

Міністерство освіти і науки України
Запорізький національний технічний університет

Кафедра програмних засобів

МЕТОДИЧНІ ВКАЗІВКИ
до проведення лабораторних робіт
з дисципліни

“Технології розробки мережевих додатків”

ДЛЯ МАГІСТРІВ НАПРЯМУ “ПРОГРАМНА ІНЖЕНЕРІЯ”
(всіх форм навчання)

2016

Методичні вказівки до проведення лабораторних робіт з дисципліни “Технології розробки мережевих додатків” для магістрів напрямку “Програмна інженерія” денної форми навчання (всіх форм навчання) /– Запоріжжя: ЗНТУ, 2016. – 88с.

Автори: О.О. Степаненко , к.т.н., доцент

Рецензент: С.О. Субботін, к.т.н., доцент

Відповідальний
за випуск: С.О. Субботін, д.т.н., проф.

Затверджено
вченою радою інституту
інформатики та
радіоелектроніки

Протокол № 1
від “15” березня 2010 р.

Затверджено
на засіданні
кафедри
програмних
засобів

Протокол № 14
від “08” червня
2012 р.

Методичні вказівки призначені для виконання курсу лабораторних робіт з дисципліни «Технології розробки мережових додатків».

Описуються базові поняття технологій BsdCокетів, показується типова структура мережних додатків. Приводяться варіанти завдань для лабораторних робіт із проектування мережних додатків з використанням технологій сокетів.

Розглядаються технології проектування мережних додатків на рівні команд прикладних протоколів на прикладі бібліотеки Internet Direct. Приводяться варіанти завдань для лабораторних робіт із програмування клієнтських і серверних додатків.

Описуються мережні утиліти операційних систем, застосовувані при розробці й налагодженню мережних додатків.

ЗМІСТ

1. ВИВЧЕННЯ ІНТЕРФЕЙСУ СОКЕТІВ.....	6
<i>Розробка мережних додатків за допомогою технології сокетів</i>	<i>7</i>
ЛАБОРАТОРНА РАБОТА №1. Використання сокетів для рішення типових задач	18
Система обміну повідомленнями	19
Система передачі файлів	20
Система електронної пошти	21
Інформаційна система	22
Калькулятор.....	24
Система дистанційного тестування.....	25
Мережний форум.....	26
Мережна гра «Рулетка»	28
Мережний сервер обліку помилок « <i>Bug tracking</i> »	29
Система термінального доступу.....	31
Електронний магазин	33
Система торгів	34
Система пошуку/публікації новин	35
Платіжна система.....	37
Служба каталогів x.500 (<i>LDAP</i>)	38
Система голосування/рейтингів	40
Система публікації й пошуку вакансій.....	42
Система оповіщення про події й передплати на них	44
Система виставляння курсів валют	46
Система розподілених математичних розрахунків	47
2. ВИВЧЕННЯ ПРИКЛАДНИХ ПРОТОКОЛІВ ТСП/IP.....	50
<i>Розробка прикладного протоколу</i>	<i>50</i>
ЛАБОРАТОРНА РАБОТА №2. Мережеві клієнти різних протоколів	55
Клієнт протоколу POP-3	55
Клієнт протоколу SMTP.....	56
Клієнт протоколу FTP, що працює із сервером в активному режимі	58
Клієнт протоколу FTP, що працює із сервером у пасивному режимі.	59
Збирач електронної пошти	61
Програма, що забезпечує завантаження всіх файлів FTP- каталогу.....	63
Програма – завантажник умісту WebСайтів	64
РОЗРОБКА СЕРВЕРУ ПРИКЛАДНОГО ПРОТОКОЛУ	65
Лабораторна робота №3. Прикладні сервери	69
Сервер протоколу POP-3	69
Сервер протоколу SMTP.....	70
Сервер протоколу FTP, що функціонує в активному режимі	71

Сервер протоколу FTP, що функціонує в пасивному режимі.....	73
Сервер протоколу HTTP	75
РОЗРОБКА ДОДАТКУ-ПОСЕРЕДНИКА ПРИКЛАДНОГО ПРОТОКОЛУ	76
Лабораторна робота №4. Проксі-сервер	77
Проксі-Сервер протоколу HTTP	77
Проксі-Сервер для протоколу FTP, що функціонує в активному режимі .	78
Проксі-Сервер для протоколу FTP, що функціонує в пасивному режимі .	79
Http-проксі-Сервер для протоколу FTP	81
Проксі-Сервер для протоколу POP-3	82
3. СИСТЕМНІ ПРОГРАМНІ ЗАСОБИ, ЩО ВИКОРИСТОВУЮТЬСЯ ПРИ	
ВІДЛАДЦІ МЕРЕЖЕВИХ ДОДАТКІВ.....	84
УТИЛІТА IFCONFIG	84
УТИЛІТА IPCONFIG	84
УТИЛІТА ARP	85
УТИЛІТА NETSTAT	86
УТИЛІТА ROUTE.....	86
УТИЛІТА PING.....	87
УТИЛІТА TRACEROUTE	87
ПРОГРАМА TELNET	88
СПИСОК ЛІТЕРАТУРИ	91

ВСТУП

Локальні мережі, Що бурхливо розвиваються останнім часом глобальні й, викликають постійне вдосконалювання надаваних мережних сірий- висів і поява нових розподілених додатків, функціонуючих у мережах TCP/IP. Мета даних методичних вказівок – навчити базовим підходам, що реалізують основні технології, використовувані при проектуванні мережних додатків.

Розробка мережного програмного забезпечення вимагає від програміста детальної реалізації всіх клієнтських і серверних функції указанного протоколу. Виконання цих функцій можна забезпечувати на різних рівнях деталізації:

- на рівні TCP/UDP сокетів;
- на рівні обробки команд прикладних протоколів;
- на рівні спеціалізованих прикладних бібліотек;

за допомогою різних розподілених об'єктних технологій. Реалізація функцій прикладних протоколів на рівні сокетів докладно описано в розділі 1.

Реалізація мережних додатків на рівні проробки команд прикладних протоколів викладено в розділі 2. Реалізація функцій прикладних протоколів за допомогою спеціалізованих бібліотек, а також за допомогою розподілених об'єктних технологій, таких як, CORBA, DCOM, RMI і інші, виходить за рамки даних методичних вказівок.

Для розробки й налагодження мережних додатків слід використовувати спеціальні утиліти й програми операційних систем. Перелік ключових утиліт операційних систем Linux і MS Windows і їх основні опції наведено в розділі 3.

1. ВИВЧЕННЯ ІНТЕРФЕЙСУ СОКЕТІВ

Розробка мережних додатків за допомогою технології сокетів.

З погляду архітектури TCP/IP сокетом називається пара (IP-адреса, порт), що однозначно ідентифікує прикладний додаток у мережі Internet.

З погляду операційної системи BsdCокет (Berkley Software Distribution) або просто сокет — це виділені операційною системою набір ресурсів, для організації мережної взаємодії. До таких ресурсів відносять, наприклад, буфера для приймання/посилки даних або черги повідомлень. Технологія сокетів була розроблена в університеті Берклі й уперше з'явилася в операційній системі BSD-UNIX.

В операційній системі MS Windows є аналогічна бібліотека мережної взаємодії Winsock, реалізована на основі бібліотеки BsdCокетів. Існує дві версії бібліотеки Winsock1 і Winsock2. У переважній більшості випадків функції й типи бібліотеки Winsock збігаються з функціями й типами BsdCокетів. Про наявні відмінності буде сказано окремо.

Створення сокета

Для створення сокета в бібліотеках Bsd-socket і Winsock є системний виклик socket:

```
int socket(int domain, int type, int protocol);
```

У випадку успіху результат виклику функції – дескриптор створеного сокета, у випадку помилки (-1) у бібліотеці Bsd-socket і INVA- LID_SOCKET у бібліотеці Winsock.

Параметр domain указує на домен, у просторі якого створюється даний сокет. Домен AF_UNIX використовується для міжпроцесної

взаємодії, домен AF_INET – для передачі з використанням стека протоколів TCP/IP.

Параметр type визначає тип створюваного сокета. Цей параметр може мати значення:

- SOCK_STREAM – для організації надійного каналу зв'язку з встановленням з'єднань
- SOCK_DGRAM – для організації ненадійного дейтаграмного каналу зв'язку
- SOCK_RAW – для організації низькорівневого доступу на основі «сирих» сокетів

Параметр protocol – ідентифікатор використовуваного протоколу. У більшості випадків протокол однозначно визначається типом створюваного сокета, і передане значення цього параметра – 0. Якщо ж це не так (наприклад, у випадку SOCK_RAW), то необхідно явно задавати ідентифікатор протоколу. Для його одержання є системні виклики getprotobyname і getprotobynumber, які розбирають файл /etc/protocols і одержують ідентифікатор мережного протоколу:

```
struct protoent* getprotobyname(const char *name);
```

```
struct protoent* getprotobynumber(int proto);
```

Ці функції заповнюють структуру protoent, поле p_proto яку слід використовувати в якості параметра protocol виклику socket:

```
struct protoent
{
    c      p_name;           ім'я протоколу з файлу protocols
    c      p_aliases        список псевдонімів
    in     p_proto;          ідентифікатор протоколу
}
```

Структура protoent описана у файлі /usr/include/netdb.h

Організація з'єднання

Для встановлення Тср З'єднання використовується виклик connect:

```
int connect(int s, const struct sockaddr* serv_addr,
int addr_len);
```

Результатом виконання функції є встановлення TCP- з'єднання з ТсрeСервером. Функція повертає значення 0 у випадку успіху й -1 у випадку помилки.

Параметр `s` – дескриптор створеного сокета.

Параметр `serv_addr` – покажчик на структуру, що містить параметри віддаленого вузла.

Параметр `addr_len` – розмір у байтах структури, на яку вказує параметр `serv_addr`.

При програмуванні сокетів з домена `AF_INET` замість структури `sockaddr` використовується структура, що приводиться до неї, `sockaddr_in`, що знаходяться у файлі, що підключається, `/usr/include/linux/in.h`:

```
struct sockaddr_in
{
    sa_family_t      sin_family;      // Комунікаційний домен
    unsigned short    sin_port;        // Номер порту
    struct in_addr     sin_addr;        // IP-адреса
    ...
};
```

Успіх виконання функції `connect` означає коректне встановлення логічного каналу зв'язку й можливість початку передачі й приймання даних по протоколу TCP.

У випадку використання виклику `connect` для протоколу UDP встановлення з'єднання не відбувається, а адреса й порт зі структури `serv_addr` використовується як адреса за замовчуванням для наступних викликів `send` і `recv`. Для більш простого заповнення параметрів структури `sockaddr_in` використовуються системні виклики `htons` і `inet_addr`, що здійснюють заміну порядку проходження байт у номері порту й переклад Ір Адреси зі строкового виду в числовий відповідно, наприклад:

```
serv_addr.sin_port = htons(3128);
```

```
serv_addr.sin_addr.s_addr = inet_addr("192.168.1.1");
```

Передача й приймання даних по протоколу **TCP**

Передача й приймання даних у рамках устанавленого TCP З'єднання здійснюється викликами `send` і `recv`:

```
int send(int s, const void *msg, size_t len, int flags);
```

```
int recv(int s, void *msg, size_t len, int flags);
```

Параметр `s` – дескриптор сокета, параметр `msg` – покажчик на буфер, що містить дані (виклик `send`), або покажчик на буфер, призначений для приймання даних (виклик `recv`). Параметр `len` – довжина буфера в байтах, параметр `flags` – опції посилки або приймання даних.

значення, що вертається, – число успішне посланих або прийнятих байтів, у випадку помилки функція повертає значення -1.

Передача й приймання даних по протоколу **UDP**

У випадку встановленого адреси за замовчуванням для протоколу UDP (ви- заклик `connect`) функції для передачі й приймання даних по протоколу UDP можна використовувати виклики `send` і `recv`.

Якщо адреса й порт за замовчуванням для протоколу UDP не встановлений, то параметри вилученої сторони необхідно вказувати або одержувати при кожному виклику операцій записи або читання. Для протоколу UDP є два аналогічні виклики `sendto` і `recvfrom`:

```
int sendto(int s, const void *buf, size_t len, int flags,
```

```
struct sockaddr *to, int*tolen);
```

```
int recvfrom(int s, void *buf, size_t len, int flags,
```

```
struct sockaddr *from, int*fromlen);
```

Параметри `s`, `buf`, `len` і `flags` мають той же зміст, що й у випадку використання функцій `send` і `recv`, параметри `to` і `tolen` – атрибути адреси віддаленого сокета при посилці даних, параметри `from` і `fromlen` – атрибути структури даних у яку містяться параметри віддаленого сокета при одержанні даних.

Завершення Тср З'єднання

Завершення встановленого ТсрЗ'єднання здійснюється в бсблsocketws Bsd-socket за допомогою виклику shutdown:

```
int shutdown(int s, int how);
```

Параметр s – дескриптор сокета, параметр how – визначає спосіб закриття:

- SHUT_RD – заборонене приймання даних
- SHUT_WR – заборонена передача даних
- SHUT_RDWR – заборонені й приймання й передача даних.

У бібліотеці Winsock семантика виклику трохи відрізняється:

```
int shutdown(SOCKET s, int how);
```

Параметр s – дескриптор сокета, параметр how – визначає спосіб закриття:

- SD_RECEIVE – заборонене приймання даних. У випадку наявності даних у черзі з'єднання розривається.

- SD_SEND – заборонена передача даних.
- SD_BOTH – заборонені й приймання й передача даних.

Закриття сокета

По закінченню роботи слід закрити сокет, для цього в бібліотеці Bsd-socket передбачений виклик close:

```
int close(int s);
```

Аналогічний виклик у бібліотеці Winsock має назва closesocket:

```
int closesocket(SOCKETs);
```

Параметр s – дескриптор сокета. значення, що вертається, – 0, у випадку успіху.

Прив'язування сокета

Створений сокет є об'єктом операційної системи, використовуючій її окремі ресурси. У той же час у більшості випадків не-досить просто виділити ресурси операційної системи, а слід також зв'язати ці ресурси з конкретними мережними параметрами: мережною адресою й номером порту. Особливо це важливо для серверних сокетів, для яких такий зв'язок – необхідна вимога

доступності розроблювального мережного сервісу.

Організація прив'язки створеного викликом `socket()` сокета до вказаного Ір Адреса і порта здійснюється за допомогою функції `bind`:

```
int bind(int s, struct sockaddr *addr, socklen_t addrlen);
```

Параметр `s` – дескриптор сокета, параметр `addr` задає покажчик на структуру, що зберігає параметри адреси й порту, `addrlen` – розмір структури `addr` у байтах.

Перевід ТсрСокета в стан прослуховування

Для переводу сокета в стан прослуховування служить системний виклик `listen`:

```
int listen(int s, int backlog);
```

Параметр `s` – дескриптор сокета, параметр `backlog` – задає максимальну довжину, до якої може рости черга з'єднань, що очікують.

У випадку успіху значення, що вертається, – 0. При помилці вертається -1.

Приймання вхідного Тср З'єднання

У випадку, коли сокет перебуває в стані прослуховування (`listen`) необхідно відслідковувати вступ вхідних з'єднань. Для цього мається системний виклик `accept`:

```
int accept(int s, struct sockaddr *addr, socklen_t *addrlen);
```

Параметр `s` – дескриптор, що прослуховує сокета, параметр `addr` – покажчик на структуру, що містить параметри сокета, що ініціює з'єднання, `addrlen` – розмір структури `addr` у байтах.

значення, що вертається, – дескриптор сокета, створеного для нового з'єднання. Більшість параметрів нового сокета відповідають параметрам слухаючого сокета. Отриманий сокет надалі може використовуватися для передачі й приймання даних.

У випадку якщо вхідних з'єднань нема, то функція `accept` очікує поступу запиту на вхідне з'єднання.

Додаткові функції

Для одержання текстової вистави помилки в операційній системі Linux використовується функція `strerror`:

```
char* strerror(int errnum);
```

Функція спостереження за зміною файлового дескриптора `select`:

```
int select(int n, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct
timeval *timeout);
```

Ця функція може використовуватися для визначення наявності інформації в прийомному сокеті.

Функції, призначені для установки й читання значень опцій сокетів `setsockopt` і `getsockopt`:

```
int setsockopt(int s, int level, int optname, const void *optval, socklen_t optlen);
```

```
int getsockopt(int s, int level, int optname, void *optval, socklen_t *optlen);
```

Параметр `s` – дескриптор сокета, `level` – рівень, на якому повинна відбуватися інтерпретація прапора, `optname` – ідентифікатор опції. Параметри `optval` і `optlen` використовуються у функції `setsockopt` для доступу до значень прапорів, для `getsockopt` вони задають буфер, у який потрібно помістити запитане значення.

Структура `TsrКлієнта`

Перераховані в попередніх параграфах функції використовуються для організації клієнтських і серверних додатків.

Клієнт протоколу TCP створює екземпляр сокету, необхідний для взаємодії із сервером, організує з'єднання, здійснює обмін даними, відповідно до протоколу прикладного рівня.

Типова структура TCP Клієнта представлена на мал. 1.1.

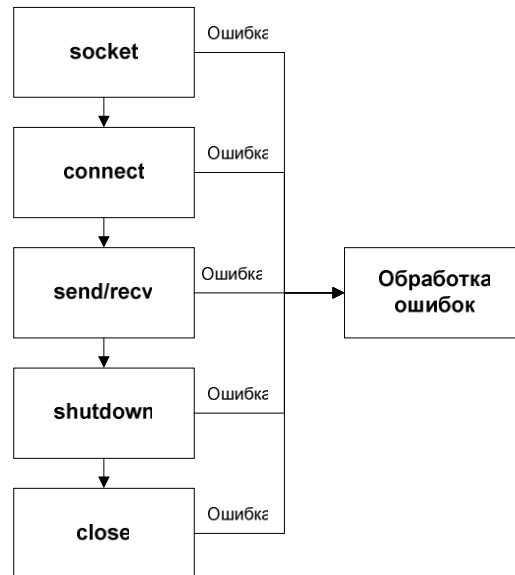


Рис. 1.1. Типова структура TCP Клієнта

Якщо ініціатором розриву з'єднання є клієнтський додаток (наприклад, канал даних протоколу FTP), то далі слід викликати функцію shutdown і після цього закрити сокет.

Кожний виклик функцій бібліотеки сокетів повинен супроводжуватися перевіркою на наявність помилкової ситуації й обробкою цієї ситуації.

Структура TCP Сервера

Організація TCP Сервера відрізняється від TCP Клієнта в першу чергу створенням слухаючого сокета (див. мал. 1.2 а). Такий сокет перебуває в стані listen і призначений тільки для приймання вхідних з'єднань. У випадку приходу запиту на з'єднання створюється додатковий сокет, який і займається обміном даними із клієнтом. Типова структура TCP Сервера й взаємозв'язок сокетів зображена на мал. 1.2 а) і б).

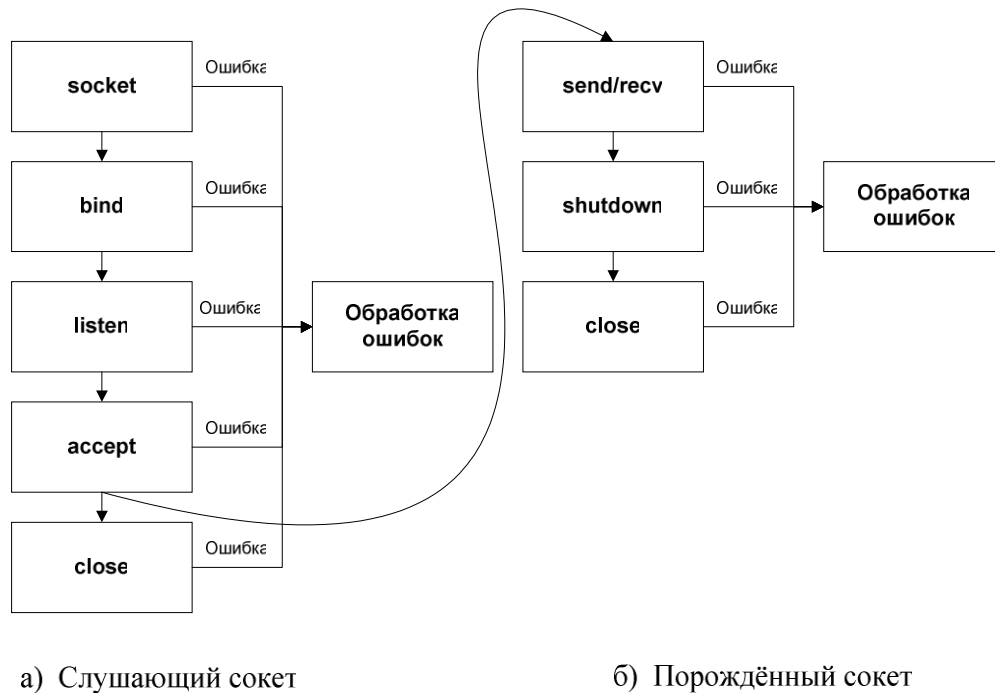
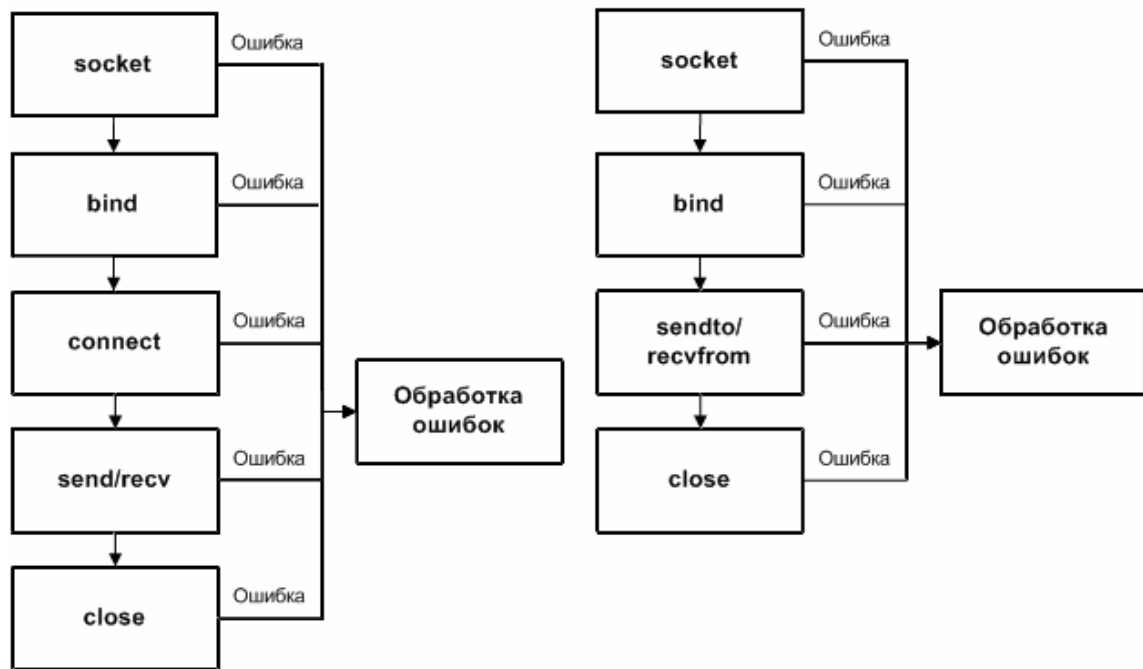


Рис. 1.2. Типова структура Тср Сервера

Структура Udp Клієнта

Структура UdpКлієнта ще більш проста, чому в ТсрКлієнта, тому що немає необхідності створювати й розривати з'єднання. Варіанти організації Udp Клієнта зображені на мал. 1.3.

Наявність двох варіантів організації пов'язане з можливістю в UDP-додатках використовувати виклик connect, що встановлює значення за замовчуванням для Ір Адреси й порту сервера.



а) С заданием адреса по умолчанию б) Без задания адреса по умолчанию

Рис. 1.3. Типова структура Udp Клієнта

Структура Udp Сервера

Через те, що в протоколі UDP не встановлюється логічний канал зв'язку між клієнтом і сервером, то для обміну даними між клієнтами й сервером немає необхідності використовувати з боку сервера декілька сокетів. Для визначення джерела отриманої дейтаграми серверний сокет може використовувати поля структури `from` виклику `recvfrom`. Типовий спосіб організації Udp Сервера наведений на мал. 1.4.

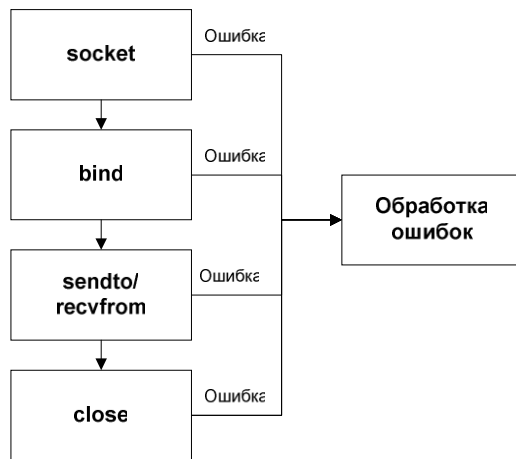


Рис. 1.4. Типова структура Udp Сервера

Лабораторна робота №1. Використання сокетів для рішення типових задач

Наведені нижче завдання є базовими для проведення першої частини лабораторних робіт з курсу «Технології комп'ютерних мереж». Кожне завдання може бути виконано з використанням одного із двох протоколів: TCP або UDP, а також з різним вибором операційних систем для програмі-клієнта й програми-сервера. Усі комбінації варіантів протоколів обміну й розподілу клієнт-серверних функцій наведені в табл. 1.

Таблиця 1. Варіанти завдань

№	Мережний протокол	Серверна операційна система	Клієнтська операційна система
1	TCP	Linux	Windows
2	TCP	Windows	Linux
3	UDP	Linux	Windows
4	UDP	Windows	Linux

Система обміну повідомленнями

Завдання: розробити додаток-клієнт і додаток -сервер, що забезпечують функції миттєвого обміну повідомлень між користувачами.

Основні можливості. Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові від клієнтів
- 3) Підтримка одночасної роботи декількох клієнтів через механізм

ниток

- 4) Передача текстового повідомлення одному клієнтові
- 5) Передача текстового повідомлення всім клієнтам
- 6) Приймання й ретрансляція вхідних повідомлень від клієнтів
- 7) Обробка запиту на відключення клієнта
- 8) Примусове відключення зазначеного клієнта

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Установлення з'єднання із сервером
- 2) Передача повідомлення всім клієнтам
- 3) Передача повідомлення зазначеному клієнтові
- 4) Приймання повідомлення від сервера з наступною індикацією
- 5) Розрив з'єднання
- 6) Обробка ситуації відключення клієнта сервером

Налаштування додатків. Розроблений клієнтський додаток повинний надавати користувачеві налаштування Ір Адреси або доменного імені сервера повідомлень і номера порту сервера.

Методика тестування. Для тестування додатків запускається один екземпляр серверного додатка й кілька екземплярів клієнтських. У процесі тестування перевіряються основні можливості додатків

по передачі і прийманню повідомлень, ситуації самостійного й примусового відключення клієнта.

Джерела: конспект лекцій [1], [2]

Система передачі файлів

Завдання: розробити додаток-клієнт і додаток-сервер, що забезпечують функції обміну файлами.

Основні можливості. Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові від клієнтів
- 3) Підтримка одночасної роботи декількох клієнтів через механізм

ниток

- 4) Приймання файлу від клієнта
- 5) Передача по запиту клієнта списку файлів поточного каталогу
- 6) Приймання запитів на передачу файлу й передача файлу клієнтові
- 7) Навігація по системі каталогів
- 8) Обробка запиту на відключення клієнта
- 9) Примусове відключення клієнта

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Установлення з'єднання із сервером
- 2) Одержання від сервера списку файлів каталогу
- 3) Операції навігації по системі каталогів
- 4) Передача файлу серверу
- 5) Приймання файлу від сервера
- 6) Розрив з'єднання
- 7) Обробка ситуації відключення клієнта сервером

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві настроювання Ір Адреси або доменного імені файлсервера й номера порту, використовуваного сервером.

Розроблений серверний додаток повинний надавати користувачу настроювання кореневого каталогу для клієнтських додатків.

Методика тестування. Для тестування додатків запускається файловий сервер і кілька клієнтів. У процесі тестування перевіряються основні можливості додатка по передачі файлів і навігації по системі каталогів.

Джерела: конспект лекцій [1], [2]

Система електронної пошти

Завдання: розробити додаток-клієнт і додаток-сервер електронної пошти.

Основні можливості. Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові від клієнтів
- 3) Підтримка одночасної роботи декількох поштових клієнтів через механізм ниток

- 4) Приймання поштового повідомлення від одного клієнта для іншого
- 5) Зберігання електронної пошти для клієнтів
- 6) Посилка клієнтові поштового повідомлення по запиту з наступним видаленням повідомлення

- 7) Посилка клієнтові відомостей про стан поштового ящика
- 8) Обробка запиту на відключення клієнта
- 9) Примусове відключення клієнта

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Установлення з'єднання із сервером

- 2) Передача електронного листа на сервер для іншого клієнта
- 3) Перевірка стану своєї поштової скриньки
- 4) Одержання конкретного листа із сервера
- 5) Розрив з'єднання
- 6) Обробка ситуації відключення клієнта сервером

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві настроювання IP адреси або доменного імені сервера електронної пошти, номера порту, використовуваного сервером, і ідентифікаційної інформації користувача.

Розроблений серверний додаток повинний надавати користувачу настроювання списку користувачів поштового сервера.

Методика тестування. Для тестування додатків запускається сервер електронної пошти й кілька клієнтів. У процесі тестування перевіряються основні можливості додатків по передачі й прийманню повідомлень.

Джерела: конспект лекцій [1], [2]

Інформаційна система

Завдання: розробити розподілену інформаційну систему, що складаються з додатка-сервера й додатка-клієнта. Інформаційна система є ієрархічним сховищем статей, кожна з яких складається з назви, автора й тексту статті. Інформаційна система повинна забезпечувати паралельний доступ до інформації декільком клієнтам.

Основні можливості. Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові від клієнтів

- 3) Підтримка одночасної роботи декількох клієнтів через механізм ниток
- 4) Зберігання ієрархічної структури статей інформаційної системи
- 5) Передача користувачеві списку поточних розділів системи, списку статей
- 6) Перехід у конкретний розділ системи по запиту клієнта
- 7) Повернення на попередній рівень по запиту клієнта
- 8) Передача користувачеві конкретної статті за назвою
- 9) Передача користувачеві всіх статей поточного розділу, приналежний певному авторові
- 10) Приймання від клієнта нової статті й збереження в інформаційній системі
- 11) Обробка запиту на відключення клієнта
- 12) Примусове відключення клієнта

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Установлення з'єднання із сервером
- 2) Одержання й печатка списку підрозділів і статей роздягнула
- 3) Передача команди на перехід у конкретний розділ
- 4) Передача команди на перехід у розділ на рівень вище
- 5) Одержання конкретної статті з інформаційної системи
- 6) Одержання статей конкретного автора
- 7) Посилка нової статті в систему
- 8) Розрив з'єднання
- 9) Обробка ситуації відключення клієнта сервером

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві можливість введення ідентифікаційної інформації, настроювання Ір Адреси або доменного імені, а також номера порту сервера інформаційної системи.

Розроблений серверний додаток повинний надавати користувачу можливість налаштування початкової крапки входу в інформаційну систему кожного користувача.

Методика тестування. Для тестування додатків запускається сервер інформаційної системи й кілька клієнтів. У процесі тестування перевіряються основні можливості додатків по передачі статей і навігації по розділах інформаційної системи.

Джерела: конспект лекцій [1], [2]

Калькулятор

Завдання: розробити додаток-сервер «Віддалений калькулятор», що дозволяє по запиту виконувати математичні операції, і віддалений клієнт для сервера.

Основні можливості. Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові від клієнтів
- 3) Підтримка одночасної роботи декількох клієнтів через механізм ниток
- 4) Приймання «швидких» операцій з аргументами від клієнта. Повинні підтримуватися наступні операції: додавання, вирахування, множення, ділення.
- 5) Обчислення «довгих» математичних операцій (факторіал, квадратний корінь) з наступною відкладеною посилкою результату клієнтові (окрема операція, що ініціюється сервером).
- 6) Обробка запиту на відключення клієнта
- 7) Примусове відключення клієнта

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Установлення з'єднання із сервером
- 2) Посилка операції з аргументами на обчислення
- 3) Одержання результату обчислень «швидких» операцій
- 4) Одержання результату обчислень «довгих» операцій
- 5) Розрив з'єднання
- 6) Обробка ситуації відключення клієнта сервером

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві настроювання IP адреси або доменного імені віддаленого калькулятора й номера порту, використовуваного сервером.

Розроблений серверний додаток повинний надавати користувачу настроювання часу виконання «довгих» операцій.

Методика тестування. Для тестування додатків запускається сервер «Віддаленого калькулятора» і кілька клієнтів. У процесі тестування перевіряються основні можливості калькулятора по миттєвому й відкладеному виконанню віддалених операцій.

Джерела: конспект лекцій [1], [2]

Система дистанційного тестування

Завдання: розробити клієнт-серверну систему дистанційного тестування знань, що полягає із централізованого сервера тестування й клієнтів тестування.

Основні можливості. Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові від клієнтів
- 3) Підтримка одночасної роботи декількох клієнтів через механізм ниток
- 4) Реєстрація клієнта, видача клієнтові результату його останнього теста, видача клієнтові списку тестів

- 5) Одержання від клієнта номера тесту
- 6) Послідовна видача клієнтові питань тесту й одержання от- тов на запитання
- 7) Після проходження тесту – видача клієнтові його результату
- 8) Обробка запиту на відключення клієнта
- 9) Примусове відключення клієнта

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Установлення з'єднання із сервером
- 2) Посилка реєстраційних даних клієнта
- 3) Вибір тесту
- 4) Послідовна видача відповідей на запитання сервера
- 5) Індикація результатів тесту
- 6) Розрив з'єднання
- 7) Обробка ситуації відключення клієнта сервером

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві настроювання Ір Адреси або доменного імені віддаленого сервера тестів і номера порту, використовуваного сервером. Розроблений серверний додаток повинний зберігати питання й правильні відповіді декількох тестів.

Методика тестування. Для тестування додатків запускається сервер «Віддаленого тестування» і кілька клієнтів. У процесі тестування перевіряються основні можливості сервера по паралельнім тестуванню декількох користувачів.

Джерела: конспект лекцій [1], [2]

Мережний форум

Завдання: розробити клієнт-серверну систему мережного форуму, ої складається із сервера форуму й користувацьких клієнтів.

Основні можливості. Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові від клієнтів
- 3) Підтримка одночасної роботи декількох клієнтів через механізм ниток

- 4) Реєстрація клієнта, що підключився
- 5) Видача клієнтові переліку нових повідомлень («постів») форуму
- 6) Видача клієнтові ієрархічної вистави форуму
- 7) Приймання від клієнта повідомлення у вітку форуму
- 8) Видача списку поточних активних користувачів форуму
- 9) Обробка запиту на відключення клієнта
- 10) Примусове відключення клієнта

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Установлення з'єднання із сервером
- 2) Посилка реєстраційних даних клієнта
- 3) Одержання й вивід переліку нових повідомлень
- 4) Одержання й вивід ієрархії форуму
- 5) Вибір поточної вітки форуму
- 6) Посилка повідомлення в поточну вітку форуму
- 7) Запит поточних активних користувачів форуму
- 8) Розрив з'єднання
- 9) Обробка ситуації відключення клієнта сервером

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві настроювання IP адреси або доменного імені віддаленого форумесервера й номера порту, використовуваного сервером.

Розроблений серверний додаток повинний зберігати ієрархію форуму, а також історію появи повідомлень у форумі.

Методика тестування. Для тестування додатків запускається сервер «Форум» і кілька клієнтів. У процесі тестування перевіряються основні можливості сервера по паралельній роботі на форумі декількох клієнтів.

Джерела: конспект лекцій [1], [2]

Мережна гра «Рулетка»

Завдання: розробити клієнт-серверну гру, що імітує мережну гру «Рулетка», що полягає з ігрового сервера й клієнтів.

Основні можливості. Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові від клієнтів
- 3) Підтримка одночасної роботи декількох клієнтів через механізм ниток
- 4) Реєстрація клієнта, що підключився, у якості круп'є (один) або в якості гравця (трохи)
- 5) Видача клієнтові (круп'є й гравцеві) списку ставок поточної гри (ста-інших гравців)
- 6) Одержання від клієнта ставки: суми ставки й типу ставки (чіт, не- пар, номер)
- 7) Одержання від круп'є команди на початок розіграшу.
- 8) Повідомлення всіх клієнтів про результат розіграшу
- 9) Обробка запиту на відключення клієнта
- 10) Примусове відключення клієнта

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Установлення з'єднання із сервером
- 2) Посилка реєстраційних даних клієнта (як круп'є або як иг- долі)
- 3) Одержання й вивід списку ставок
- 4) Для гравця: посилка своєї ставки (сума й ставки й тип ставки)
- 5) Для круп'є: посилка команди початку розіграшу
- 6) Одержання й вивід результатів розіграшу
- 7) Розрив з'єднання
- 8) Обробка ситуації відключення клієнта сервером

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві настроювання IP адреси або доменного імені віддаленого ігрового сервера й номера порту, використовуваного сервером.

Розроблений серверний додаток повинний зберігати пароль клієнта-круп'є.

Методика тестування. Для тестування додатків запускається ігровий сервер і кілька клієнтів. У процесі тестування перевіряються основні можливості сервера по паралельній роботі декількох клієнтів і одного круп'є. Перевіряється блокування входу в гру другого круп'є, спроби зробити повторну ставку й т.п.

Джерела: конспект лекцій [1], [2]

Мережний сервер обліку помилок «Bug tracking»

Завдання: розробити клієнт-серверну систему реєстрації, обліку й керування помилками в програмних проектах. Система повинна складатися із сервера, що зберігає репозиторій помилок і робочих клієнтських місць, що дозволяють програмістам і тестерам управляти помилками, що знайдені в програмних проектах.

Основні можливості. Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові від клієнтів
- 3) Підтримка одночасної роботи декількох клієнтів через механізм ниток
- 4) Реєстрація клієнта, що підключився, у якості тестера або розробника
- 5) Видача тестеру списку виправлених помилок
- 6) Видача тестеру списку активних (невиправлених) помилок
- 7) Приймання від тестера нової помилки із вказівкою розроблювача, проекту, ідентифікатора помилки, тексту помилки
- 8) Приймання від тестера команди про підтвердження або відхилення виправління помилки
- 9) Видача розроблювачеві списку знайдених у його проектах помилок: ідентифікаторів і текстів
- 10) Приймання команди розроблювача про виправлення помилки
- 11) Обробка запиту на відключення клієнта
- 12) Примусове відключення клієнта

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Установлення з'єднання із сервером
- 2) Посилка реєстраційних даних клієнта (як тестера або розробника)
- 3) Для тестера: одержання списку активних помилок
- 4) Для тестера: одержання списку виправлених помилок
- 5) Для тестера: додавання нової помилки
- 6) Для тестера: посилка команди підтвердження або відхилення виправлення активної помилки

- 7) Для розроблювача: одержання списку активних помилок
- 8) Для розроблювача: посилка команди виправлення активної помилки
- 9) Розрив з'єднання
- 10) Обробка ситуації відключення клієнта сервером

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві настроювання IP адреси або доменного імені віддаленого сервера обліку помилок і номера порту, використовуваного сервером.

Розроблений серверний додаток повинний зберігати списки тестерів і розроблювачів.

Методика тестування. Для тестування додатків запускається сервер обліку помилок і кілька клієнтів. У процесі тестування перевіряються основні можливості сервера по паралельній роботі декількох клієнтів-тестерів і клієнтів-розроблювачів. Перевіряється коректність зміни статусу помилки розроблювачем і тестером.

Джерела: конспект лекцій [1], [2]

Система термінального доступу

Завдання: розробити клієнт-серверну систему термінального доступу, що дозволяє клієнтам приєднувати до сервера й виконувати елементарні команди операційної системи.

Основні можливості. Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові від клієнтів
- 3) Підтримка одночасної роботи декількох термінальних клієнтів через механізм ниток
- 4) Проведення аутентифікації клієнта на основі отриманих імені користувача й пароля

5) Виконання команд користувача:

- ls – видача вмісту каталогу
- cd – зміна поточного каталогу
- who – видача списку зареєстрованих користувачів з указаним їм їхнього

поточного каталогу

- kill – Привілейована команда. Завершення сеансу іншого користувача.
- logout – вихід із системи

6) Примусове відключення клієнта

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Установлення з'єднання із сервером
- 2) Посилка аутентифікаційних даних клієнта (ім'я й пароль)
- 3) Посилка однієї з команд (ls, cd, who, kill, logout) серверу
- 4) Одержання відповіді від сервера
- 5) Розрив з'єднання
- 6) Обробка ситуації відключення клієнта сервером або іншим клієнтом

ентом

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві настроювання IP адреси або доменного імені віддаленого термінального сервера й номера порту, використовуваного сервером.

Розроблений серверний додаток повинний зберігати аутентифікаційні дані для всіх користувачів, а також списки дозволених кожному користувачеві команд.

Методика тестування. Для тестування додатків запускається термінальний сервер і кілька клієнтів. У процесі тестування перевіряються основні можливості сервера по паралельній роботі деяких клієнтів, що мають різні привілеї (списки дозволених ко-

манд). Перевіряється коректність виконання всіх команда в різних ситуаціях.

Електронний магазин

Завдання: розробити додаток–клієнт і додаток–сервер електронного магазину. Товар в електронному магазині має унікальний ідентифікатор, іменування, ціну.

Основні можливості. Серверний додаток повинний реалізовувати наступні функції:

- 7) Прослуховування певного порту
- 8) Обробка запитів на підключення по цьому портові від клієнтів електронного магазину
- 9) Підтримка одночасної роботи декількох клієнтів електронного магазину через механізм ниток
- 10) Приймання запитів на додавання або покупку товару
- 11) Здійснення додавання товару, облік кількості одиниць товару
- 12) Передача клієнтові електронного магазину інформації про товари й підтверджень про здійснення покупки
- 13) Обробка запиту на відключення клієнта
- 14) Примусове відключення клієнта

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Установлення з'єднання із сервером
- 2) Передача запитів про додавання, покупку товарів серверу
- 3) Одержання відповідей на запити від сервера
- 4) Розрив з'єднання
- 5) Обробка ситуації відключення клієнта сервером

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві настроювання IP–адреси або доменного

імені сервера електронного магазину й номера порту, використовуваного сервером.

Методика тестування. Для тестування додатків запускається сервер електронного магазину й кілька клієнтів. У процесі тустування перевіряються основні можливості додатків по передачі й прийманню повідомлень.

Джерела: конспект лекцій [1], [2]

Система торгів

Завдання: розробити додаток–клієнт і додаток–сервер системи торгів. На торги виставляються лоти, що мають початкову стои- мость. Учасники торгів можуть підвищувати вартість лота. Розпорядник може припинити торги. При закінченні торгів усім учасникам розсилаються результати торгів.

Основні можливості. Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові від клієнтів системи торгів (як розпорядник або учасник торгів)
- 3) Підтримка одночасної роботи декількох клієнтів системи торгів через механізм ниток
- 4) Від учасників торгів: приймання запитів на передачу списку лотів
- 5) Від учасників торгів: приймання запитів на підвищення вартості лота
- 6) Від розпорядника: приймання запиту на додавання нового лота з першонаочною вартістю
- 7) Від розпорядника: приймання запиту про закінчення торгів
- 8) Здійснення додавання лота, облік підвищення вартості лота учасниками, завершення торгів і розсилання результатів торгів усім учасникам

- 9) Обробка запиту на відключення клієнта
- 10) Примусове відключення клієнта

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Установлення з'єднання із сервером
- 2) Реєстрація як розпорядник або учасника
- 3) Учасник: передача запиту про вивід списку лотів
- 4) Учасник: передача запиту про підвищення вартості лота
- 5) Розпорядник: передача запиту про додавання лота
- 6) Розпорядник: передача запиту про припинення торгів
- 7) Одержання результатів торгів від сервера
- 8) Розрив з'єднання
- 9) Обробка ситуації відключення клієнта сервером

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві настроювання IP-адреси або доменного імені сервера системи торгів і номера порту, використовуваного сервером.

Методика тестування. Для тестування додатків запускається сервер системи торгів і кілька клієнтів. У процесі тестування перевіряються основні можливості додатків по передачі й прийманню повідомлень.

Джерела: конспект лекцій [1], [2]

Система пошуку/публікації новин

Завдання: розробити додаток-клієнт і додаток-сервер системи пошуку/публікації новин. Новини згруповані по темах. Кожна новина має унікальний ідентифікатор, назву й текст новини.

Основні можливості.

Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту

2) Обробка запитів на підключення по цьому портові від клієнтів системи пошуку/публікації новин

3) Підтримка одночасної роботи декількох клієнтів системи пошуку/публікації новин через механізм ниток

4) Приймання запитів від клієнта на передачу списку тем, списку новин по темі, тексту новини, додавання новини по темі

5) Здійснення додавання тем, новин по темах

6) Передача списків тем, списків новин і текстів новин

7) Обробка запиту на відключення клієнта

8) Примусове відключення клієнта

Клієнтський додаток повинний реалізовувати наступні функції:

1) Установлення з'єднання із сервером

2) Передача запитів на передачу списку тем, списку новин по темі, тексту новини, додавання новини по темі

3) Одержання результатів від сервера

4) Розрив з'єднання

5) Обробка ситуації відключення клієнта сервером

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві настроювання IP-адреси або доменного імені сервера системи пошуку/публікації новин і номера порту, що використовується сервером.

Методика тестування. Для тестування додатків запускається сервер системи пошуку/публікації новин і кілька клієнтів. У процесі тестування перевіряються основні можливості додатків по передачі й прийманню повідомлень.

Джерела: конспект лекцій [1], [2]

Платіжна система

Завдання: розробити додаток–клієнт і додаток–сервер платіжної системи. Учасники платіжної системи мають електронні гаманці. Електронний гаманець має унікальний номер. При реєстрації користувача в платіжній системі на його рахунок зараховується певна сума. Користувача платіжної системи можуть здійснювати платежі один одному через додаток–сервер.

Основні можливості. Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові від клієнтів платіжної системи
- 3) Підтримка одночасної роботи декількох клієнтів платіжної системи через механізм ниток
- 4) Приймання запитів від клієнта на реєстрацію користувача, передачу списку електронних гаманців користувачів платіжної системи, здійснення платежів одного користувача іншому, перевірка з-стояння рахунку гаманця
- 5) Здійснення додавання користувача в платіжну систему, зберігання й зміна стану електронних гаманців залежно від платежів користувачів
- 6) Передача запитів на платежі від одного користувача іншому, підтвердження платежів, номера нового гаманця при реєстрації користувача, списку електронних гаманців

7) Обробка запиту на відключення клієнта

8) Примусове відключення клієнта

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Установлення з'єднання із сервером

- 2) Передача запитів на передачу списку електронних гаманців користувачів платіжної системи, платежі одного користувача іншому, перевірку стану рахунку гаманця
- 3) Одержання від сервера запитів на платіж від іншого користувача, результатів платежу
- 4) Розрив з'єднання
- 5) Обробка ситуації відключення клієнта сервером

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві настроювання IP-адреси або доменного імені сервера платіжної системи й номера порту, використовуваного сервером.

Методика тестування. Для тестування додатків запускається сервер платіжної системи й кілька клієнтів. У процесі тестування перевіряються основні можливості додатків по передачі й прийманню повідомлень.

Джерела: конспект лекцій [1], [2]

Служба каталогів x.500 (LDAP)

Завдання: розробити додаток-клієнт і додаток-сервер служби каталогів. Усякий запис у каталозі складається з одного або декількох атрибутів і має унікальне ім'я. Унікальне ім'я складається з одного або декількох відносних унікальних імен, розділених комою (наприклад, “ cn=Users, dc=myserver, dc=myprovider, dc=ru ”). Відносне унікальне ім'я має вигляд *ім'яАтрибута=значення*. Значення атрибути відділяється від імені атрибути двокрапкою (наприклад, “ objectclass: top ”). На одному рівні каталогу не може існувати двох записів з однаковими відносними унікальними іменами. У силу такої структури унікального імені записи в каталозі можна легко перед- ставити у вигляді дерева. Перший рядок кожного блоку визначає

вузол пошуку. Від цього вузла проводиться пошук. Приклад трьох записів:

```
dn: cn=Users, dc=myserver, dc=myprovider, dc=ru
objectClass: top

dn: cn=ldapuser1, cn=Users, dc=myserver, dc=myprovider, dc=ru
objectClass: posixAccount
cn: LDAP User 1
uid: ldapuser1
uidNumber: 1001
gidNumber: 10
homeDirectory: /home/ldapuser1
loginShell: /bin/bash

dn: cn=ldapuser2, cn=Users, dc=myserver, dc=myprovider, dc=ru
objectClass: posixAccount
cn: LDAP User 2
uid: ldapuser2
uidNumber: 1002
gidNumber: 10
homeDirectory: /home/ldapuser2
loginShell: /bin/bash
```

Основні можливості. Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові від клієнтів
служби каталогів
- 3) Підтримка одночасної роботи декількох клієнтів служби каталогів
через механізм ниток
- 4) Приймання запитів на пошук, додавання й видалення записів
служби каталогів
- 5) Здійснення пошуку, додавання, видалення записів служби ката- балок

6) Передача клієнтові записів служби каталогів і підтверджень про вставку й видалення записів

7) Обробка запиту на відключення клієнта

8) Примусове відключення клієнта

Клієнтський додаток повинний реалізовувати наступні функції:

1) Установлення з'єднання із сервером

2) Передача запитів про пошук, додавання, видалення записів серверу

3) Одержання відповідей на запити від сервера

4) Розрив з'єднання

5) Обробка ситуації відключення клієнта сервером

Налаштування додатків. Розроблений клієнтський додаток повинний надавати користувачеві налаштування IP-адреси або доменного імені сервераслужбикаталогівіномерапорту,використовуваногосервером.

Методика тестування. Для тестування додатків запускається сервер служби каталогів і кілька клієнтів. У процесі тестування перевіряються основні можливості додатків по передачі й прийманню повідомлень.

Джерела: конспект лекцій [1], [2], [15]

Система голосування/рейтингів

Завдання: Розробити розподілену систему, що складається з додатків клієнта й сервера, для формування рейтингів на задану тему. Мається на увазі використання однорівневої системи рейтингів (без складної вкладеної ієрархії): є набір тем, у кожній з тем задана фіксована кількість альтернатив вибору, за кожен з яких можна віддати деяка кількість голосів. Інформаційна система повинна забезпечувати паралельну роботу декількох клієнтів.

Основні можливості:

Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові клієнтів
- 3) Підтримка одночасної роботи декількох клієнтів з використанням механізму ниток і засобів синхронізації доступу до розподільних між нитками ресурсів.
- 4) Примусове відключення конкретного клієнта
- 5) Додавання нової теми рейтингу (із вказівкою кількості альтернатив вибору)
- 6) Видалення теми рейтингу разом з усіма даними
- 7) Додавання альтернативи вибору в рейтинг (якщо кількість альтернатив менше заданого)
- 8) Відкриття/закриття голосування по темі
- 9) Облік голосів за конкретну альтернативу в конкретному рейтингу
- 10) Видача користувачеві списку наявних тем рейтингів.
- 11) Видача користувачеві стану конкретного рейтингу: закритий/відкритий, таблицю рейтингу в абсолютному й відносному виразі
- 12) * Збереження стану при вимиканні сервера

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Можливість паралельної роботи декількох клієнтів з один або різних ІрсАдрес
- 2) Установлення з'єднання із сервером (можливо, з реєстрацією на сервері)
- 3) Розрив з'єднання
- 4) Обробка ситуації відключення сервером
- 5) Одержання й вивід списку тем рейтингів

- 6) Одержання й вивід стану конкретного рейтингу
- 7) Передача команди на створення теми рейтингу
- 8) Передача команди на видалення теми рейтингу
- 9) Передача команди на додавання альтернативи в рейтинг
- 10) Передача команди на відкриття/закриття голосування по темі рейтингу
- 11) Подача декількох голосів за конкретну альтернативу конкретного рейтингу

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві можливість завдання IP адреси або доменного імені сервера, а також номера порту сервера.

Тестування. Для тестування запускається сервер рейтингової системи й кілька клієнтів. У процесі тестування перевіряються основні функціональні можливості розробленої рейтингової системи.

Джерела: конспект лекцій [1], [2]

Система публікації й пошуку вакансій

Завдання: Розробити розподілену систему, що полягає із додатків клієнта й сервера, для ведення бази вакансій. Мається на увазі використання однорівневої системи вакансій (без складної вкладеної ієрархії): є набір спеціальностей, у кожному з них можна подати вакансію. Крім подачі вакансій повинна здійснюватися можливість багатокритеріального пошуку по вакансіях. Інформаційна система повинна забезпечувати паралельну роботу декількох клієнтів.

Основні можливості:

Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові клієнтів

3) Підтримка одночасної роботи декількох клієнтів з використанням механізму ниток і засобів синхронізації доступу до розподілених між нитками ресурсам.

- 4) Примусове відключення конкретного клієнта
- 5) Додавання нової спеціальності
- 6) Видалення спеціальності разом з усіма поданими вакансіями
- 7) Додавання вакансії (ID, спеціальність, компанія, посада, меж віку, заробітна плата)
- 8) Видалення вакансії (по ID)
- 9) Видача користувачеві списку спеціальностей.
- 10) Видача користувачеві списку вакансій, що задовольняє за- просу на пошук (спеціальність, вік, посада, заробітна плата)
- 11) *Збереження стану при вимиканні сервера

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Можливість паралельної роботи декількох клієнтів з один або різних ІрсАдрес
- 2) Установлення з'єднання із сервером (можливо, з реєстрацією на сервері)
- 3) Розрив з'єднання
- 4) Обробка ситуації відключення сервером
- 5) Одержання й вивід списку спеціальностей
- 6) Передача команди на додавання спеціальності
- 7) Передача команди на видалення спеціальності
- 8) Передача команди на додавання вакансії
- 9) Одержання й вивід списку вакансій за умовами пошуку (при цьому необхідно організувати пошук по повному й неповному набору критеріїв)

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві можливість завдання IP адреси або доменного імені сервера, а також номера порту сервера.

Тестування. Для тестування запускається сервер системи керування вакансіями й кілька клієнтів. У процесі тестування перевіряються основні функціональні можливості розробленої інформаційної системи.

Джерела: конспект лекцій [1], [2]

Система публікацій та пошуку вакансій.

Завдання: Розробити розподілену систему, що полягає із додатків клієнта й сервера, для розподіленого оповіщення про події по принципу підписки. Мається на увазі разова або багаторазова генерація подій сервером (з деяким заданим інтервалом), з інформуванням підписаних клієнтів про ці події. Інформаційна система повинна забезпечувати паралельну роботу декількох клієнтів.

Основні можливості:

Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові клієнтів
- 3) Підтримка одночасної роботи декількох клієнтів з використанням механізму ниток і засобів синхронізації доступу до розподілених між нитками ресурсам.
- 4) Примусове відключення конкретного клієнта
- 5) Додавання нової події (однократного або багаторазового з ука-
занням періоду генерації)
- 6) Видалення події
- 7) Генерація багаторазових подій із заданим інтервалом по таймеру

- 8) Генерація однократних подій по команді від клієнта
- 9) Видача користувачеві списку наявних подій
- 10) Одержання й обробка команд підписки/зняття з передплати на події
- 11) Розсилання повідомлень про події клієнтам з фільтрацією за фактом підписки
- 12) * Збереження стану при вимиканні сервера – списку подій і розкладу багаторазових подій

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Можливість паралельної роботи декількох клієнтів з один або декількох ІрсАдрес
- 2) Установлення з'єднання із сервером (можливо, з реєстрацією на сервері)
- 3) Розрив з'єднання
- 4) Обробка ситуації відключення сервером
- 5) Одержання й вивід списку подій
- 6) Передача команди на додавання події (із вказівкою типу й періоду для багаторазових подій)
- 7) Передача команди на видалення події
- 8) Передача команди на генерацію однократної події
- 9) Передача команди передплати на подію
- 10) Передача команди скасування передплати на подію
- 11) Вивід реакції на настання подій, на які клієнт підписаний

Налаштування додатків. Розроблений клієнтський додаток повинний надавати користувачеві можливість завдання IP адреси або доменного імені сервера, а також номера порту сервера.

Тестування. Для тестування запускається сервер системи передплати на події й кілька клієнтів. У процесі тестування перевіряються основні функціональні можливості розробленої системи.

Джерела: конспект лекцій [1], [2]

Система виставлення курсів валют

Завдання: Розробити розподілену систему, що полягає із додатків клієнта й сервера, для розподіленого виставлення/перегляду курсів валют. Інформаційна система повинна забезпечувати паралельну роботу декількох клієнтів.

Основні можливості:

Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові клієнтів
- 3) Підтримка одночасної роботи декількох клієнтів з використанням механізму ниток і засобів синхронізації доступу до розподілених між нитками ресурсам.
- 4) Примусове відключення конкретного клієнта
- 5) Додавання нової валюти (коду валюти)
- 6) Видалення валюти
- 7) Додавання курсу конкретної валюти
- 8) Видача користувачеві списку наявних валют з поточними курсами й абсолютними/відносними збільшеннями до попередніх значенням
- 9) Видача користувачеві історії курсу конкретної валюти
- 10) * Збереження стану при вимиканні сервера

Клієнтський додаток повинний реалізовувати наступні функції:

- 1) Можливість паралельної роботи декількох клієнтів з один або декількох ІрсАдрес
- 2) Установлення з'єднання із сервером (можливо, з реєстрацією на сервері)
- 3) Розрив з'єднання
- 4) Обробка ситуації відключення сервером
- 5) Одержання й вивід списку валют з котируваннями/змінami
- 6) Передача команди на додавання валюти
- 7) Передача команди на видалення валюти
- 8) Передача команди на додавання курсу валюти
- 9) Одержання й вивід історії котирувань валюти

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві можливість завдання ІР адреси або доменного імені сервера, а також номера порту сервера.

Тестування. Для тестування запускається сервер системи котирувань валют і кілька клієнтів. У процесі тестування перевіряються основні функціональні можливості розробленої системи.

Джерела: конспект лекцій [1], [2]

Система розподілених математичних розрахунків

Завдання: Розробити розподілену систему, що складається із ~~додатків~~ клієнта й сервера, для розподіленого розрахунку простих чисел.

Інформаційна система повинна забезпечувати паралельну роботу декількох клієнтів.

Основні можливості:

Серверний додаток повинний реалізовувати наступні функції:

- 1) Прослуховування певного порту
- 2) Обробка запитів на підключення по цьому портові клієнтів

3) Підтримка одночасної роботи декількох клієнтів з використанням механізму ниток і засобів синхронізації доступу до розподілених між нитками ресурсам.

4) Примусове відключення конкретного клієнта

5) Зберігання розрахованих клієнтами простих чисел, а також поточної нижньої границі діапазону для нового запиту на розрахунок

6) Видача клієнтам максимального розрахованого простого числа

7) Видача клієнтам останніх N розрахованих простих чисел

8) Видача клієнтам необхідного діапазону розрахунку чисел

9) * Збереження стану при вимиканні сервера

Клієнтський додаток повинний реалізовувати наступні функції:

1) Можливість паралельної роботи декількох клієнтів з один або декількох ІрсАдрес

2) Установлення з'єднання із сервером (можливо, з реєстрацією на сервері)

3) Розрив з'єднання

4) Обробка ситуації відключення сервером

5) Одержання від сервера й вивід максимально розрахованого простого числа

6) Одержання від сервера й вивід останніх N розрахованих простих чисел

7) Одержання від сервера діапазону розрахунку простих чисел (нижню грань видає сервер, кількість чисел, що перевіряються, - клієнт)

8) Розрахунок простих чисел у необхідному діапазоні (має сенс перевірка залишки від розподілу на всі непарні числа в межах $\sqrt{N_{\max}}+1$)

9) Передача серверу набору розрахованих простих чисел

Настроювання додатків. Розроблений клієнтський додаток повинний надавати користувачеві можливість завдання IP адреси або доменного імені сервера, а також номера порту сервера.

Тестування. Для тестування запускається сервер системи розрахунку простих чисел і кілька клієнтів. У процесі тестування перевіряються основні функціональні можливості розробленої системи.

Джерела: конспект лекцій [1], [2]

2. ВИВЧЕННЯ ПРИКЛАДНИХ ПРОТОКОЛІВ TCP/IP

У першому розділі був представлений низькорівневий спосіб організації мережних додатків на основі програмування BSD сокетів. У реальних умовах не завжди необхідна настільки глибока деталізація опису, частіше потрібні більш високорівневі програмні інтерфейси, що спрощують проектування TCP- і UDP Додатків.

Даний розділ присвячений реалізації прикладних протоколів на рівні обробки команд прикладних протоколів. У якості механізму реалізації обробки команд використовується вільно розповсюджувана компонентна бібліотека Internet Direct версії 9 (Indy9) [5]. Ця бібліотека містить широкий набір компонентів для реалізації клієнтських і серверних додатків, що використовують протоколи TCP і UDP, і реалізуючих усі необхідні функції по підготовці, передачі, одержанні й обробці даних.

Розробка клієнта прикладного протоколу

Загальні положення

Для організації клієнтського програмного забезпечення бібліотека Internet Direct надає програмістові компонентів Tidtcpclient, інкапсулюючий у собі всі можливості, необхідні для побудови повноцінного мережного додатка-клієнта.

Установлення з'єднання

Бібліотека надає два способи організації клієнтського з'єднання: статичний і динамічний.

Статичний спосіб полягає в розміщенні під час розробки (в design time) на формі (спадкоємці класу TForm) або в модулі даних (насліднику класу Tdatamodule) екземпляра компонента Tidtcpclient. Па-

параметри встановлююмого з'єднання задаються статично властивостями компонента в інспекторі об'єктів (Object Inspector).

Динамічний спосіб створення з'єднання полягає в створенні під час виконання (в run time) екземпляра об'єкта Tdtcpclient. Параметри встановлююмого з'єднання задаються за допомогою виклику методів об'єкта й завдання значень відповідних властивостей.

Завдання параметрів з'єднання

Параметри встановлююмого з'єднання задаються за допомогою завдання властивостей компонента Tdtcpclient.

Для завдання параметрів віддаленого серверного додатка використовуються властивості Host і Port:

- властивість **Host** задає в текстовому виді IP-адреса або доменне ім'я вузла мережі, на якому функціонує серверний додаток.
- властивість **Port** визначає в TcpPort, використовуваний серверним додатком.

Для задання прив'язки клієнтського додатка до конкретних IP-адресам і Tcp Портів використовуються наступні властивості:

- **Boundip** – задає в текстовому виді IP-адреса використовувану клієнтським додатком. Порожній рядок позначає використання будь-якої адреси, наявної в даного мережного вузла.
- **Boundport** – задає номер Tcp-порту, виділюваний для клієнтського додатка. Значення 0 позначає використання номера з діапазону вільних непривілейованих портів, пропонованих операційною системою.
- **Boundportmax** – верхня границя діапазону непривілейованих портів, пропонованих операційною системою (у випадку, коли нульового значення властивості Boundip)

- **Boundportmin** – нижня границя діапазону непривілейованих портів, пропонуваніх операційною системою (у випадку, нульового значення властивості Boundip)

Установлення й розрив з'єднання

Для встановлення з'єднання з вилученим додатком використовується метод Connect:

```
procedure Connect(const Atimeout: Integer = Idtimeoutdefault);
```

Параметр – граничний час установлення з'єднання, по замовчуванні – час не обмежений.

Для розриву встановленого з'єднання використовується метод Disconnect:

```
procedure Disconnect;
```

Для встановлення з'єднання й одержання всіх даних із сокета (наприклад, при читанні файлу в протоколі FTP) використовується метод Connectandgetall:

```
function Connectandgetall: string;
```

Посилка Rfc Команд

Посилка команд прикладних протоколів (команд RFC) здійснюється викликом методу Sendcmd:

```
function Sendcmd(const Aout: string;
```

```
const Aresponse: Smallint = -1): Smallint;
```

```
function Sendcmd(const Aout: string;
```

```
const Aresponse: array of Smallint): Smallint;
```

Параметр Aout задає команду протоколу, параметр Aresponse – можлива відповідь або список відповідей сервера.

Одержання даних

Для читання даних компонентів має набір методів і властивостей, реалізуючих одержання типізованих і нетипізованих даних від сірій- вірної сторони:

1) для одержання цілочислених значень використовуються методи Read- Integer, Readsmallint, Readcardinal:

```
function Readinteger(const Aconvert: boolean = True): Integer; function Readcardinal(const
Aconvert: boolean = True): Cardinal; function Readsmallint(const Aconvert: boolean = True): Smallint;
```

2) Для одержання символьних і строкових значень використовуються методи Readchar, Readstring, Readln, Readlnwait і властивість Alldata:

```
function Readchar: Char;
function Readln(Aterminator: string = LF;
const Atimeout: Integer = Idtimeoutdefault; Amaxlinelength: Integer = -1): string;
function Readlnwait(Afailcount: Integer = Maxint): string; function
Readstring(const Abytes: Integer): string; property Alldata: string;
```

3) Для одержання буферів, потоків даних і списків рядків використовуються методи Readbuffer, Readstream і Readstrings:

```
procedure Readbuffer(var Abuffer; const Abytecount: Longint);
procedure Readstream(Astream: Tstream; Abytecount: Longint = -1;
const Areaduntildisconnect: boolean = False);
procedure Readstrings(Adest: Tstrings; Areadlinescount: Integer = -1);
```

Посилка даних

Для запису даних у сокет компонент має набір методів, реалізуючих посилку типізованих і нетипізованих даних серверній стороні:

1) для посилки цілочислених значень використовуються методи Writeinteger, Writesmallint і Writecardinal:

```
procedure Writeinteger(Avalue: Integer;
const Aconvert: Boolean = True);
procedure Writecardinal(Avalue: Cardinal;
const Aconvert: Boolean = True);
procedure Writesmallint(Avalue: Smallint;
const Aconvert: Boolean = True);
```

2) Для посилки символьних і строкових значень використовуються методи Writechar, Write і Writeln:

```
procedure Writechar(Avalue: Char); procedure Write(const
Aout: string); procedure Writeln(const Aout: string = ");
```

3) Для посилки буферів, потоків даних, файлів і списків рядків використовуються методи Writebuffer, Writestream, Writefile і Writestrings:

```
procedure Writebuffer(const Abuffer; Abytecount:Longint;
const Awritenow: Boolean = False);
procedure Writestream(Astream: Tstream;
const Aall: Boolean = True;
const Awritebytecount: Boolean = False;
const Asize: Integer = 0);
function Writefile(const Afile: string;
const Aenabletransferfile: Boolean =False):Cardinal;
procedure Writestrings(Avalue: Tstrings;
const Awritelinescount: Boolean = False);
```

4) Для посилки списків рядків у форматі RFC і стандартної відповіді RFC використовуються методи Writeheader, Writerfcstrings і Writerfcreply:

```
procedure Writeheader(Aheader: Tstrings); procedure
Writerfcstrings(Astrings: Tstrings); procedure Writerfcreply(Areply:
Tidrfcreply);
```

Обробка відповідей сервера

Обробка відповідей сервера полягає в аналізі результату посланої команди й в обробці виникаючих подій:

- 1) Подія Onconnected. Виникає після успішного встановлення з'єднання клієнтом.
- 2) Подія Ondisconnected виникає після розриву з'єднання.
- 3) Властивість Lastcmdresult містить текстовий і числовий результат виконання останньої команди.

Лабораторна робота №2.. Мережеві клієнти різних протоколів

Клієнт протоколу POP-3

Розробити додаток для операційних систем сімейства Windows або Linux, що забезпечує функції клієнта протоколу POP-3.

Основні можливості. Додаток повинний реалізовувати наступні функції:

- 1) Підключення до зазначеного сервера по ІреАдресі або доменному імені
- 2) Одержання стану ящика (кількість нових листів, їх сумарна довжина)
- 3) Одержання списку заголовків усіх нових листів сервера без попереднього завантаження
- 4) Завантаження всіх нових або конкретних обраних листів
- 5) Позначка конкретних листів для наступного видалення
- 6) Видалення позначених листів
- 7) Вихід з додатка без видалення позначених листів
- 8) Докладне протоколювання з'єднання сервера із клієнтом

Підтримувані команди. Розроблений додаток повинний реалізовувати наступні команди протоколу POP-3:

- USER – передача серверу ідентифікаційної інформації користувача
- PASS – передача серверу пароля користувача
- STAT – одержання стану поштової скриньки
- LIST – одержання списку повідомлення поштової скриньки
- RETR – одержання повідомлення
- DELE – позначка повідомлення на видалення
- TOP – одержання перших декількох рядків повідомлення
- UIDL – одержання унікального ідентифікатора повідомлення

- RSET – скидання всіх позначок на видалення повідомлень
- QUIT – видалення всіх позначених повідомлень і завершення сеансу

Налаштування додатка. Розроблений додаток повинний надавати користувачеві налаштування наступних параметрів:

- 1) Ір Адреса або доменне ім'я поштового сервера
- 2) Номер порту сервера (за замовчуванням - 110)
- 3) Ім'я користувача
- 4) Пароль користувача

Методика тестування. Для тестування додатка впливає використовувати поштові сервери, наявні в лабораторії, а також біс- платні поштові сервери, наявні в мережі Internet (<http://www.mail.ru>, <http://www.yandex.ru>, <http://www.rambler.ru> і т.п.).

Штатними клієнтами електронної пошти (Mozilla Thunderbird, MS Outlook Express, The Bat, Netscape Messenger) на сервер містяться тестові повідомлення. У процесі тестування перевіряються основні можливості додатка.

Джерела: конспект лекцій [1], [6], RFC [7].

Клієнт протоколу SMTP

Розробити додаток для операційних систем сімейства Windows або Linux, що забезпечує функції клієнта протоколу SMTP.

Основні можливості. Додаток повинний реалізовувати наступні функції:

- 1) Створення нового листа, що включає такі поля, як **From** (від-правитель), **To** (одержувач), **Subject** (тема), **Carbon copy** (додаткові адресати), **Blind copy** (додаткові сховані адресати), **Body** (текст)
- 2) Формування всіх необхідних заголовків листа, для того, щоб прийомна сторона не розглядала даний лист як спам.

3) Підключення до зазначеного SmtppСерверу й відсилання створеного листа

4) Докладне протоколювання з'єднання клієнта із сервером Підтримувані команди. Розроблений додаток повинний реалізувати наступні команди протоколу SMTP:

- HELO – передача серверу інформації про домені користувача
- MAIL FROM – передача серверу адреси відправника листа
- RCPT TO – передача серверу адреси одержувача листа
- DATA – передача серверу тіла листа
- QUIT – завершення сеансу зв'язки

Настроювання додатка. Розроблений додаток повинний забезпечувати настроювання наступних параметрів:

- 1) Власне доменне ім'я для передачі в команді HELO
- 2) Адреса відправника
- 3) Ір Адреса або доменне ім'я сервера SMTP

Методика тестування. Для тестування додатка впливає використовувати поштові сервери, наявні в лабораторії, а також біс- платні поштові сервери, наявні в мережі Internet (<http://www.mail.ru>,<http://www.yandex.ru>,<http://www.rambler.ru> і т.п.).

Засобами розробленого додатка здійснюється передача листа на зазначені ящики електронної пошти. Штатними клієнтами електронної пошти перевіряється коректність доставки пошти й правильність параметрів листа.

Джерела: конспект лекцій [1], [6], RFC ([8, 9, 10, 11])

Клієнт протоколу FTP, що працює із сервером в активному режимі

Розробити додаток для операційних систем сімейства Windows або Linux, що забезпечує функції клієнта протоколу FTP.

Основні можливості. Додаток повинний реалізовувати наступні функції:

- 1) Підключення до зазначеного сервера
- 2) Одержання списку файлів у каталозі
- 3) Навігація по системі каталогів
- 4) Копіювання файлу на сервер
- 5) Копіювання файлу із сервера
- 6) Створення й видалення каталогів
- 7) Видалення файлів
- 8) Забезпечення роботи з посиланнями на файли й каталоги
- 9) Підтримка передачі в бінарному й текстовому режимі
- 10) Протоколювання з'єднання сервера із клієнтом

Підтримувані команди. Розроблений додаток повинний реалізовувати наступні команди протоколу FTP:

- USER – передача серверу ідентифікаційної інформації користувача
- PASS – передача серверу пароля користувача
- CWD – зміна поточного каталогу сервера
- MKD – створення каталогу
- RMD – видалення каталогу
- DELE – видалення файла на сервері
- PORT – передача на сервер параметрів (адреси й порту) сокета,

здійснюючий приймання й передачу даних

- LIST – одержання списку файлів у поточному каталозі сервера в розширеному форматі

- NLST – одержання списку файлів у поточному каталозі сервера в спрощеному форматі

- RETR – одержання файлу із сервера

- STOR – посилка файлу на сервер

- TYPE – завдання режиму передачі даних

- QUIT – завершення сеансу

Настроювання додатка. Розроблений додаток повинний забезпечувати настроювання наступних параметрів:

- 1) IP-адреса або доменне ім'я файлового сервера
- 2) Ім'я користувача
- 3) Пароль користувача
- 4) Режим передачі: текстовий / бінарний

Методика тестування. Для тестування додатка впливає використовувати файлові сервери, наявні в лабораторії, а також відкриті файлові сервери, наявні в мережі Internet (ftp://ftp.funet.fi, ftp://ftp.relcom.ru і т.п.).

Для розроблених додатків перевіряються можливості під'єднання до сервера, копіювання файлу на сервер, копіювання файлу із сірий- віра, видалення файлу на сервері, перейменування файлу на сервері, створення каталогу, видалення каталогу.

Джерела: конспект лекцій [1], [6], RFC [12].

Клієнт протоколу FTP, що працює із сервером у пасивному режимі

Розробити додаток для операційних систем сімейства Windows або Linux, що забезпечує функції клієнта протоколу FTP.

Основні можливості. Додаток повинний реалізовувати наступні функції:

- 1) Підключення до зазначеного сервера
- 2) Одержання списку файлів у каталозі
- 3) Навігація по системі каталогів
- 4) Копіювання файлу на сервер
- 5) Копіювання файлу із сервера
- 6) Створення й видалення каталогів
- 7) Видалення файлів
- 8) Забезпечення роботи з посиланнями на файли й каталоги
- 9) Підтримка передачі в бінарному й текстовому режимі
- 10) Протоколювання з'єднання сервера із клієнтом

Підтримувані команди. Розроблений додаток повинний реалізовувати наступні команди протоколу FTP:

- USER – передача серверу ідентифікаційної інформації користувача
- PASS – передача серверу пароля користувача
- CWD – зміна поточного каталогу сервера
- MKD – створення каталогу
- RMD – видалення каталогу
- DELE – видалення файла на сервері
- PASV – перемикання сервера в пасивний режим
- LIST – одержання списку файлів у поточному каталозі сервера в розширеному форматі
- NLST – одержання списку файлів у поточному каталозі сервера в спрощеному форматі
- RETR – одержання файлу із сервера
- STOR – посилка файлу на сервер

•QUIT – завершення сеансу

Настроювання додатка. Розроблений додаток повинний забезпечувати настроювання наступних параметрів:

- 1) IP-адреса або доменне ім'я файлового сервера
- 2) Ім'я користувача
- 3) Пароль користувача
- 4) Режим передачі: текстовий / бінарний

Методика тестування. Для тестування додатка впливає використовувати файлові сервери, наявні в лабораторії, а також відкриті файлові сервери, наявні в мережі Internet (ftp://ftp.funet.fi, ftp://ftp.relcom.ru і т.п.).

Для розроблених додатків перевіряється можливість під'єднання до сервера, копіювання файлу на сервер, копіювання файлу із сервера, видалення файлу на сервері, перейменування файлу на сервері, створення каталогу, видалення каталогу.

Джерела: конспект лекцій [1], [6], RFC [12].

Збирач електронної пошти

Розробити додаток для операційних систем сімейства Windows або Linux, що забезпечує функції збирача пошти по протоколу POP-3 з декількох поштових серверів.

Основні можливості. Додаток повинний реалізовувати наступні функції:

- 1) Обробка запиту на підключення клієнта
- 2) Підключення до зазначених серверів
- 3) Одержання списку всіх нових листів усіх поштових серверів без їхнього попереднього завантаження
- 4) Завантаження всіх нових або конкретних обраних листів
- 5) Протоколювання з'єднання клієнта із серверами

Підтримувані команди. Розроблений додаток повинний реалізовувати наступні команди протоколу POP-3:

- USER – передача серверу ідентифікаційної інформації користувача
- PASS – передача серверу пароля користувача
- STAT – одержання стану поштової скриньки
- LIST – одержання списку повідомлення поштової скриньки
- RETR – одержання повідомлення
- TOP – одержання перших декількох рядків повідомлення
- UIDL – одержання унікального ідентифікатора повідомлення
- QUIT – видалення всіх позначених повідомлень і завершення сеансу

Настроювання додатка. Розроблений додаток повинний забезпечувати настроювання наступних параметрів:

- 1) IP адреси або доменні імена всіх поштових серверів
- 2) Номера портів усіх серверів (за замовчуванням - 110)
- 3) Аутентифікаційну інформацію користувача для всіх серверів (ім'я користувача й пароль)

Методика тестування. Для тестування додатка необхідно використовувати поштові сервери, наявні в лабораторії, а також безкоштовні поштові сервери, наявні в мережі Internet (<http://www.mail.ru>, <http://www.yandex.ru>, <http://www.rambler.ru> і т.п.).

Штатними клієнтами електронної пошти на сервери є тестові повідомлення. У процесі тестування перевіряються основні можливості додатка.

Джерела: конспект лекцій [1], [6], RFC [7].

Програма, що забезпечує завантаження всіх файлів FTP- каталогу

Розробити додаток для операційних систем сімейства Windows або Linux, що забезпечує функції одержання вмісту зазначеного каталогу сервера FTP і всіх його підкаталогів.

Основні можливості. Додаток повинний реалізовувати наступні функції:

- 1) Підключення до зазначеного Ftp-серверу
- 2) Вибір режиму роботи: активний або пасивний
- 3) Вибір каталогу для завантаження
- 4) Копіювання всіх файлів каталогу й рекурсивно всіх його підкаталогів із сервера
- 5) Забезпечення коректності роботи з посиланнями на файли й каталоги
- 6) Протоколювання з'єднання сервера із клієнтом

Підтримувані команди. Розроблений додаток повинний реалізовувати наступні команди протоколу FTP:

- USER – передача серверу ідентифікаційної інформації користувача
- PASS – передача серверу пароля користувача
- CWD – зміна поточного каталогу сервера
- PORT – передача на сервер параметрів (адреси й порту) сокета, здійснюючого приймання й передачу даних
- PASV – перемикання сервера в пасивний режим
- LIST – одержання списку файлів у поточному каталозі сервера в розширеному форматі
- NLST – одержання списку файлів у поточному каталозі сервера в спрощеному форматі
- RETR – одержання файлу із сервера

- STOR – посилка файлу на сервер
- QUIT – завершення сеансу

Настроювання додатка. Розроблений додаток повинний забезпечувати настроювання наступних параметрів:

- 1) IP-адреса або доменне ім'я файлового сервера
- 2) Ім'я користувача
- 3) Пароль користувача

Методика тестування. Для тестування додатка впливає використовувати файлові сервери, наявні в лабораторії, а також відкриті файлові сервери, наявні в мережі Internet (ftp://ftp.funet.fi, ftp://ftp.relcom.ru і т.п.). Для розроблених додатків перевіряються можливості приєднання до сервера, вибору необхідного каталогу, копіювання вмісту каталогу і його підкаталогів на комп'ютертклієнт, коректність роботи в активному й пасивному режимах.

Джерела: конспект лекцій [1], [6], RFC [12].

Програма – завантажник вмісту Web-сайтів

Розробити додаток для операційних систем сімейства Windows або Linux, що забезпечує функції одержання вмісту WebСайту.

Основні можливості. Додаток повинний реалізовувати наступні функції:

- 1) Підключення до зазначеного HttpServerу
- 2) Розбір коду HtmlpСторінок і одержання списку посилань на медіа- елементи й інші сторінки
- 3) Копіювання в локальний каталог заданого числа рівнів вкладеності сторінок Web Сайту
- 4) Протоколювання з'єднання сервера із клієнтом

Настроювання додатка. Розроблений додаток повинний забезпечувати настроювання наступних параметрів:

- 1) ім'я Web-сервера
- 2) глибину вкладеності сторінок, яку слід завантажити

Методика тестування. Для тестування додатка необхідно використовувати Web Сервери, наявні в мережі Internet. Для розробленого додатка перевіряються можливості приєднання до сервера, копіювання ресурсів, відповідних до заданого числа рівнів вкладеності комп'ютертклієнт.

Джерела: конспект лекцій [1], [6], RFC [13, 14].

Розробка сервера прикладного протоколу

Загальні положення

Для організації серверного програмного забезпечення бібліотека Internet Direct надає програмістові компонентів Tidtcpserver, що включає в себе всі можливості, необхідні для побудови повнофункційного багатониткового мережного серверного додатка.

Організація з'єднання

Бібліотека Internet Direct забезпечує можливість організації сірий- вірної служби за допомогою компонента Tidtcpserver. Для організації багатониткової мережної служби, що прослуховує певний порт, необхідно розмістити екземпляр компонента на формі (спадкоємці класу Tform) або в модулі даних (спадкоємці класу Tdatamodule). Ще один варіант – динамічне створення екземпляра класу Tidtcpserver під час виконання (run time).

Задання параметрів з'єднання

Параметри серверного з'єднання задаються за допомогою завдання властивостей компонента Tidtcpserver.

Для завдання параметрів прослуховування сокетів використовуються властивості **Bindings** і **Defaultport**:

- властивість **Bindings** задає список сокетів (пара Ір Адреса й Тср-порт), на яких сервер буде здійснювати прослуховування
- властивість **Defaultport** задає номер порту за замовчуванням, на якому буде проводитися прослуховування

Для визначення параметрів одночасної обробки вхідних з'єднань використовуються властивості **Maxconnections** і **Listenqueue**:

- властивість **Maxconnections** визначає максимальне число з'єднань, що обслуговуються сервером
- властивість **Listenqueue** задає максимальне число неопрацьованих з'єднань, які можуть перебувати в черзі запитів

Запуск і останів функції прослуховування здійснюється за допомогою булевої властивості **Active**, яке вказує на поточний стан сервера (запущений і прослуховує відповідний порт або зупинений).

Завдання списку оброблюваних команд

Сервер протоколу TCP звичайно будується на основі організації обробки RfcсmКоманд, заданих у специфікації протоколу в RFC. Кожній команді зі списку команд прикладного протоколу необхідно поставити у відповідність екземпляр об'єкта Tidcommandhandler, що перебуває в списку Commandhandlers компонента. Для роботи зі списками команд використовуються наступні властивості:

- булева властивість **Commandhandlersenabled** сигналізує про використання механізму оброблювачів команд у даному екземплярі компонента Tidtcpserver
- властивість **Commandhandlers** задає список оброблювачів команд, створених для цього сервера. Кожний оброблювач обслуговує одну RFC команду.

- властивість **Replyunknowncommand** задає RfctВідповідь на прихід невідомої команди

- подія **Onbeforecommandhandler** дозволяє визначити дії, які необхідно виконати безпосередньо перед викликом обробника команди

- подія **Onaftercommandhandler** дозволяє визначити дії, які необхідно виконати відразу після виклику оброблювача команд

- подія **Onnocommandhandler** визначає дії сервера в випадку приходу невідомої команди

У кожної створеної команди (об'єкт типу `Tidcommandhandler`) необхідно встановити групу властивостей, що визначають реакцію сервера на одержання цієї команди:

- булева властивість **Enabled** визначає, чи дозволене використання цієї команди в цей момент

- текстова властивість **Command** визначає синтаксис команди

- символна властивість **Cmddelimiter** задає код символу, що відокремлює команду від параметрів

- символна властивість **Paramdelimiter** задає код символу, відокремлюючого параметри друг від друга

- булева властивість **Disconnect** визначає чи необхідно розривати з'єднання після обробки даної команди

- оброблювач **Oncommand** визначає дії сервера у випадку приходу даної команди.

Обробка запитів клієнта

Крім механізмів обробки команд існують додаткові способи обробки запитів клієнта:

- подія **Onconnected** виникає після успішного встановлення з'єднання клієнтом.
- подія **Ondisconnected** виникає після розриву з'єднання
- подія **Onexecute** виникає при приході запиту від клієнта в слу- чаї, коли сервер не використовує механізм обробки RfcmКоманд (значення властивості Commandhandlersenabled встановлене в False).

Обмін даними із клієнтським додатком

Після успішного приєднання клієнтського додатка існує два способи обміну даними клієнта й сервера: з використанням механізму оброблювачів RfcmКоманд і без нього.

У випадку використання механізму оброблювачів RfcmКоманд уся процедура обміну даними реалізується в оброблювачі події Oncommand відповідного екземпляра Tidcommandhandler. У якості параметра передається об'єкт Asender типу Tidcommand, що містить необхідні властивості:

- властивість **Params** використовується для доступу до списку параметрів про- рабатываемой команди
- властивість **Thread** містить екземпляр нитки, створеної для обробки команди від даного клієнта.

Для одержання й посилки даних використовується об'єкт Connection: Asender.Thread.Connection. Цей об'єкт має всі необхідні методи, що реалізують функції обміну даними із клієнтським додатком.

У випадку відмови від використання механізму оброблювачів команд, увесь обмін забезпечується оброблювачем Onexecute, параметром якого є екземпляр нитки Thread. У цьому варіанті всі функції обміну здійснюються за допомогою об'єкта Thread.Connection аналогічно попередньому випадку.

Лабораторна робота №3. Прикладні сервери

Сервер протоколу POP-3

Завдання: розробити додаток для операційних систем сімейства

Windows або Linux, що забезпечує функції сервера протоколу POP-3.

Основні можливості. Додаток повинний реалізовувати наступні функції:

- 1) Зберігання ідентифікаційної й аутентифікаційної інформації декількох користувачів
- 2) Зберігання поштових папок вхідних повідомлень декількох користувачів
- 3) Обробка підключення клієнта
- 4) Видача стану ящика (кількість нових листів, їх сумарна довжина)
- 5) Видача списку всіх листів сервера з довжиною в байтах
- 6) Видача вмісту зазначеного листа
- 7) Позначка листа для наступного видалення
- 8) Видалення всіх позначених листів при розриві з'єднання
- 9) Протоколювання з'єднання сервера із клієнтом

Підтримувані команди. Розроблений додаток повинний реалізовувати наступні команди протоколу POP-3:

- USER – одержання від клієнта ідентифікаційної інформації користувача
- PASS – одержання від клієнта пароля користувача
- STAT – відправлення клієнтові стану поштової скриньки
- LIST – відправлення клієнтові списку повідомлення поштової скриньки
- RETR – відправлення клієнтові повідомлення
- DELE – позначка повідомлення на видалення
- TOP – відправлення клієнтові перших декількох рядків повідомлення

- UIDL – видача унікального ідентифікатора повідомлення
- RSET – скидання всіх позначок на видалення повідомлень
- QUIT – видалення всіх позначених повідомлень і завершення сеансу

Налаштування додатка. Розроблений додаток повинний забезпечувати:

- налаштування номера порту сервера (за замовчуванням - 110)
- створення, редагування й видалення користувачів поштового сервера
- налаштування аутентифікаційної інформації для користувачів

поштового сервера

Методика тестування. Для тестування додатка необхідно використовувати додатки-клієнти, установлені в лабораторії (Mozilla Thunderbird, The Bat, MS Outlook Express).

За допомогою наявних клієнтів електронної пошти здійснюється підключення до сервера з різної аутентифікаційної інформації. У процесі тестування перевіряються основні можливості сервера POP3.

Джерела: конспект лекцій [1], [6], RFC [7].

Сервер протоколуSMTP

Завдання: розробити додаток для операційних систем сімейства

Windows або Linux, що забезпечує функції сервера протоколу SMTP.

Основні можливості. Додаток повинний реалізовувати наступні функції:

- 1) Підтримку зберігання поштових папок користувачів для якого-або домена (наприклад, test.ru)
- 2) Одержання пошти для підтримуваного домена (test.ru)
- 3) Реалізація функції ретрансляції пошти для інших доменів. Для цього за допомогою DnsЗапиту необхідно визначити IP-адресу Smtп-сервера, відповідального за шуканий поштовий домен, (можна

використовувати компонент бібліотеки Internetdirect Tiddnsresolver)

і переслати по цій адресі необхідні повідомлення

4) Одночасна робота з декількома клієнтами

5) Протоколювання з'єднання сервера із клієнтом

Підтримувані команди. Розроблений додаток повинний реалізовувати наступні команди протоколу SMTP:

- HELO – одержання інформації про домене користувача
- MAIL FROM – одержання адреси відправника листа
- RCPT TO – одержання адресата листа
- DATA – одержання тіла листа
- QUIT – завершення сеансу зв'язки

Настроювання додатка. Розроблений додаток повинний забезпечувати:

- 1) завдання номер порту сервера (за замовчуванням - 25)
- 2) завдання імені підтримуваного поштового домена

Методика тестування. Для тестування додатка впливає використовувати додатканклієнти, установлені в лабораторії (Mozilla Thunderbird, The Bat, MS Outlook Express).

За допомогою наявних клієнтів електронної пошти здійснюється підключення до сервера й посилка листів на підтримуваний домен (на-приклад, test.ru) і на інші домени електронної пошти. У процесі тестування перевіряються функції зберігання «своїх» пошти й коректна відсылка поштових повідомлень іншим поштовим серверам.

Джерела: конспект лекцій [1], [6], RFC [8, 9, 10, 11].

Сервер протоколу FTP, що функціонує в активному режимі

Завдання: розробити додаток для операційних систем сімейства

Windows або Linux, що забезпечує функції Ftp-сервера.

Основні можливості. Додаток повинний реалізовувати наступні функції:

1) Зберігання ідентифікаційної й аутентифікаційної інформації декількох користувачів

2) Підтримка анонімного входу (користувач anonymous)

3) Обробка підключення клієнта

4) Видача по запиту клієнта вмісту каталогу

5) Навігація по системі каталогів

6) Створення нового каталогу

7) Видалення каталогу

8) Посилка по запиту клієнта вмісту зазначеного файлу

9) Приймання по запиту клієнта вмісту зазначеного файлу

10) Видалення зазначеного файлу

11) Протоколювання з'єднання сервера із клієнтом

Підтримувані команди. Розроблений додаток повинний реалізовувати наступні команди протоколу FTP:

•USER – одержання від клієнта ідентифікаційної інформації користувача

•PASS – одержання від клієнта пароля користувача

•LIST – відправлення клієнтові розширеної інформації про список файлів каталогу

•NLST – відправлення клієнтові скороченої інформації про список файлів каталогу

•CWD – зміна поточного каталогу сервера

•MKD – створення каталогу

•RMD – видалення каталогу

•DELE – видалення файлу на сервері

- PORT – одержання параметрів сокета клієнта (адреси й порту), що здійснює приймання й передачу даних

- RETR – посилка файлу клієнтові

- STOR – запис отриманого від клієнта файлу

- DELE – видалення файлу

- TYPE – завдання режиму передачі даних

- QUIT – видалення всіх позначених повідомлень і завершення сеансу

Настроювання додатка. Розроблений додаток повинний забезпечувати:

- 1) настроювання номерів порту сервера (за замовчуванням – 21 і 20)
- 2) настроювання кореневого каталогу сервера для кожного користувача

Методика тестування. Для тестування додатка необхідно використовувати стандартні Ftp-Клієнти, установлені в лабораторії (Mozilla Firefox, MS Explorer, Far, Total Commander).

За допомогою наявних клієнтів протоколу FTP здійснюється підключення до сервера з різною аутентифікаційною інформацією. У процесі тестування перевіряються основні можливості сервера по передачі, прийманню, видаленню файлів, навігації по файловій системі, функції по роботі з каталогами.

Джерела: конспект лекцій [1], [6], RFC [12].

Сервер протоколу FTP, що функціонує в пасивному режимі

Завдання: розробити додаток для операційних систем сімейства Windows або Linux, що забезпечує функції Ftp-сервера.

Основні можливості. Додаток повинний реалізовувати наступні функції:

- 1) Зберігання ідентифікаційної й аутентифікаційної інформації декількох користувачів

- 2) Підтримка анонімного входу (користувач anonymous)
- 3) Обробка підключення клієнта
- 4) Видача по запиту клієнта вмісту каталогу
- 5) Навігація по системі каталогів
- 6) Створення нового каталогу
- 7) Видалення каталогу
- 8) Посилка по запиту клієнта вмісту зазначеного файлу
- 9) Приймання по запиту клієнта вмісту зазначеного файлу
- 10) Видалення зазначеного файлу
- 11) Протоколювання з'єднання сервера із клієнтом

Підтримувані команди. Розроблений додаток повинний реалізовувати наступні команди протоколу FTP:

- USER – одержання від клієнта ідентифікаційної інформації користувача

- PASS – одержання від клієнта пароля користувача

- LIST – відправлення клієнтові розширеної інформації про список файлів каталогу

- NLST – відправлення клієнтові скороченої інформації про список фай- лов каталогу

- CWD – зміна поточного каталогу сервера

- MKD – створення каталогу

- RMD – видалення каталогу

- DELE – видалення файлу на сервері

- PASV – переклад сервера в пасивний режим

- RETR – посилка файлу клієнтові

- STOR – запис отриманого від клієнта файлу

- DELE – видалення файлу

- TYPE – завдання режиму передачі даних

- QUIT – видалення всіх позначених повідомлень і завершення сеансу

Налаштування додатка. Розроблений додаток повинний забезпечувати:

- 1) налаштування номера порту сервера (за замовчуванням – 21)
- 2) налаштування кореневого каталогу сервера для кожного користувача

Методика тестування. Для тестування додатка впливає використовувати стандартні Ftp-клієнти, установлені в лабораторії (Mozilla Firefox, MS Explorer, Far, Total Commander).

За допомогою наявних клієнтів протоколу FTP здійснюється підключення до сервера з різної аутентифікаційної інформацією. У процесі тестування перевіряються основні можливості сервера по передачі, прийманню, видаленню файлів, навігації по файловій системі, функції по роботі з каталогами.

Джерела: конспект лекцій [1], [6], RFC [12].

Сервер протоколу HTTP

Завдання: розробити додаток для операційних систем сімейства Windows або Linux, що забезпечує базові функції сервера протоколу HTTP (Web-сервера).

Основні можливості. Додаток повинний реалізовувати наступні функції:

- 1) Обробка підключення клієнта
- 2) Розбір рядка URL
- 3) Видача клієнтові запитаного ресурсу
- 4) Забезпечення паралельного завантаження клієнтові сторінок і медіа- елементів
- 5) Забезпечення паралельної роботи декількох клієнтів
- 6) Відображення параметрів, переданих разом з методом POST
- 7) Формування необхідних заголовків протоколу HTTP

8) Протоколювання з'єднання клієнта із сервером

Реалізовані методи протоколу:

- GET – для передачі WebaСторінок і медіа-елементів
- HEAD – для передачі заголовків WebaСторінок і медіа-елементів
- POST – для одержання від клієнта параметрів WebфФорм.

Настроювання додатка. Розроблювальний додаток повинний забезпечувати настроювання кореневого каталогу Web-сервера.

Методика тестування. У робочий каталог сервера помістити вміст якого-небудь Web-сайтів мережі Internet. Використовуваний сайт діл- дружин мати кілька рівнів вкладеності й містити медіа-елементи (наприклад, графічні зображення).

Для тестування додатка слід використовувати стандартні браузері, наявні в лабораторії (Mozilla Firefox, Internet Explorer, Opera). У процесі тестування перевіряються основні можливості сервера по передачі WebaСторінок, медіа-елементів, тестується коректність разбо- ра рядка URL, контролюються параметри, передані за допомогою ме- тода POST.

Джерела: конспект лекцій [1], [6], RFC [13, 14].

Розробка додатка-посередника прикладного протоколу

Загальні положення

Додатки-Посередники для прикладних протоколів призначені для організації опосередкованого доступу клієнтських додатків до уда- ленним серверним службам через проміжні додатки. Додатки-посередники застосовуються у випадках неможливості встановити пряме з'єднання між клієнтом і сервером і при необхідності реалізувати додаткові можливості, такі як кешування інформації, трансляція протоколів і т.п.

У випадку організації сервера-посередника (проксі-сервера) за допомогою бібліотеки Internet Direct слід використовувати пару компонентів Tdtpclient і Tdtpserver. Компонент TsrpСервер буде використовуватися для обслуговування клієнтів, що підключаються, а компонент TcriКлієнт для забезпечення з'єднання із прикладним сервером. Властиво додаток-посередник забезпечує трансляцію команд прикладних протоколів від клієнтів до сервера, і, при необхідності, перетворення транслюємих команд і даних.

Лабораторна робота №4. Проксі-сервер

Проксі-Сервер протоколу HTTP

Завдання: розробити додаток для операційних систем сімейства Windows або Linux, що забезпечує функції проксі-сервера для протокола HTTP.

Основні можливості. Додаток повинний реалізовувати наступні функції:

- 1) Обробка підключення клієнта
- 2) Одержання запитів від клієнта й перенапрямок їх серверу
- 3) Одержання від сервера файлів і кешування їх
- 4) Передача клієнтові відповіді від сервера або кешованого запиту
- 5) Забезпечення одночасної роботи декількох клієнтів
- 6) Протоколювання сеансу зв'язки клієнтом із сервером

Підтримувані методи протоколу:

- GET – для забезпечення завантаження WebaСторінок і медіа-елементів
- HEAD – для передачі заголовків WebaСторінок і медіа-елементів
- POST – для одержання від клієнта параметрів WebфФорм.

Настроювання додатка. Розроблений додаток повинний забезпечувати настрювання номера Тср-порту, використовуваного в проксі-сервером, обсягу кеша й часу зберігання інформації в кеші.

Методика тестування. Для перевірки працездатності додатку використовуються браузері, установлені в лабораторії (Mozilla Firefox, Internet Explorer, Opera, Netscape). У якості параметрів проксі- сервера встановлюється ІР-адреса й Тср-порт, зайняті розробленим додатком.

Перевіряється коректність завантаження різних Web-сайтів, наявних у мережі Internet. Особлива увага приділяється коректності паралельної загрузки медіа-елементів.

Джерела: конспект лекцій [1], [6], RFC [13, 14].

Проксі-Сервер для протоколу FTP, що функціонує в активном у режимі

Завдання: розробити додаток для операційних систем сімейства Windows або Linux, що забезпечує функції проксісервера для протоколу FTP. Проксі-Сервер повинен використовувати активний режим протоколу FTP.

Основні можливості. Додаток повинний реалізовувати наступні функції:

- 1) Обробка підключення клієнта
- 2) Розбір параметра username для визначення реального імені FTP-сервера
- 3) Переадресація всіх команд клієнта Ftp-серверу
- 4) Переадресація відповідей сервера клієнтові
- 5) Трансляція каналу даних, що відкривається сервером, клієнтові
- 6) Кешування даних, присланих сервером
- 7) Протоколювання з'єднання сервера із клієнтом

Підтримувані команди. Розроблений додаток повинний обробляти наступні команди протоколу FTP:

- USER – передача серверу ідентифікаційної інформації користувача разом з параметрами Ftp-сервера
- PASS – передача серверу пароля користувача
- PORT – передача на сервер параметрів (адреси й порту) сокета, здійснюючого приймання й передачу даних

Інші команди протоколу FTP повинні транслюватися FTP-серверу.

Настроювання додатка. Розроблений додаток повинний забезпечувати:

- 1) конфігурування номера порту, що прослуховується проксі-сервером
- 2) настроювання обсягу кешу

Методика тестування. Для тестування додатка потрібно використовувати стандартні Ftp-клієнти, установлені в лабораторії (Mozilla Firefox, MS Explorer, Far, Total Commander) і наявні в мережі Internet Ftp-сервери (<ftp://ftp.funet.fi>, <ftp://ftp.relcom.ru> і т.п.).

За допомогою наявних клієнтів протоколу FTP здійснюється підключення до проксі-серверу із вказівкою різних реальних FTP-серверів. У процесі тестування перевіряються основні можливості проксі-сервера по трансляції команд і даних, кешуванню інформації.

Проксі-Сервер для протоколу FTP, що функціонує в пасивном режимі

Завдання: розробити додаток для операційних систем сімейства

Windows або Linux, що забезпечує функції проксі-сервера для прото-

кола FTP. Додаток повинний використовувати пасивний режим протоколу FTP.

Основні можливості. Додаток повинний реалізовувати наступні функції:

- 1) Обробка підключення клієнта
- 2) Розбір параметра username для визначення реального імені FTP-сервера
- 3) Переадресація всіх команд клієнта Ftp-серверу
- 4) Переадресація відповідей сервера клієнтові
- 5) Трансляція каналу даних, що відкривається клієнтом, серверу
- 6) Кешування даних, присланих сервером
- 7) Протоколювання з'єднання сервера із клієнтом

Підтримувані команди. Розроблений додаток повинний реалізовувати наступні команди протоколу FTP:

- USER – передача серверу ідентифікаційної інформації користувача разом з параметрами Ftp-сервера
- PASS – передача серверу пароля користувача
- PASV – перемикання сервера в пасивний режим

Інші команди протоколу FTP повинні транслюватися FTP-серверу.

Налаштування додатка. Розроблений додаток повинний забезпечувати:

- 1) конфігурування номера порту, що прослуховується проксі-сервером
- 2) налаштування обсягу кешу

Методика тестування. Для тестування додатка потрібно використовувати стандартні Ftp Клієнти, установлені в лабораторії

(Mozilla Firefox, MS Explorer, Far, Total Commander) і наявні в мережі Internet Ftp-сервери (<ftp://ftp.funet.fi>, <ftp://ftp.relcom.ru> і т.п.).

За допомогою наявних клієнтів протоколу FTP здійснюється підключення до проксі-серверу із вказівкою різних реальних FTP-серверів. У процесі тестування перевіряються основні можливості проксі-сервера по трансляції команд і даних, кешуванню інформації.

Джерела: конспект лекцій [1], [6], RFC [12, 13, 14].

Http-проксі-сервер для протоколу FTP

Завдання: розробити додаток для операційних систем сімейства Windows або Linux, що забезпечує функції HttpПроксіСервера для протоколу FTP.

Основні можливості:

- 1) Підключення клієнта по протоколу HTTP
- 2) Перетворення HttpЗапитів в Ftp-команди
- 3) Трансляція аутентифікаційної інформації на FtpСервер
- 4) Підтримка режиму анонімного користувача
- 5) Протоколювання всього обміну між клієнтом і сервером FTP

Підтримувані методи протоколу **HTTP** і команди протоколу **FTP**:

- GET – одержання файлу або каталогу Ftp-сервера. Перетвориться в команди RETR, LIST або NLST протоколу FTP.

- PUT – запис файлу. Перетвориться в команду STOR протоколу FTP

Інші команди протоколу FTP реалізуються по необхідності. Методика тестування. Для тестування додатка потрібно використовувати стандартні Ftp-клієнти, що підтримують механізм HTTP-проксі для протоколу FTP, установлені в лабораторії (Mozilla Firefox,

MS Explorer) і наявні в мережі Internet Ftp-сервери (<ftp://ftp.funet.fi>, <ftp://ftp.relcom.ru> і т.п.).

За допомогою наявних клієнтів протоколу FTP здійснюється підключення до проксі-серверу із вказівкою різних реальних FTP-серверів. У процесі тестування перевіряються основні можливості проксі-сервера по трансляції команд і даних протоколу FTP в HTTP і навпаки, по кешуванню інформації.

Джерела: конспект лекцій [1], [6], RFC [12, 13, 14]

Проксі-Сервер для протоколу POP-3

Завдання: розробити серверспосередник для протоколу POP-3, функціонуючий під керуванням операційних систем Windows або Linux.

Основні можливості додатка:

- 1) Підключення декількох поштових клієнтів
- 2) Розбір аутентифікаційних даних і одержання імені POP3-сервера
- 3) Трансляція команд від клієнта до сервера й назад
- 4) Протоколювання обміну між поштовим клієнтом і сервером

Підтримувані команди. Розроблений додаток повинний реалізовувати наступні команди протоколу POP-3:

- USER – одержання від клієнта ідентифікаційних даних користувача разом з адресою реального сервера

- PASS – одержання від клієнта пароля користувача
- STAT – відправлення клієнтові стану поштової скриньки
- LIST – відправлення клієнтові списку повідомлення поштової скриньки
- RETR – відправлення клієнтові тексту повідомлення
- DELE – позначка повідомлення на видалення
- TOP – відправлення клієнтові перших декількох рядків повідомлення
- UIDL – видача унікального ідентифікатора повідомлення

- RSET – скидання всіх позначок на видалення повідомлень
- QUIT – видалення всіх позначених повідомлень і завершення сеансу

Методика тестування. Для тестування додатка використовується стандартний клієнт електронної пошти, наявний у лабораторії (Mozilla Thunderbird, The Bat, MS Outlook Express). Параметри облікової записи настраюються на сокет, використовуваний розробленим додатком. У якості поштового сервера можна використовувати лабораторні POP-3- сервери або безкоштовні поштові служби (наприклад, www.mail.ru, www.yandex.ru і т.п.). Спроектований проксі-сервер перевіряється на виконання набору основних функцій протоколу POP3.

Джерела: конспект лекцій [1], [6], RFC [7].

1. СИСТЕМНІ ПРОГРАМНІ ЗАСОБИ, ЩО ВИКОРИСТОВУЮТЬСЯ ПРИ НАЛАГОДЖЕННІ МЕРЕЖНИХ ДОДАТКІВ

Для налагодження розроблювальних мережних додатків можна використати набір утиліт операційних систем Linux [2, 3] і MS Windows [4], що дозволяють проводити контроль стану й налаштування мережних компонентів.

Утиліта **ifconfig**

Утиліта `ifconfig` операційної системи Linux призначена для відображення або модифікації всіх параметрів мережного інтерфейсу.

Основні форми команди:

- `ifconfig` – відобразити інформацію про всі інтерфейси;
- `ifconfig <interface>` – відобразити інформацію про конкретний інтерфейс;
- `ifconfig <interface> <опції>` – установити конфігураційні параметри для інтерфейсу.

Утиліта `ifconfig` дозволяє встановити великий набір параметрів мережного інтерфейсу, підтримує безліч мережних архітектур, дозволяє зупиняти й запускати мережні інтерфейси. В операційній системі Linux звичайно не потрібно безпосередньо використовувати утиліту `ifconfig` для конфігурування інтерфейсів, у більшості випадків для цього існують більш високорівневі засоби: файли сценаріїв, ініціалізаційні файли, програми-конфігуратори.

Утиліта **ipconfig**

Аналогом утиліти `ifconfig` в операційній системі MS Windows XP є утиліта `ipconfig`. На відміну від `ifconfig` основними функціями утиліти `ipconfig` є:

- відображення поточних мережних установок;
- відображення кешу DnsResolver (Dns-resolver) і очищення його;

- операції з орендою Ipv4Адрес, отриманих через DnsServer.

Установка параметрів мережних інтерфейсів даною утилітою не передбачена.

Основні форми команди:

- `ipconfig` – відображення базових мережних параметрів інтерфейсів;
- `ipconfig /all` – відображення всіх мережних параметрів інтерфейсів, а також загальної інформації про мережні налаштування стека TCP/IP;
- `ipconfig /displaydns` – відображення поточного стану кешу DNS-перетворювача;
- `ipconfig /flushdns` – очищення кешу DnsResolver;
- `ipconfig /release` – звільнення отриманого в оренду в DHCP-сервера IP адреси;
- `ipconfig /renew` – поновлення оренди IP адреси.

Утиліта **arp**

Утиліта призначена для взаємодії з таблицею ARP (Address Resolution Protocol), яка зберігає відповідність адрес мережного й канального рівня.

Основні форми команди:

- `arp` – відобразити вміст АрпТаблиці (тільки для ОС Linux);
- `arp -a` – відобразити вміст АрпТаблиці (для всіх ОС);
- `arp -s <IP> <MAC>` – додати статичний запис в АрпТаблицю;
- `arp -d <IP>` – вилучити запис із АрпТаблиці.

Утиліта **netstat**

Призначена для відображення інформації про статистику мережних протоколів мережних з'єднаннях, що течуть, стані сокетів, мережних інтерфейсах, таблиці маршрутизації.

Основні форми команди:

- `netstat` – відображення інформації про активні мережні з'єднання (Linux і Windows) і стані сокетів Unix Домена (тільки в Linux);
- `netstat -a` – відображення інформації про активні мережні з'єднання, що й прослуховують сокети;
- `netstat -r` – відображення таблиці маршрутизації;
- `netstat -i` – відображення таблиці інтерфейсів (тільки Linux).

Додатково в ОС Linux утиліта `netstat` має опцію “-p”, яка відображає процес, що володіє сокетом.

Утиліта **route**

Призначена для відображення й модифікації таблиці маршрутизації операційної системи.

Основні форми команди для ОС Linux:

- `route` – вивід таблиці маршрутизації;
- `route add <...>` – додавання маршруту;
- `route del <...>` – видалення маршруту;
- `route flush` – очищення таблиці

маршрутизації. Основні форми команди для ОС MS Windows XP:

- `route print` – вивід таблиці маршрутизації;
- `route add <...>` – додавання маршруту;
- `route change <...>` – зміна маршруту;
- `route delete <...>` – видалення маршруту;

- route -f – очищення таблиці маршрутизації;
- route -p add <...>– додавання постійного маршруту.

Утиліта **ping**

Утиліта ping (**packet internet groper**) є одним з основних засобів первинного тестування мережного середовища й використовується:

- для перевірки працездатності віддаленого вузла;
- для тестування тимчасових параметрів маршруту;
- для перегляду маршруту проходження

пакетів. Формат виклику: ping <опції> <адреса>.

Основні форми утиліти в операційній системі Linux:

- ping -z N <адреса>— посилка N тестових пакетів;
- ping -S N <адреса> — установка розміру пакетів, що посилають;
- ping -R <адреса> — роздруківка перших 9ти кроків

маршруту. Основні форми утиліти в операційній системі MS Windows:

- ping -n N <адреса>— посилка N тестових пакетів;
- ping -l N <адреса> — установка розміру пакетів, що посилають;
- ping -r N <адреса> — роздруківка перших N кроків маршруту.

Утиліта **tracert**

Утиліта tracert (в MS Windows – tracert) призначена для відображення шляхів проходження пакета від одного вузла до іншого. Для побудови маршруту використовується ланцюжок тестових Udp Дейтаграм або ICMP- пакетів. Варіант утиліти для операційної системи MS Windows завжди використовує IcmpaПакети. В операційній системі Linux за замовчуванням використовуються Udp Дейтаграми, використання Icmp Пакетів включається за допомогою опції -I.

Формат утиліти: tracert <опції> <адреса>.

Програма **telnet**

Програма telnet (або її аналоги) може використовуватися як універсальний клієнт для Тср Додатків. Імітація переданих даних здійснюється користувачем за допомогою клавіатурного введення, відповіді ТсрвСервера відображаються в текстовому виді у вікні програми telnet. Програма telnet входить до складу всіх сучасних операційних систем.

СПИСОК ЛІТЕРАТУРИ

1. О.О.Степаненко. Конспект лекцій з курсу «Технології розробки мережевих додатків».
2. Й. Снейдер. Эффективное программирование TCP/IP. Библиотека программиста—СПб: Питер, 2001. — 320 с: ил.
3. Э. Немец, Г. Снайдер, С. Сибасс, Т. Хейн. UNIX: Руководство системного администратора: Пер. с англ. — К.:BHV, 1996 — 832 с.
4. Титтел Э., Хадсон К., Стюарт М. TCP/IP. Сертификационный экзамен — экстерном. — СПб: Издательство «Питер», 1999. — 416 с.: ил.
5. Библиотека InternetDirect. <http://www.indyproject.org/>
6. Золотов С. Протоколы Internet – СПб.: BHV – Санкт-Петербург, 1998. 304 с., ил.
7. RFC 1939. Post Office Protocol - Version 3. J. Myers, M. Rose. May 1996.
8. RFC 0821. Simple Mail Transfer Protocol. J. Postel. Aug-01-1982
9. RFC 0822. Standard for the format of ARPA Internet text messages. D. Crocker. Aug-13-1982.
10. RFC 2821. Simple Mail Transfer Protocol. J. Klensin, Ed.. April 2001.
11. RFC 2822. Internet Message Format. P. Resnick, Ed.. April 2001.
12. RFC 0959. File Transfer Protocol. J. Postel, J.K. Reynolds. Oct-01-1985.
13. RFC 1945 Hypertext Transfer Protocol -- HTTP/1.0. T. Berners-Lee, R. Fielding, H. Frystyk. May 1996.
14. RFC 2616 Hypertext Transfer Protocol -- HTTP/1.1. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. June 1999.
15. <http://ru.wikipedia.org/wiki/LDAP>