайти в [14 – 16].

У контексті досліджуваної проблематики можна сказати, що multicast можна використовувати, коли певна кількість користувачів має синхронно слухати/переглядати одну лекцію. Наприклад, це може бути подача лекції у чітко визначений час цілій групі або аудиторії студентів, де кожен працює за окремим ПК.

Та найперспективніший, на нашу думку, є спосіб – P2P (peer-to-peer, вузол-до-вузла). Протоколи цього сімейства розглядають кожен вузол (node) мережі як клієнтом, так і сервером одночасно. Певна кількість найбільш надійних та потужних із них вважається “супервузлами” (supernodes). Кожен звичайний вузол отримує доступ до мережі через певний супервузол, таким чином здійснюється маршрутизація. За політиками протоколів P2P, дані передаються від клієнтів, що вже їх отримали або зберігають до тих, хто надсилає запит на їх завантаження.

Це вирішує проблему “пляшкової горлянки” (bottleneck), якою є звичайний сервер, що має відповідати на запити великої кількості користувачів одночасно. Варто лише згадати популярний нині Skype. Звичайним користувачам Skype відомий у першу завдяки своїй феноменальній швидкості та якості передачі і відтворення відео та звуку [17].

До недоліків такої комунікації можна віднести складність технічної організації P2P-мережі, її функціонування, якості обслуговування, а також питання спільних і приватних даних, що недостатньо врегульоване законодавством.

Вимоги до системи підтримки електронних лекцій

Виділимо основні вимоги до системи підготовки і трансляції мультимедійних лекцій:

- підготовка та запис лекційних матеріалів *(*необхідна можливість попередньої підготовки лекційних матеріалів; наявність визначених структур даних, на базі яких можна було б створити типову послідовність демонстраційних матеріалів; можливість створення власних структур матеріалів, якщо відсутня пристосована типова структура даних; можливість запису лекції в реальних умовах (в аудиторії, конференц-залі тощо), безпосередньо під час виступу перед аудиторією; засоби підтримки прямої трансляції лекції разом з демонстраційними матеріалами через локальну мережу або інтернет; засоби збереження лекції разом з демонстраційними матеріалами для наступного перегляду);
- відтворення та трансляція (синхронне подання лекції – можливість перегляду прямої трансляції лекції разом з демонстраційними матеріалами через локальну мережу або інтернет; асинхронне подання лекції - можливість перегляду збереженої лекції разом з демонстраційними матеріалами; підтримка „змішаного режиму” перегляду лекції: якщо лекція іде в прямому ефірі, користувач повинен мати можливість дивитись як трансляцію, так і передивлятись вже збережений матеріал; додатково можна виділити можливість запису і перенесення лекцій на дискових накопичувачах (CD, DVD, жорсткий диск, флеш-пам’ять тощо);
- засоби синхронізації мультимедійних матеріалів лекції (лектор повинен мати можливість мінімальним зусиллям переключатися між демонстраційними матеріалами, а часові мітки цих дій мають зберігатися; треба також надати можливість подачі нових матеріалів за таймером, попередньо визначеним при підготовці лекції;

забезпечити користувача засобами навігації як за демонстраційними матеріалами, так і за мультимедійним вмістом лекції із певною логікою їхньої синхронізації; мінімізація затримок при підвантаженні аудіовідеопотоку).

Програмне забезпечення підтримки електронної лекції

Тепер, маючі визначені загальні вимоги до програмного забезпечення підтримки синхронізації мультимедійних матеріалів лекції, підберемо необхідне програмне забезпечення. Будемо враховувати в першу чергу такі показники як: можливість використання системи якомога більшою кількістю користувачів різних апаратно-програмних платформ, відсутність або мінімальна кількість додаткового необхідного ПЗ, мінімізація вартості розробки та впровадження, простота та зручність у використанні, можливість нарощування функціональності.

Веб-браузер як тонкий клієнт

Від самого початку мова йшла про дистанційне навчання, яке б здійснювалось через доступ до організованих інтернет-ресурсів. Всі користувачі мережі Інтернет мають той чи інший веб-браузер, який ще часто називають „тонким клієнтом". Стандартні засоби та можливості останніх версій Internet Explorer, Mozilla, Netscape, Opera та ін. дозволяють нам уникнути необхідності розробляти якесь власне середовище для перегляду електронних лекцій.

Застосувавши розбиття робочого вікна на фрейми, ми отримаємо необхідну кількість функціонально-незалежних частин, кожна з яких працює як окремий об'єкт document. В кожен фрейм ми можемо завантажувати будь-який вміст, і це не відобразиться на інших. Таким чином, поки в нас в одному фреймі буде працювати об'єкт (<OBJECT>) медіа-програвача, ми можемо надати користувачеві можливість передивлятись демонстраційні матеріали в фреймі основної частини. Будь-які дії користувача будуть оброблятись спеціальними функціями JavaScript, на яких і буде побудована синхронізація матеріалів на боці клієнта.

Вибір медіа-програвача залежатиме від вибору формату запису аудіовідеопотоку. Але заздалегіть відомо, що всі стандартні плагіни-програвачі (Windows Media Player, Real Player, QuickTime Player, Flash Player etc.) надають засоби керування ними через методи та атрибути об'єктної моделі.

Підтримка запису та прямої трансляція мультимедіа

У контексті електронної лекції, було б зручно здійснювати запис і трансляцію мультимедіа зі звичайної веб-камери, що посилає потік на певний порт – стандартно це 1121, але може бути будь-який інший. Таким чином доступ до потоку буде здійснюватись за адресою типу ip:port або DNS:port. Якщо комп'ютер не має ні статичної ip-адреси, ні DNS-ім'я, то цю проблему можна вирішити за допомогою таких безкоштовних сервісів як *no-ip.com* або *dyndns.org*.

У випадку прямої трансляції тепер достатньо зазначити всередині об'єкта медіа-програвача подібний параметр: <param name ="URL" value ="< address>:1121" > де <address> відповідає або статичному ip, або DNS.

Із записом мультимедіа ситуація складніше: тут необхідно використовувати додаткове спеціалізоване програмне забезпечення. На Windows-платформі це може бути Media Encoder, як це наприклад, реалізовано в системі GWOTS [18]. Windows Media Encoder надає широкі можливості для запису, відкритої трансляції, компресії та обробки мультимедіа. Так наприклад, при трансляції на обраний порт, відео потік вже іде закодований у форматі AVI, що дозволяє його перегляд будь-яким сучасним програвачем.

Окрім цього, інсталяція WME надає програмістові доступ до потужного API, і це дозволяє здійснювати власні розробки. Той самий GWOTS був написаний мовою Delphi із використанням Media Encoder API.

Існує незалежна від операційної системи альтернатива – Java Media Framework (JMF) [19]. Цей фреймворк (поточна версія 2.1.1) являє собою API, що дозволяє девелоперам працювати із часово-залежним (time-based) мультимедіа, використовувати його в своїх програмах та розширювати функціональність JMF за рахунок дописування власних плагінів для підтримки специфічних задач. Зазначимо, що JMF надає широкі можливості для роботи з відео у форматі AVI – це запис, трансляція, обробка і компресія. Перевага відео в форматі AVI в тому, що він декодується більшістю програвачів, в тому числі Windows Media Player, який встановлений майже в усіх користувачів ОС Windows.

У випадку прямої трансляції лекції, Java Media Framework дозволяє здійснювати передачу як за протоколу TCP (або його надбудові – HTTP), так і за RTP (що є надбудовою над UDP). Це дозволяє говорити про повноцінну передачу потокового відео через Інтернет.

Синхронізація аудіовідеопотоку та документів лекції

Як було з'ясовано попередньо, для синхронізації мультимедійних матеріалів лекції, потрібно в першу чергу встановити взаємно однозначну відповідність між часовим мітками аудіовізуальної частини та URL-адресами тих демонстраційних матеріалів, які мають бути показані в цей момент часу. Здійснювати відлік за прямою часу (timeline) можна окремо для кожної лекції, починаючи з нульової позначки. Тоді для кожного слайду лекції буде встановлене значення в секундах, яке зазначає на те, якому моменту від початку лекції,

тобто її аудіовізуальної частини, буде відповідати цей слайд. Під “слайдом” розуміємо звичайно визначені статичні електронні документи.

Встановлення цих відповідностей можна покласти на автора лекції – викладача, або того, хто буде здійснювати підготовку лекційних матеріалів і проводити запис лекції. Послідовність може бути більш менш довільною: якщо лекція має транслюватись наживо, то підготовка має бути здійснена заздалегідь; якщо створюється асинхронна лекція для подальшого перегляду, то додати посилання на демонстраційні матеріали та їхні часові мітки можна будь-коли, поки триває робота над підготовкою контенту.

Засоби синхронізації на стороні сервера

Домовимось, що на виході, тобто після закінчення всіх підготовчих робіт, у визначеному місці на сервері має існувати xml-документ приблизно такого вигляду:

```
<?xml version="1.0" encoding="UTF-8"?>
<lecture>
        <slide>
                <title>Документ #1 – заголовок</title>
                <author>Автор документу #1</author>
                <url>Локальна або віддалена ur l– адреса ресурсу</url>
                <timestamp>Кількість секунд від початку лекції</timestamp>
        </slide>
        <slide>
                <title>Документ #2 – заголовок</title>
                <author>Автор документу #2</author>
                <url>Локальна або віддалена url-адреса ресурсу</url>
                <timestamp>Кількість секунд від початку лекції</timestamp>
        </slide>
...
</lecture>
```

Як бачимо, в цій простій структурі закладено все, що може знадобитись для відтворення лекції з сервера на запит користувачів. У кожному вузлі *<slide></slide>* містяться теги зі значеннями: 1) заголовка, який можна показувати користувачеві; 2) автора документа (вибірково, але бажано); 3) адреса ресурсу, тобто документа або у локальній папці на сервері, або на віддаленому сервері в мережі; 4) часова мітка відносно початку лекції, коли настане цей момент у програванні мультимедіа – документ має стати доступним для перегляду.

До речі, існує принаймні три варіанти, як представляти користувачеві кожен наступний слайд із набуттям або переходом за його часову мітку:

- не надавати інформації про ті слайди, час яких ще не настав, а підвантажувати посилання на них під час лекції – це може бути корисним при прямій трансляції, коли небажано, щоб студент “забігав” поперед викладача;
- примусово підвантажувати контент документа в основну частину робочого вікна – це досить грубий спосіб;
- одразу із початком лекції зробити доступними всі слайди і посилання на них в навігаційній панелі, а в кожен момент часу не переводити примусово користувача до перегляду документа, а лише підсвічувати посилання на поточний слайд – це найкращий варіант у випадку асинхронного подання лекції.

Питання запису і трансляції мультимедійної частини лекції було висвітлено в попередній частині. До цього лише необхідно додати можливість створення вищеописаного xml-документа – каталога посилань на демонстраційні матеріали лекції. Для цього існує безліч способів, і все залежить від того, в якому середовищі викладач буде проводити запис лекції. Це середовище-систему, за наявності вихідного коду, можна доповнити невеликим модулем, який би створював і редагував ці xml-документи, у відповідності до даних викладача, що вводяться. Це можна здійснити будь-якою високорівневою мовою програмування.

У крайньому випадку – це може бути веб-інтерфейс із HTML-формою. У нашій розробці запрограмовано серверну частину мовою PHP, тому довелося лише створити нескладний php-модуль, який би породжував, парсив та редагував вміст xml-документів. Наголосимо, що це може бути будь-яка інша технологія, яка лежить в основі серверної частини ПЗ. Тим більше, що для всіх поширених мов програмування існують готові рішення – API для створення, читання та редагування документів XML. У нашому випадку – це php-extensions: XMLWriter, XMLReader та XML Parser. Хоча при бажанні можна написати і власні бібліотеки функцій для роботи з XML, адже це лише plain text із добре визначеною структурою.

Отже, для створення та редагування вмісту лекції у форматі XML, ми використовуємо форми в HTML, з яких методом POST дані передавались на серверний скрипт-обробник подій. Далі, отримані дані з масиву $_POST потрібно було записати у xml.

Наведемо невеликий фрагмент коду, який створює файл з визначеною раніше в цьому розділі структурою:

```
<?php
$filename = '/lecture/contents.xml'; // шлях до локального xml-файлу для запису
$xw = new xmlWriter();
$xw->openURI($filename);
$xw->startDocument('1.0','UTF-8');
$xw->startElement ('lecture');
$xw->startElement('slide');

$xw->writeElement ('title', $_POST['title']);
$xw->writeElement ('author', $_POST['author']);
$xw->writeElement ('url', $_POST['url']);
$xw->writeElement ('timestamp', $_POST['timestamp']);

$xw->endElement();
$xw->endElement();
?>
```

Можливий запис кількох міток одночасно, або їх редагування тими ж самими засобами. Тільки у разі редагування весь вміст доведеться попередньо прочитати за допомогою xmlReader, і опрацьовувати далі згідно даним отриманим від клієнта по POST, але труднощів це не представляє.

Маючи такий документ із списком лекційних матеріалів, можна передавати клієнтові на запит цей вміст або весь одразу, або за пунктами відповідно до їхньої послідовності та поточного часу від початку лекції – це залежить від бізнес-логіки системи. Та обидва варіанти мало чим відрізняються, відносно побудови серверної частини ПЗ. Необхідно лише створити модуль-обробник подій, тобто запитів користувача.

Якщо це випадок разового завантаження всього вмісту лекції, то обробник спрацює єдиний раз – розпарсить весь xml-документ та передасть його вміст у вигляді готового HTML (який буде згенерований на сервері) – це буде вже готовий вміст панелі навігації за матеріалами лекції.

Якщо ж посилання на статичні ресурси мають підвантажуватись динамічно під час лекції (випадок прямої трансляції), то кожне посилання на наступний слайд можна "підтягувати" викликом за *XMLHttpRequest* із клієнтської частини ПЗ, без перезавантаження всієї сторінки. На сервері буде здійснено парсинг xml, і залежно від поточного часу лекції (цей параметр потрібно передавати від клієнта із запитом) буде переданий новий фрагмент HTML. На стороні клієнта, функції JavaScript отримають цей доданок і внесуть його до розмітки сторінки (фрейму) за допомогою методу *appendChild*, яким можна динамічно змінювати вміст HTML-сторінки за специфікацією деревоподібної об'єктної моделі документа DOM.

Після визначення структури xml-документа, що буде містити необхідні для синхронізації дані, і програмування обробників запитів на боці сервера, можна перейти до клієнтської частини, яка буде достатньо динамічною, адже синхронізація матеріалів лекції буде багато в чому залежати від того, що відбувається саме на стороні клієнта.

Розробка клієнтської частини на XHTML & JavaScript

Можна підсумувати, що користувач має бути забезпечений засобами навігації як за мультимедійною частиною лекції, так і за наявних демонстраційних матеріалах. При цьому будь-які такі дії мають викликати за собою функцію JavaScript для перевірки синхронізації подачі матеріалів на боці клієнта.

Як було зазначено раніше, мультимедійна складова частина лекції може передаватись за різними протоколам (HTTP або RTP/RTSP), може бути представлена в різних форматах даних, та в різному їх поданні – цілим файлом, потоковим медіа (streaming media) або розбиттям на певну кількість файлів незначного розміру. Залежно від формату даних, тобто кодування аудіо відео потоку, буде обиратись медіа програвач – об'єкт нашої клієнтської частини.

Вибір медіа програвача не є принциповим для забезпечення синхронізації подання мультимедійних матеріалів лекції. Нам достатньо лише знати, що всі вони, як об'єкти веб-сторінки і частина її DOM, мають певні необхідні атрибути і методи, до яких можна звертатись із JavaScript. Виключенням є лише певні реалізації Flash Player, але підходящу версію можна знайти.

Для нашої реалізації ми обрали Windows Media Player: він доступний, поширений і добре кооперується із різними браузерами. Необхідну інформацію за атрибутами і методами об'єкта медіапрогравача, можна прочитати в [20].

Вставити об'єкт WMP у веб-сторінку можна таким фрагментом коду:

```
<OBJECT id="contentPlayer" width="320" height="240"
        style="position:absolute; left:0;top:0;"
        CLASSID="CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6"
```

```
        type="application/x-oleobject">
        <PARAM NAME="URL" VALUE="<media file URL>" />
        <PARAM NAME="SendPlayStateChangeEvents" VALUE="True" />
        <PARAM NAME="AutoStart" VALUE="True" />
        <PARAM name="uiMode" value="none" />
        <PARAM name="enableContextMenu" value="false" />
</OBJECT>
```

Параметр *uiMode*, визначає які засоби керування програвачем будуть доступними користувачеві. Встановлюючи значення в '*none*', ми приховуємо всі стандартні кнопки, і замість них пропонуємо власні, які будуть викликати необхідні функції-обробники подій. Основні – це Play, Pause, Stop. Параметр *enableContextMenu* із значенням '*false*' також робить недоступним контекстне меню, яке зазвичай викликається натисненням правої кнопки миші на об'єкті.

Кнопки керування програвачем задаються таким чином:

```
<input type='button' name='playButton' value='Play' onClick='playHandler()'>
<input type='button' name='pauseButton' value='Pause' onClick='pauseHandler()'>
<input type='button' name='stopButton' value='Stop' onClick='stopHandler()'>
```

Базовими обробниками цих подій будуть такі функції:

```
<script type="text/javascript" language="JavaScript">
<!--
function playHandler() {
        var wmp = document.getElementById('contentPlayer');
        if (wmp.controls.isAvailable('Play'))
                wmp.controls.play();
}
function pauseHandler() {
        var wmp = document.getElementById('contentPlayer');
        if (wmp.controls.isAvailable('Pause'))
                wmp.controls.pause();
}
function stopHandler() {
        var wmp = document.getElementById('contentPlayer');
        if (wmp.controls.isAvailable('Stop'))
                wmp.controls.stop();
}
-->
</script>
```

До атрибутів об'єкта медіапрогравача, якими потрібно буде оперувати відносяться:

- *URL* – це файл, який буде програватись. Встановлюється один раз при завантаженні лекції, і, якщо це випадок розбиття відео на частки – то значення буде змінюватись динамічно (передаватись з сервера);
- *currentPosition* – це найважливіший параметр для цілей синхронізації, він показує поточну часову мітку медіа, що програється, в секундах.

Тому, потрібна певна callback функція, яка б запускалась із зазначеною періодичністю і перевіряла значення *contentPlayer.currentPosition.* Для кожного слайду лекції визначена відповідна йому часова мітка – це ми і вибираємо за основу синхронізації матеріалів лекції на стороні клієнта. За кожним поточним значенням *contentPlayer.currentPosition* буде однозначно визначатися слайд, який має бути активним у цей момент часу. Що буде відбуватись далі із цим слайдом – питання бізнес логіки: або він буде примусово завантажуватись в основну частину користувацького вікна, або буде ставати доступним посилання на нього, або посилання просто має виділятись кольором або напівжирним шрифтом (на чому ми і зупиняємося).

Якщо все-таки буде потрібне підвантаження нового посилання за часовою міткою, то можна використовувати *XMLHttpRequest,* щоб не відбувалося перезавантаження сторінки. Нагадаю, що цей API застосовується для того, щоб встановлювати незалежний канал зв'язку за протоколом HTTP між клієнтським браузером і сервером. Цей метод набув визнання завдяки поширенню методології розробки Ajax.

Останнє питання: як організувати періодичний виклик функції, яка буде реалізовувати цю бізнес логіку, наприклад кожні 5 або 10 секунд. Це може бути виклик за таймером – тобто динамічно змінюваним проміжком часу, який буде визначатись, як різниця в секундах між часовою міткою поточного слайду і наступного.

Для цього в JavaScript є корисна функція *setTimeout(expression, msec)*, де замість *expression* ми зазначаємо на потрібну функцію, а *msec* дорівнює інтервалу в мілісекундах до здійснення виклику.

Додатково ще треба доповнити функції playHandler(), pauseHandler() та stopHandler() таким чином, щоб:

- коли користувач буде натискати 'Pause' або 'Stop' – призупинявся виклик функції за таймером. Якщо не вітворюється медіа потік, то синхронізувати власне нема що і не треба робити зайвих викликів;
- коли користувач натисне 'Play', має здійснитись перевірка поточного значення *contentPlayer.currentPosition*, і відповідно до цього зробити активним потрібний слайд, і передати нове значення функції виклику за таймером.

Отже, завдяки можливостям медіапрогравача як об'єкта веб-сторінки, а також потужним засобам JavaScript API та XMLHttpRequest API, можна перенести значну частину засобів синхронізації аудіовідеопотоку та статичних матеріалів лекції на сторону клієнта, що позбавить необхідності перезавантаження сторінки і зайвих викликів серверної частини. Одним словом, можна отримати застосування (client-side application) задовільної якості із відкритим кодом.

Альтернативний варіант – клієнтська частина як документи в форматі SMIL Відносно нова мультимедійна і одночасно веб технологія, яка нажаль досі не набула широкої популяризації та підтримки з боку браузерів – Synchronized Multimedia Integration Language або SMIL. Від початку її розробкою займалась компанія Real Networks з метою створення XML-подібної мови розмітки мультимедійних презентацій для програвача RealPlayer. Файли SMIL відтоді мають розширення .smil.

Із серпня 2001 року специфікація SMIL 2.0 набула статусу офіційної рекомендації W3C, як основана на XML мова розмітки документів. Поточна версія 2.1 також має статус “W3C Recommendation” з початку 2005 року [21].

У процесі розробки перебуває нова специфікація SMIL 3.0.

Застосування SMIL – це розмітка документів – складних мультимедійних презентацій. В документі визначаються часові налаштування, паралельна або послідовна подача різних частин (об'єктів) презентації, їхнє розташування у робочому вікні, параметри відображення, а також, найголовніше – підтримка більшості сучасних форматів даних: це текст і гіпертекст, зображення, анімація, різні формати аудіо і відео.

Структура документа – це розмітка за принципами XML і дуже нагадує HTML. В секції <*head*> містяться параметри відображення та метадані, в <*body*> – тіло документа, де зазначається які об'єкти коли і як мають показуватись.

```
<smil>
<head>
        <layout>
                <!-- layout tags -->
        </layout>
</head>
<body>
        <!-- body tags -->
</body>
</smil>
```

Основними керуючими тегами в <*body*> є <*seq*> (послідовний показ) і <*par*> (паралельний показ). У них вже вказуються об'єкти, точніше посилання на них, адже SMIL оперує саме посиланнями на мульмедійні об'єкти за їхніми URL адресами. Це надає перевагу в тому, що презентація може містити зовнішні ресурси, з будь-якої кількості інших веб-серверів одночасно. Варто зазначити, що в розмітці можна враховувати різну пропускну здатність для віддалених об'єктів.

Елементи-тегі презентації у мові SMIL це: <*animation*>,<*audio*>, <*brush*>, <*img*>, <*param*>, <*ref*>, <*text*>, <*textstream*>, <*video*>.

Для кожного об'єкта розмітки визначається джерело (атрибут *src)*, коли і як він буде відображатись, що іде за чим, в якій області робочого вікна воно буде показуватись і т.д.

В Інтернет доступні багато прикладів і інструкцій по застосуванню SMIL, наприклад такий корисний сайт як [22].

Після ознайомлення зі SMIL саме-собою виникає розуміння доцільності використання цієї технології для відтворення електронних лекцій. Адже лекція, з технічної точки зору – це ніщо інше як мульмедійна презентація, складена з різних частин (двох чи більше).

До того ж, можна генерувати файли у форматі .smil на стороні сервера за тими даними про матеріали електронної лекції, які вносить автор, так само, як генерувався XML (адже це і є XML). Всі часові мітки послідовного виклику демонстраційних матеріалів можна задати в розмітці документа.

Але є один суттєвий недолік: SMIL поки, що підтримується далеко не всіма браузерами. Наприклад, Internet Explorer 6 підтримував SMIL 2.1, але в версії IE 7 Microsoft чомусь відмовились від цього. Хоча в восьмій версії нібито збираються повернути підтримку цього формату.

Висновки

У роботі описано простий та ефективний розв'язок поставленої задачі створення ефективних засобів передачі, обробки і подання мультимедіа в освітніх системах. Але, усвідомлюючи, що це лише один з можливих розв'язків, ми намагалися проаналізувати альтернативні рішення.

Сподіваємося, що результати цього параграфу будуть ще одним підтвердженням таких простих фактів, що:

- електронна освіта має бути веб-орієнтованою;
- сучасні веб-браузери – це зручні і потужні середовища, в яких може оброблятись значна частина бізнес логіки застосування;
- розвиток мережевих та мультимедійних технологій сприятиме підвищенню якості передачі і відтворення складних мультимедійних презентацій, а стандартизація дозволить їм набути більшого поширення та підтримки. Приклад – перспективна мова SMIL.

На нашу думку, розробка систем підтримки проведення, запису та відтворення електронних лекцій, має базуватись на технологіях з відкритим кодом.

1. *Глибовець М.М.* Дистанційна освіта та комп'ютерні системи її підтримки, Вісник Київського університету. Серія: Фізико-математичні науки. – 2002. – Вип. № 2. – С. 179 – 192.
2. *Бублик В., Генстен К.* До питання розвитку освітніх інформаційних технологій, Наукові записки НаУКМА, Комп'ютерні науки. – 2002. – Том 19–20. – С. 28–34.
3. *Boublik V., Hesser W.* E-Learning: Challenges and Perspectives, Наукові записки НаУКМА, Комп'ютерні науки. – 2005. – Том 36. – С. 28–34.
4. *Солнцева Л.И.* Теоретические и практические проблемы современной тифлопсихологии и тифлопедагогики. М.: «ИПТК «Логос» ВОС». – 2006.
5. *Peter Brusilovsky.* 'Asynchronous Electronic Lectures in Web-based education', Carnegie Technology Education, US, 2000. – http://www2.sis.pitt.edu/~peterb/papers/WL-NEW2.pdf
6. *OpenOffice.org* Homepage http://www.openoffice.org/
7. *LizardTech:* Technical Papers on DjVu Technology http://www.lizardtech.com/products/doc/techinfo.php
8. *Mozilla* Firefox Add-ons: Zoho QuickRead https://addons.mozilla.org/en-US/firefox/addon/3328
9. *TEX* THE WORLD Firefox plug-in http://thewe.net/tex/
10. *Alexander N.oe* 'AVI File Format', online documentation, December 2006. http://www.alexander-noe.com/video/documentation/avi.pdf
11. *QuickTime 7,* Technical Brief http://images.apple.com/quicktime/pdf/QuickTime7_Tech_Brief_V2.pdf
12. *Flash* Developer Center http://www.adobe.com/devnet/flash/
13. *RealVideo* 10 Technical Overview http://docs.real.com/docs/rn/rv10/RV10_Tech_Overview.pdf
14. *John G. Apostolopoulos, Wai- tian Tan, Susie J. Wee.* 'Video Streaming: Concepts, Algorithms, and Systems', Mobile and Media Systems Laboratory, HP Laboratories Palo Alto, September 18th , 2002. - http://www.hpl.hp.com/techreports/2002/HPL-2002-260.pdf
15. *Annapureddy S., Guha S., Gkantsidis C., Gunawardena D., Rodriguez P.* 'Is HighQuality VoD Feasible using P2P Swarming', WWW 2007, May 8–12, 2007, Banff, Alberta, Canada. – http://www.scs.stanford.edu/~reddy/research/redcarpet/redcarpet-www07.pdf
16. *S. Annapureddy, C. Gkantsidis, P. Rodriguez, and L. Massoulie.* 'Providing Video-on-Demand using Peer-to-Peer Networks', New York University, NY and Microsoft Research, Cambridge, 2007. http://www.scs.stanford.edu/~reddy/research/redcarpet/redcarpet.pdf
17. *Saikat* Guha (Cornell University), Neil Daswani, Ravi Jain (Google). 'An Experimental Study of the Skype Peer-to-Peer VoIP System', US, 2006. -http://saikat.guha.cc/pub/iptps06-skype.pdf
18. *Andreas Rebs.* 'Interaktionskomponenten und wechselseitige Abstimmung der Ressourcennutzung in einem Distance-Learning-System', Magisterarbeit, Leipzig, 14. April 2003. - http://www.imn.htwk-leipzig.de/~haenssge/dipl/arbeiten/rebs-ma-dls-fin.pdf
19. *'Java* Media Framework API Guide',1998-99 Sun Microsystems, Inc. – http://www.cdt.luth.se/~johank/smd151/jmf/jmf2_0-guide.pdf.
20. *MSDN.* 'Player Object'. – http://msdn2.microsoft.com/en-us/library/bb249349.aspx
21. *Synchronized* Multimedia Integration Language (SMIL 2.1), – W3C Proposed Recommendation 27 September 2005. – http://www.w3.org/TR/2005/PR-SMIL2-20050927/
22. Multimedia for Everyone. SMIL Tutorials& Examples. – http://www.multimedia4everyone.com/