- “засоби надають додаткову інформацію про процес”;
- “засоби підвищують зручність роботи з системою”;
- “в статичному вигляді цей процес показати неможливо”;
- “для людей, які не бачать, це ефективний метод надання інформації” тощо.

1. Каптерев А.И. Мультимедиа как социокультурный феномен: учеб. пособие для вузов / А.И. Каптерев. – М.: Профиздат, 2002. – 224 с. 2. Вовк Е.Т. Информатика: уроки по Flash [электронный ресурс] / Е.Т. Вовк. – М.: КУДИЦ-Образ, 2005. – 176 с. 3. Крапивенко А.В. Технологии мультимедиа и восприятие ощущений: учеб. пособ. / А.В. Крапивенко. – М. : БИНОМ. Лаборатория знаний, 2009. – 271 с. 4. The Cambridge Handbook of Multimedia Learning / [ed. by Richard E. Mayer]. – New York: Cambridge University Press, 2005. – 680 p. 5. Популярные Web-приложения на Flash MX + CD / Чанг Т.К., Кларк Ш., Долецки Э.Е. и др. – М.: КУДИЦ-Образ, 2003. – 269 с. 6. Дунаев В.В. Macromedia Flash MX 2004 / В.В. Дунаев. – СПб.: Питер, 2004. – 368 с. 7. Давидова Е.В. Створення графіки для Web-сторінок / Е.В. Давидова // Інформатика та освіта. – 2001. – № 6. – С. 72–80.

УДК 681.3, 621.3

Ю. Морозов, І. Пастернак
Національний університет “Львівська політехніка”,
кафедра електронних обчислювальних машин

МЕРЕЖНІ ІНТЕРФЕЙСИ РІВНЯ КЛІЄНТ-СЕРВЕР

Запропоновано реалізацію клієнт-серверної взаємодії на основі моделі запит/відповідь у вигляді формул. А також тестування мережного інтерфейсу на його навантаження клієнтськими запитами.

Ключові слова: мережа, програмне забезпечення, тестування мережі.

An implementation of client-server interaction model based on the request / response in the form of formulas. As well as testing network interface to load its client requests.

Key words: network, software, network testing.

Вступ. Постановка проблеми

Зрозуміло, що в загальному, щоб прикладна програма, яка виконується на робочій станції, могла запросити послугу в деякого сервера, як мінімум потрібно деякий інтерфейсний програмний рівень, що підтримує такого роду взаємодію (було б щонайменше неприродно вимагати, щоб прикладна програма прямо користувалася примітивами транспортного рівня локальної мережі). Із цього і випливають основні принципи системної архітектури “клієнт-сервер”.

Система розбивається на дві частини, які можна виконувати в різних вузлах мережі, – клієнтську й серверну частини. Прикладна програма або кінцевий користувач взаємодіють із клієнтською частиною системи, що у найпростішому випадку забезпечує просто надмережний інтерфейс. Клієнтська частина системи за потреби звертається по мережі до серверної частини. Помітимо, що в розвинених системах мережне звертання до серверної частини може й не знадобитися, якщо система може вгадувати потреби користувача, і в клієнтській частині втримуються дані, здатні задовольнити його наступний запит.

Інтерфейс серверної частини визначений і фіксований. Тому можливо створення нових клієнтських частин існуючої системи (приклад інтероперабельності на системному рівні).

Основною проблемою систем, заснованих на архітектурі “клієнт-сервер”, є те, що відповідно до концепції відкритих систем від них потрібна мобільність у якомога ширшому класі апаратно-програмних рішень відкритих систем. Навіть якщо обмежитися UNIX-орієнтованими локальними мережами, у різних мережах застосовується різна апаратура й протоколи зв'язку. Спроби створення систем, що підтримують всі можливі протоколи, приводить до їхнього перевантаження та шкодить функціональності.

Ще складніший аспект цієї проблеми пов'язаний з можливістю використання різних подань даних у різних вузлах неоднорідної локальної мережі. У різних комп'ютерах може існувати різна адресація, подання чисел, кодування символів тощо. Це особливо істотно для серверів високого рівня: телекомунікаційних, обчислювальних, баз даних.

Загальним рішенням проблеми мобільності систем, заснованих на архітектурі “клієнт-сервер” є опора на програмні пакети, що реалізують протоколи вилученого виклику процедур (RPC – Remote Procedure Call). При використанні таких засобів звертання до сервісу у вилученому вузлі виглядає як звичайний виклик процедури. Засоби RPC, у яких, природно, утримується вся інформація про специфіку апаратур локальної мережі й мережних протоколів, переводить виклик у послідовність мережних взаємодій. Отже, специфіка мережного середовища й протоколів прихована від прикладного програміста.

У разі виклику вилученої процедури програми RPC роблять перетворення форматів даних клієнта в проміжні машинно-незалежні формати й потім перетворення у формати даних сервера. Під час передавання відповідних параметрів виробляються аналогічні перетворення. Якщо система реалізована на основі стандартного пакета RPC, її можна легко перенести в будь-яке відкрите середовище.

Технологія “клієнт-сервер” стосовно СУБД зводиться до поділу системи на дві частини – додаток-клієнт (front-end) і сервер бази даних (back-end). Ця архітектура сполучає кращі ознаки обробки даних на мейнфреймах і технології “файл-сервер”. Від мейнфреймів технологія “клієнт-сервер” запозичила такі ознаки, як централізоване адміністрування, безпека, надійність. Від технології “файл-сервер” успадковані низька вартість і можливість розподіленої обробки даних, використовуючи ресурси комп'ютерів-клієнтів. Зараз графічний інтерфейс користувача став стандартом для систем “клієнт-сервер”. Крім того, архітектура “клієнт-сервер” значно спрощує й прискорює розробку додатків за рахунок того, що правила перевірки цілісності даних перебувають на сервері. Неправильно працюючий клієнтський додаток не може спричинити втрату або перекручування даних. Всі ці можливості, раніше властиві тільки складній і дорогій системі, тепер доступні навіть невеликим організаціям. Вартість устаткування, програмного забезпечення й обслуговування для персональних комп'ютерів у десятки разів нижчі, ніж для мейнфреймів.

Для коректції взаємодії клієнта з сервером у глобальній мережі використовується мережний інтерфейс. Мережний інтерфейс – це два або більше комп’ютери, об’єднані кабелем так, щоб вони могли обмінюватись інформацією.

Комп'ютерна мережа (Network) – це два і більше персональних комп’ютерів, з'єднаних між собою з метою швидкого обміну даними та спільного використання ресурсів. Для реалізації мережі необхідні компоненти двох типів: апаратні та програмні. Апаратна частина забезпечує фізичне з'єднання комп'ютерів. Для налаштування під'єднання персонального комп’ютера до мережі потрібно послідовно виконати такі операції:

1 етап. Фізично під'єднати модем до персонального комп’ютера і телефонної лінії. Після цього необхідно встановити його драйвери та налаштувати параметри роботи.

2 етап. Налаштувати параметри під'єднання до серверу віддаленого доступу.

3 етап. Налаштувати параметри протоколу TCP/IP.

З'єднання персональних комп’ютерів може відбуватись за допомогою кабелю та мережевих адаптерів, або кабелю, під'єднаного до портів (пряме кабельне з'єднання), телефонної чи оптоволоконної лінії та модему, або радіозв'язком за допомогою радіомодему [1,10].

Мережне програмне забезпечення складається з двох найважливіших компонентів:

– мережевого програмного забезпечення, встановленого на комп’ютерах-клієнтах;
– програмного забезпечення, встановленого на серверах.

Операційна система зв’язує всі комп’ютери і периферійні пристрої в мережі, координує функції всіх комп’ютерів і периферійних пристроїв у мережі, забезпечує захищений доступ до даних і периферійних пристроїв у мережі. Програмна частина мережі – це мережева операційна система, що забезпечує роботу персонального комп’ютера, протоколи і прикладні програми, що підтримують роботу в мережі.

Аналіз літературних джерел

Основним завданням, що виконується під час створення таких неоднорідних комп'ютерних мереж, є забезпечення сумісності обладнання за електричними та механічними характеристиками і забезпечення сумісності програм та даних за системою кодування і форматом. Виконання цього завдання в мережі Інтернет ґрунтується на моделі взаємодії відкритих систем (OSI). Відповідно до цієї моделі стандартизація апаратури і програмного забезпечення в мережі Інтернет проводиться на підставі протоколів, що утворюють ієрархічну систему правил взаємодії [3,12]. Як і в усякій іншій мережі, в Інтернет можна використовувати до семи рівнів взаємодії між комп'ютерами.

Розглянемо схему створення TCP-з'єднання:

- Припустимо, що хосту А необхідно створити TCP-з'єднання з хостом В. Тоді А посилає на В таке повідомлення: SYN, ISSa.
- Це означає, що в переданому А повідомленні встановлений біт SYN. (Synchronize Sequence Number), а в поле Sequence Number установлено початкове 32-бітне значення ISSa (Initial Sequence Number). Хост В відповідає: SYN, АСК, ISSb, ACK(ISSa+l).
- У відповідь на отриманий від А запит В посилає повідомлення, у якому встановлені біт SYN і біт АСК; у поле Sequence Number хостом В задається своє початкове значення лічильника – ISSb; поле Acknowledgment Number містить значення ISSa, отримане в першому пакеті від хосту А і збільшене на одиницю.
- Хост А, завершуючи рукостискання (handshake), посилає: АСК, ISSa+1, ACK(ISSb+l). У цьому пакеті встановлений біт АСК; поле Sequence Number містить значення ISSa+1; поле Acknowledgment Number містить значення ISSb+1. Посилкою цього пакета на хост В закінчується триступінчастий handshake, і TCP-з'єднання між хостами А і В вважається встановленим.
- Тепер хост А може посилати пакети з даними на хост В по тільки що створеному віртуальному TCP-каналу; передається така інформація: АСК, ISSa+1, ACK(ISSb+l); DATA.

З розглянутої схеми створення TCP-з'єднання видно, що єдиними ідентифікаторами, крім IP-адреси ініціатора з'єднання, TCP-абонентів і TCP-з'єднання, є два 32-бітних параметри Sequence Number і Acknowledgment Number. Отже, для формування хибного TCP-пакета атакуючому необхідно знати поточні ідентифікатори для цього з'єднання – ISSa і ISSb. Це означає, що кракерові досить, підібравши відповідні поточні значення ідентифікаторів TCP-пакета для даного TCP-з'єднання (наприклад, дане з'єднання може являти собою FTP- чи TELNET-під’єднання), послати пакет з будь-якого хосту в мережі Internet від імені одного з учасників даного з'єднання (наприклад, від імені клієнта), вказуючи в заголовку IP-пакета його IP-адресу, і цей пакет буде сприйнятий як правильний. Протоколи TCP/IP вирішують проблему транспортування пакетів, їх вміст може бути різним (дані користувача, службова інформація). Пакети переміщуються від вузла до вузла, утворюючи нижній рівень функціонування мережі. Наповнюють пакети даними програми верхнього рівня, з якими працює користувач. Саме ці програми визначають для користувача його можливості в мережі. Розробляються вони на підставі єдиних правил (прикладних протоколів) і називаються сервісами (службами, послугами) мережі Інтернет. Практично всі послуги мережі побудовані на принципі «клієнт-сервер». Сервер (у мережі Internet) – це комп'ютер або програма, здатні надавати клієнтам деякі мережні послуги [2, 11]. Клієнт – прикладна програма, завантажена в комп'ютер користувача, яка забезпечує передачу запитів до сервера й одержання відповідей від нього. Різні сервіси мають різні прикладні протоколи. У міру розвитку мережі з'являються нові протоколи (сервіси), змінюючи її вигляд і стрімко розширюючи коло користувачів. Отже, щоб скористатися якоюсь із служб мережі Інтернет, необхідно встановити на комп'ютері клієнтську програму, здатну працювати за протоколом цієї служби.

Постановка задачі

Оцінити межі застосування мережного інтерфейсу, його масштабованість та доступність.

Аналіз отриманих результатів досліджень

Розглянувши основні проблеми, що є перед моделюванням мережного інтерфейсу типу клієнт-сервер, перейдемо до формального опису таких інтерфейсів. Враховуючи основні принципи опису мережного інтерфейсу, виділимо основні компоненти мережного інтерфейсу – її входи та виходи. Входом мережного інтерфейсу є множина запитів клієнтів, виходами – множина відповідей на запити. У загальному випадку реакція визначається не лише поточним запитом, а й усіма попередніми. Для усунення необхідності кожен раз обробляти усі попередні запити клієнта, вводиться поняття стану мережного інтерфейсу. Стан мережного інтерфейсу є агрегованою історією запитів до цього інтерфейсу. У мережному інтерфейсі виділяються окремі рівні (рис.1), тобто вона є багаторівневою і на кожному рівні міститься не менше однієї автономної підсистеми (яка фактично є окремою системою) [4–5]. На кожному з рівнів (крім верхнього та нижнього) традиційно розглядається три вхідних потоки – потік зовнішніх збурень, потік розпоряджень з вищого рівня – специфікація подання, потік реакцій нижчого рівня – внутрішньо-компонентний.

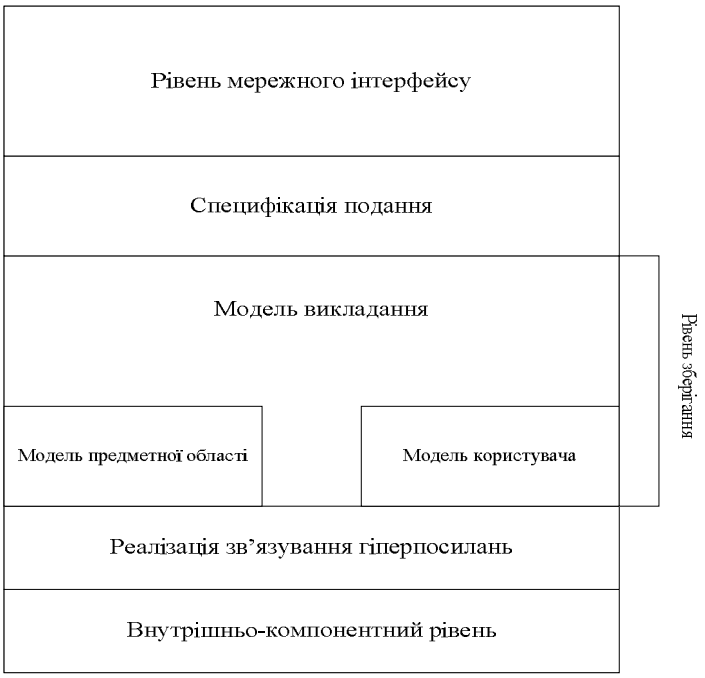

Рис. 1. Вигляд рівнів мережного інтерфейсу

Мережний інтерфейс повинен забезпечувати можливість ієрархічного з'єднання клієнта з сервером у мережі. Таке з'єднання базується на спрямуванні вихідного потоку нижчого рівня у вищий рівень мережного інтерфейсу, що разом з вхідним збудженням становить вхідний сигнал для вищого рівня мережного інтерфейсу. У формуванні відповіді сервісу на запит користувача з використанням мережного інтерфейсу (рис. 1) виділимо такі етапи:

1) отримання запиту від користувача або вищого сервісу – рівень моделі викладання;
2) формування запитів до нижчих сервісів – рівень специфікації подання;
3) отримання відповідей від нижчих сервісів – внутрішньо-компонентний рівень ;
4) формування відповіді на запит – рівень зв'язування для гіперпосилань.

Аналізуючи ці етапи, відзначимо два моменти:

- якщо розглядати тільки верхній зріз мережного інтерфейсу, то під час його опису потрібно виділяти як потік запиту користувача, так і потік реакцій підкомпонент;

- в мережному інтерфейсі загалом (зі всіма рівнями включно), слід визначати лише один вхідний потік – запити користувачів (адже відповіді нижчого рівня мережного інтерфейсу визначаються запитами вищого сервісу, а отже, вхідними запитами користувачів).

Деталізація мережного інтерфейсу проводиться до певного найнижчого рівня. Для функцій цього рівня розглядаються лише вхідні запити користувача. Такий підхід узгоджується з твердженням про неістотність опису вхідних потоків від нижчого рівня мережного інтерфейсу для мережного інтерфейсу взагалі. Весь мережний інтерфейс з усіма її рівнями розглядається як мережний інтерфейс найнижчого рівня (адже в ній завжди можна виділити компоненти найнижчого рівня).

Під час аналізу поведінки мережний інтерфейс спрощується завдяки усуненню параметрів з запитів користувача та його реакцій. Досліджуючи певні особливості функціонування мережного інтерфейсу, можна взагалі усувати або запит користувача, або вхідний потік від нижчого рівня мережного інтерфейсу. Досліджуючи мережний інтерфейс, загалом, а не тільки її певний зріз, розглядаються лише вхідні запити користувачів [8, 13].

При розгляді правил композиції мережного інтерфейсу та його використання внаслідок застосування до їх компонент певних операцій, враховуємо, що запити до компонент формуються мережний інтерфейс, а відповіді компонент направляються до неї для формування остаточної реакції. Мережний інтерфейс описується традиційним впорядкованим набором:

$$IS = (Q, R, A, St, f, y), \tag{1}$$

де Q – вхідні запити ІС; R – відповіді нижчого рівня мережного інтерфейсу; A – вихідний алфавіт ІС, St – множина станів системи; f, ψ – функції переходів та виходів.

Розглянемо детальніше кожен з об'єктів. Множину символів, що становлять вхідний алфавіт мережного інтерфейсу, опишемо у такий спосіб:

$$Q = \{Qi\}, \tag{2}$$

$$Qi = \left\{ Id_Q^{(i)}, Pq^{(1)}_1, \cdots, Pq^{(1)}_{N_Q^{(1)}} \right\}, \tag{3}$$

$$P_j^{(i)} \subset D^{(1)}_{Q_j^{(1)}} \times \cdots \times D^{(l)}_{Q_j^{(1)}}, l = N^{(j)}_{P_{Qi}}, \tag{4}$$

де множина IdQ – множина унікальних ідентифікаторів запитів.

Множину символів, що становлять вихідний алфавіт мережного інтерфейсу, опишемо у такий спосіб:

$$A = \{Ai\}, \tag{5}$$

$$Ai = \left\{ Id_A^{(i)}, Pa^{(1)}_1, \cdots, Pa^{(1)}_{N_A^{(1)}} \right\}, \tag{6}$$

$$P_j^{(i)} \subset D^{(1)}_{A_j^{(1)}} \times \cdots \times D^{(l)}_{A_j^{(1)}}, l = N^{(j)}_{P_{Ai}}, \tag{7}$$

де IdA – множина унікальних ідентифікаторів відповідей.

Множину символів, що становлять відповіді нижчого рівня мережного інтерфейсу, опишемо так:

$$R = \{Ri\}, \tag{8}$$

$$Ri = \left\{ Id_R^{(i)}, \Pr^{(1)}_1, \cdots, \Pr^{(1)}_{N_R^{(1)}} \right\}, \tag{9}$$

$$P_j^{(i)} \subset D^{(1)}_{R_j^{(1)}} \times \cdots \times D^{(l)}_{R_j^{(1)}}, l = N^{(j)}_{P_{Ri}}, \tag{10}$$

де IdR – множина унікальних ідентифікаторів відповідей нижчого рівня мережного інтерфейсу.

Кожен запит до мережного інтерфейсу та його відповідь містять дві частини: ідентифікаційну та параметричну. Ідентифікаційна – однозначно визначає тип запиту (наприклад: запит на отримання даних, оновлення, розміщення замовлення тощо). Параметри є фактично додатковою інформацією, що супроводжує запити та відповіді на них мережного інтерфейсу (наприклад: текст Web-сторінки, вибірка з бази даних, прайс-лист системи Internet-маркетингу – параметри відповіді, критерії на вибірку даних, нове замовлення – параметри запиту).

Отже, формальний опис стану мережного інтерфейсу має рекурсивний характер (до певного найнижчого рівня деталізації, на якому стани мають атомарний характер).

Кількість станів обмежується кількістю можливих станів підсистем.

Проте реально їх кількість істотно менша внаслідок обмежень та взаємозв'язків, що можуть існувати між станами мережного інтерфейсу. Серед таких обмежень виділимо два основні типи:

- внутрішні стосовно одне одного (типу обмежень на унікальність або простих обмежень типу Check);
- зовнішні (типу зовнішніх ключів).

Ієрархія станів мережного інтерфейсу повинна узгоджуватися та породжуватися відповідною FHD. Найнижчий рівень деталізації кожної з компонент відповідає атомарним функціям.

З аналізу формальної моделі клієнт-сервер системи доходимо висновку про принципову схожість класичних мережних інтерфейсів, що будуються на технологіях з архітектурою клієнт-сервер. Фактично, ми говоримо про узагальнення понять баз даних у моделях найрізноманітніших інформаційних систем (таких, як Web-системи). База даних системи у такому разі розглядається у ширшому розумінні, ніж просто набір спеціальних об'єктів (наприклад, таблиць), що обробляються та підтримуються певною СУБД. Навіть набір статичних Web-сторінок, що зберігаються як окремі файли у файловій системі сервера, розглядається як база даних. Модель даних повинна належно документуватись та проектуватись (зазвичай, з використанням такого інструментарію як ERD). Проте не викликає сумнівів те, що для побудови серйозних мережних інтерфейсів з великими об'ємами даних як засоби підтримки даних повинні використовуватися промислові СУБД реляційного або об'єктно-реляційного типу. Встановлення відповідності між базою даних системи та станом її автоматної моделі не є випадковим, а грунтується на фундаментальних засадах теорії формальних систем. Актуальний стан мережного інтерфейсу є фактично історією мережного інтерфейсу, тобто результатом усіх попередніх запитів клієнтів.

Визначення точної реакції мережного інтерфейсу на запит клієнта залежить не лише від ідентифікаторів запиту та стану мережного інтерфейсу. Реакція мережного інтерфейсу залежить також від параметрів Pj запиту та бази даних мережного інтерфейсу. У відповіді мережного інтерфейсу також міститиметься, окрім ідентифікатора, й параметрична компонента. Проте в межах автоматної моделі неможливо повністю описати реакцію мережного інтерфейсу на запит клієнта (враховуючи, що множина можливих запитів з параметрами та станів з БД може мати велику розмірність, зокрема, бути скінченними або мати потужність континуум). Параметрична компонента системи є другорядною при описі функціональності мережного інтерфейсу. Параметрична компонента усувається з розгляду шляхом використання стохастичних та недетермінованих автоматних моделей.

Розглянемо проблему формування ймовірностей для опису реакції мережного інтерфейсу на запит. Можливі два підходи до її вирішення:

1) визначення ймовірності конкретної реакції мережного інтерфейсу на запит користувача зі врахуванням значення конкретного параметра;

2) визначення загальної ймовірності конкретної реакції мережного інтерфейсу на запит користувача без врахування значення конкретного параметра.

Перший підхід забезпечує максимально можливу визначеність реакції мережного інтерфейсу на запит користувача.

Проте недоліком такого підходу є велика складність визначення ймовірності переходу. Крім того, під час використання операції усунення параметра потрібно буде перераховувати ймовірності переходів. Для проведення фізичної оптимізації також доцільно використовувати загальні значення ймовірностей реакції мережного інтерфейсу [6–7]. Тому доцільніше використовувати другий підхід. У такому разі ймовірність реакції визначається для кожної пари (St,Q) без врахування значень параметрів запиту до мережного інтерфейсу та її бази даних. Визначеність поведінки мережного інтерфейсу буде меншою, ніж у першому випадку, проте для усунення параметрів з розгляду та фізичної оптимізації мережного інтерфейсу не потрібно буде додаткових перетворень. Якщо в

мережному інтерфейсі існують запити з такими параметрами, що істотно впливають на реакцію мережного інтерфейсу та є важливими для її оптимізації, тоді пару (параметр, запит) можна перетворити в окремий запит.

Тестування навантаження мережного інтерфейсу необхідне у випадках, коли потрібно отримати відповіді на такі питання:

- що відбудеться з системою у разі збільшення кількості користувачів;
- де межа масштабованості мережного інтерфейсу;
- яке серверне програмне забезпечення (ПЗ) краще відповідає потребам мережного інтерфейсу в сенсі продуктивності;
- яке устаткування необхідне для комфортної роботи з мережним інтерфейсом;
- що є потенційним вузьким місцем мережного інтерфейсу з погляду продуктивності.

На перший погляд здається, що на перераховані вище питання можна відповісти і без проведення комплексних досліджень продуктивності мережного інтерфейсу. Але це справедливо тільки для порівняно простих мережних інтерфейсів і навіть для них справедливо далеко не завжди.

Під час ідеального тестування навантаження тими або іншими способами емулюється навантаження від всіх джерел запитів до мережного інтерфейсу (користувачі, зовнішні джерела даних тощо), а не лише звернення користувачів до певних модулів функціональності мережного інтерфейсу.

Фази процесу тестування навантаження. У будь-якому проекті навантажувального тестування можна виділити такі стадії:

1) дослідження мережного інтерфейсу;
2) підготовка проектної документації;
3) створення інструментарію тестування;
4) проведення тестування;
5) обробка результатів і підготовка звіту.

На етапі дослідження системи відбувається ознайомлення з програмно-апаратним комплексом на різних рівнях:

- рівень кінцевого користувача;
- рівень персоналу, що обслуговує мережний інтерфейс;
- прикладний рівень;
- системний рівень;
- апаратний (фізичний рівень);

Рівень кінцевого користувача мережного інтерфейсу дає уявлення про бізнес-функції і комфортні часові межі відгуку операцій. На цьому етапі відбувається визначення варіантів використання мережного інтерфейсу, спектр навантаження на мережний інтерфейс.

Рівень персоналу, що обслуговує мережний інтерфейс, необхідний для виділення функціональних частин комплексу. Визначаються зв’язки між частинами мережного інтерфейсу і проводиться їх оцінка на потенційні вузькі місця.

Прикладний рівень дає уявлення про внутрішнє влаштування комплексу з погляду розробників ПЗ. Рівень може бути корисний для оцінки архітектурних рішень, які застосовуються при створенні комплексу, що тестується. Оцінюється кількісний і якісний склад моніторів продуктивності і ресурсів.

Системний рівень необхідний для оцінки якості розгортання мережного інтерфейсу і визначення точок установки моніторів системних ресурсів.

Аналіз апаратного рівня дає змогу визначити потенційні вузькі місця мережного інтерфейсу на рівні апаратного забезпечення [10,14].

Без ретельного системного аналізу досліджуваного ПЗ неможливе проведення тестування навантаження. Цей етап дає відповідь на питання: як проводитиметься тестування, які інструменти буде потрібно, які зовнішні мережні інтерфейси мають емулювати, як здійснюватиметься моніторинг продуктивності досліджуваного ПЗ.

На цьому етапі, зазвичай, застосовується реверсивне проектування (reverse engineering). Наприклад, записується SQL графік від додатка клієнта до сервера баз даних [15–16]. У найпростішому випадку потрібно визначити які запити – це просто завантаження даних з довідників, а які є реакцією аплікації на дії користувача і вимагають параметризації (внесення змінних значень, наприклад, залежних від введених користувачем даних). Коректна параметризація – умова створення навантаження, адекватного реальному. Те саме справедливо і для HTTP трафіку у разі навантаження web-аплікацій.

Для мережного інтерфейсу, що працюють з великими об'ємами даних, завдання проведення випробувань навантажень набувають особливого значення. Часто мережний інтерфейс показує хорошу продуктивність на даних невеликого об'єму, але із збільшенням навантаження виникають різноманітні проблеми, які можуть бути пов'язані з неоптимальною побудовою запитів, технологічними конфліктами різних елементів мережного інтерфейсу, обмеженнями системних ресурсів тощо.

Навантажувальне тестування мережного інтерфейсу з клієнт-серверною архітектурою призначене для визначення максимальної кількості запитів від клієнта (пропускної здатності) та оптимальної конфігурації сервера при таких станах мережного інтерфейсу:

- нормальному (base);
- навантаженому (load);
- критичному (stress).

У всіх випадках під час тестування відбувається визначення часу стабільного перебування у фіксованому стані і межа переходу у наступний (рис. 2).

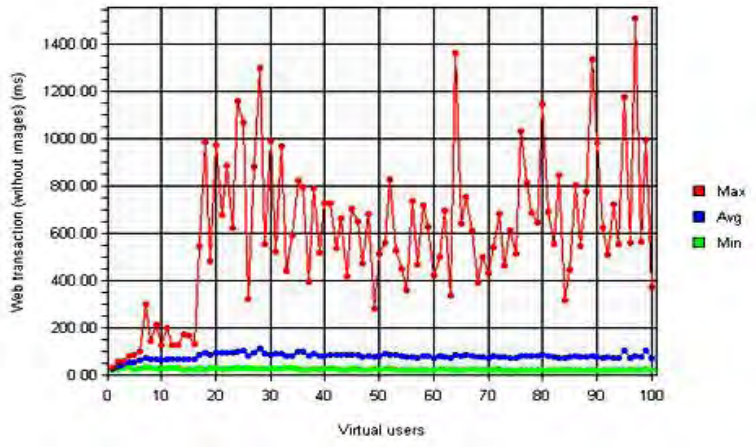

Рис. 2. Залежність часу відгуку сервера від кількості запитів від клієнта

Зазначимо, що після перевищення межі критичного навантаження відбувається збій мережного інтерфейсу різного ступеня складності.

Етапність, методика і інструментарій. У проектах з тестування навантаження можна виділити такі етапи:

- аналіз і проектування моделі навантаження;
- налаштування випробувального стенда (рис. 3);
- розробка типових сценаріїв;
- реалізація моделі навантаження;
- проведення навантажувального тестування;
- аналіз результатів випробувань і побудова моделей прогнозу.

Необхідною основою для проведення навантажувального тестування є інструментальні засоби, що дозволяють в лабораторних умовах емулювати складне оточення реального світу телекомунікаційних, клієнт-серверних і Інтернет-взаємодій і виконати усестороннє тестування мережного інтерфейсу. Завдання таких інструментальних засобів – організація лабораторного

випробувального стенда, який емулюватиме від десятків до тисяч користувачів, що надсилають і одержують інформацію, відтворюючи тим самим складну взаємодію між клієнтським і серверним ПЗ, базами даних, Інтернет-серверами і іншими системами [9].

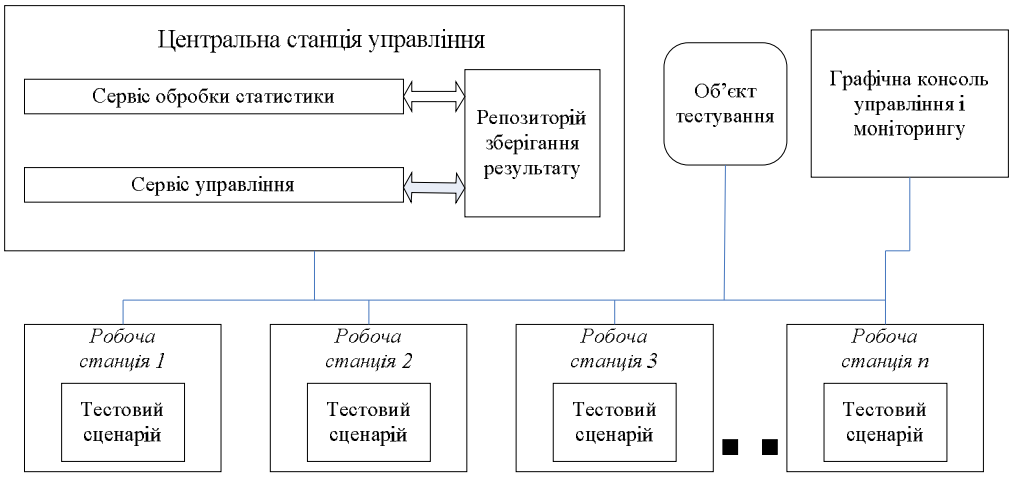

Рис. 3. Структура випробувального стенда

Більшість розвинених інструментальних засобів тестування навантаження. наприклад WAPT, Apache, JMeter, Mercury LoadRunner, дозволяють організувати випробувальний стенд з розподіленим виконанням тестів і централізованим управлінням.

Зазвичай, структура такого стенда включає:

1) об'єкт випробувань;

2) центральну станцію управління, де розташовуються сервіси, репозиторій результатів тестування і графічна консоль користувача;

3) декілька робочих станцій моделювання навантаження, на яких розташовані компоненти для імітації діяльності користувачів (до декількох сотень кожна);

4) додаткові станції моніторингу і генерації тестових даних для забезпечення безперебійного постачання тестовими даними робочих станцій. Також на окремих станціях запускаються вимірювальні монітори, що здійснюють спостереження за процесами обробки і передають виміряну статистику на центральну станцію.

Моделювання навантаження здійснюється шляхом виконання тестових транзакцій, кожна з яких містить певний набір тестів, що виконуються в послідовному або довільному порядку (рис.4):

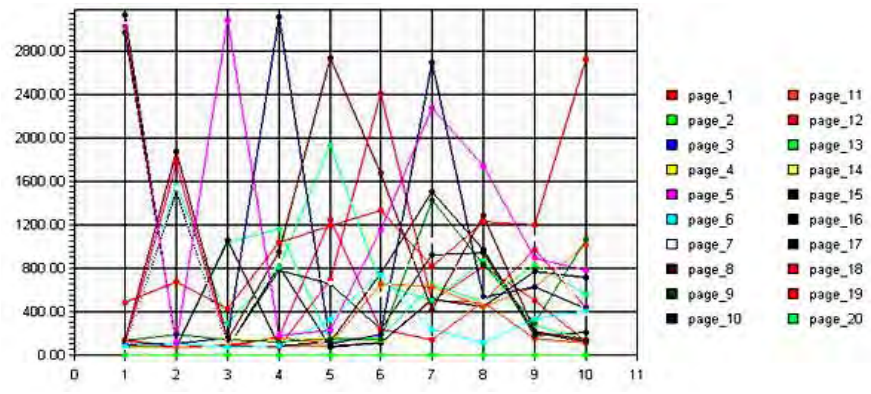

Рис. 4. Залежність середнього часу (в мс.) завантаження всіх сторінок (Pages) зі сценарію для кожного віртуального користувача (Virtual Users)

Важливим є також визначення максимального часу відгуку серверу, за якого відбувається робота системи у заданому стані (рис. 5):

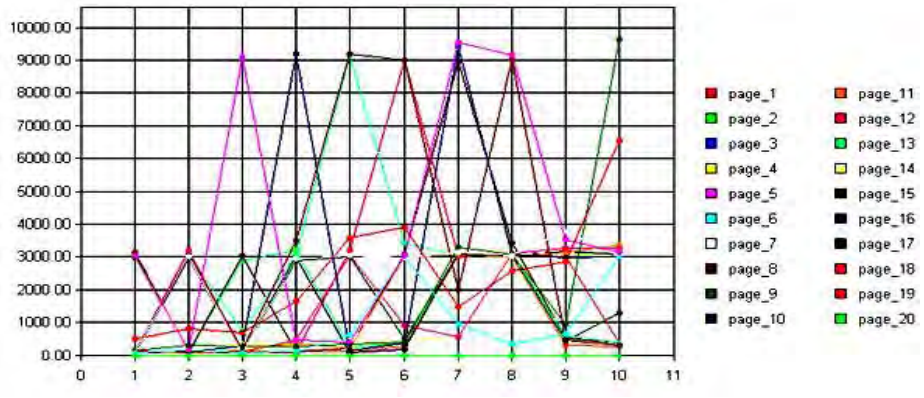

Рис. 5. Максимальний час (в мс.) завантаження всіх сторінок зі сценарію для кожного віртуального користувача

Розглянемо проблему формування ймовірностей для опису реакції мережного інтерфейсу на запит. Можливі два підходи до її вирішення:

1) визначення ймовірності конкретної реакції мережного інтерфейсу на запит користувача із врахуванням значення конкретного параметра;

2) визначення загальної ймовірності конкретної реакції мережного інтерфейсу на запит користувача без врахування значення конкретного параметра.

Перший підхід забезпечує максимально можливу визначеність реакції мережного інтерфейсу на запит користувача. Проте недоліком такого підходу є велика складність визначення імовірності переходу. Крім того, при використанні операції усунення параметра потрібно буде перераховувати імовірності переходів. Для проведення фізичної оптимізації також доцільно використовувати загальні значення ймовірностей реакції мережного інтерфейсу. Тому доцільніше використовувати другий підхід. У такому разі ймовірність мережного інтерфейсу визначається для кожної пари (St,Q) без врахування значень параметрів запиту до мережного інтерфейсу та її бази даних. Визначеність поведінки мережного інтерфейсу буде меншою ніж у першому випадку, проте для усунення параметрів з розгляду та фізичної оптимізації мережного інтерфейсу непотрібно буде додаткових перетворень. Якщо в мережного інтерфейсу існують запити з такими параметрами, що суттєво впливають на реакцію системи та є важливими для її оптимізації, тоді пару (параметр, запит) можна перетворити в окремий запит.

Висновки

Мережний інтерфейс – це два і більше ПК з'єднаних між собою з метою швидкого обміну даними та спільного використання ресурсів. Для реалізації мережного інтерфейсу необхідні компоненти двох типів: апаратні та програмні. Апаратна частина забезпечує фізичне з'єднання комп'ютерів. Вона поділяється на локальні та глобальні мережі. Для організації мережного інтерфейсу необхідна наявність мережевого програмного забезпечення, фізичного середовища передачі і комутуючі пристрої. Основними результатами проведення навантажувального тестування є обчислення показників продуктивності мережного інтерфейсу – такі, як час відгуку сервера на запит клієнта і завантаженість системних ресурсів сервера залежно від кількості користувачів. Відповідно до результатів може проводитись модифікація апаратно-програмної конфігурації мережного інтерфейсу.

1. Липаев В.В. Обеспечение качества программных средств. Методы и стандарты. – М.: Синтег, 2001. – 246 с. 2. Макгрегор Дж., Сайкс Д. Тестирование объектно-ориентированного программного обеспечения. – К.: Диасофт, 2002. – 432 с. 3. Тамре Л. Введение в тестирование программного обеспечения. – М.: Изд. дом "Вильямс", 2003. – 368 с. 4. Кеннет Г. Основы сетей Windows. – К.: Диалектика. 5. Бормотов С.В. Системное администрирование на 100%. – СПб.: Питер, 2006. – 256. 6. Єд Уілсон., Моніторинг і аналіз мереж. – М.: Лори, 2002. 7. Alex WebKnacKer

Быстро и легко. Хакинг и антихакинг: защита и нападение: Учеб. пособие. – М.: Лучшие книги, 2004 – 400 с. 8. Локальная сеть своими руками. 100% самоучитель : [учеб. пособие] / И.Я. Минаев. – М.: ТЕХНОЛОДЖИ, 3000, 2004. – 368 с. 9. Поляк-Брагинский А.В. Администрирование сети на примерах. – СПб.: БХВ-Петербург, 2005. – 320 с. 10. Мамаев М., Петренко С. Технология защиты информации в Интернете. Специальный справочник. – СПб.: Питер. 2002. – 848 с. 11. Макгрегор Дж., Сайкс Д. Тестирование объектно-ориентированного программного обеспечения. – К: Диасофт, 2002. – С. 432. 12. Тамре Л. Введение в тестирование программного обеспечения. – М.: Изд. дом "Вильямс", 2003. – С. 368. 13. Бормотов С.В. Системное администрирование на 100%. – СПб.: Питер, 2006. – С. 256. 14. Alex WebKnacKer Быстро и легко. Хакинг и антихакинг: защита и нападение. Учеб. пособие. – М.: Лучшие книги, 2004. – С. 400. 15. Гамм Э., Хелм Р., Джонсон Р., Влиссидес Д. Приёмы объектно-ориентированного проектирования. Паттерны проектирования. – Питер, Москва, 2007. – С. 366. 16. Джон Влиссидес. Применение шаблонов проектирования. – М., 2003. – С. 130.

УДК 004.652

Ю.В. Парфененко, В.В. Шендрик, С.І. Красніков
Сумський державний університет,
кафедра комп'ютерних наук

КОНЦЕПТУАЛЬНА МОДЕЛЬ ІНФОРМАЦІЙНОЇ СИСТЕМИ АНАЛІЗУ ТЕПЛОЗАБЕЗПЕЧЕННЯ

Описано розроблення концептуальної моделі аналізу процесу теплозабезпечення будівель та логічну структуру бази даних.

Ключові слова: концептуальна модель, база даних, інформаційна система, теплозабезпечення.

This article describes the the conceptual model of process analysis heating supplies of buildings and description of the logical structure of the database.

Key words: conceptual model, database, information system, heating supplies.

Вступ

Сьогодні видатки на оплату енергоносіїв у структурі бюджету органів місцевого самоврядування займають дуже значну частку. Це зумовлено тим, що рівень споживання первинних енергоносіїв, таких як природний газ зростає/залишається високим. Лише поодинокі теплогенеруючі підприємства повністю перейшли на альтернативні види палива. Тому проблема ефективного використання бюджетних коштів, які спрямовуються на задоволення потреб об'єктів бюджетної сфери в тепловій енергії, нині є надзвичайно гострою. У період енергетичної кризи, зумовленої дефіцитом ресурсів та їх високою вартістю, в Україні не поодинокими стали випадки, коли бюджетні установи несподівано припиняють свою роботу через несплату рахунків за енергію, або тривалість їх роботи скорочується до мінімуму у зв'язку із неспроможністю установи сплатити за спожиту енергію в повному обсязі. Іноді, навпаки, спостерігається значне перевикористання теплової енергії у періоди опалювального сезону з доволі високими температурами навколишнього середовища. Отже, проблема моніторнингу споживання теплової енергії з метою її раціонального використання сьогодні є актуальною. Запровадження інформаційної системи обліку та аналізу споживання теплової енергії дозволить забезпечити регулювання її необхідної кількості. Це дасть змогу забезпечити підтримку належного мікроклімату в опалювальних приміщеннях та уникнути ситуації перевикористання теплової енергії понад нормовану, що є першим кроком до енергозбереження.