Co-funded by the
Erasmus+ Programme
of the European Union

GAMEHUB University Enterprises Cooperation In Game Industry In Ukraine

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРНЕТИКИ ТА СИСТЕМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ І ТЕХНОЛОГІЙ

КИРИЙЧУК Д.Л.

ЕЛЕКТРОННИЙ НАВЧАЛЬНИЙ ПОСІБНИК

«Розробка мережевих комп’ютерних ігор мовою Java»

Для підготовки студентів
спеціальності 121 «Інженерія програмного забезпечення»

ХЕРСОН-2018

Co-funded by the
Erasmus+ Programme
of the European Union

GAMEHUB University Enterprises Cooperation In Game Industry In Ukraine

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРНЕТИКИ ТА СИСТЕМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ І ТЕХНОЛОГІЙ

ЕЛЕКТРОННИЙ НАВЧАЛЬНИЙ ПОСІБНИК

«Розробка мережевих комп'ютерних ігор мовою Java»

Для підготовки студентів
спеціальності 121 «Інженерія програмного забезпечення»

GAMEHUB: «СПІВРОБІТНИЦТВО МІЖ УНІВЕРСИТЕТАМИ ТА ПІДПРИЄМСТВАМИ В СФЕРІ ІГРОВОЇ ІНДУСТРІЇ В УКРАЇНІ»

GAMEHUB: «UNIVERSITY-ENTERPRISES COOPERATION IN GAME INDUSTRY IN UKRAINE»
561728-EPP-1-2015-1- ES-EPPKA2-CBHE-JP

The handbook were performed with support of the Erasmus+ Programme of the European Union (561728-EPP-1-2015-1- ES-EPPKA2-CBHE-JP). The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

ХЕРСОН-2018

Co-funded by the Erasmus+ Programme of the European Union

GAMEHUB University Enterprises Cooperation In Game Industry In Ukraine

УДК 004.4
Р 65

Розробка мережевих комп'ютерних ігор мовою Java: електронний навчальний посібник для підготовки студентів спеціальності 121 «Інженерія програмного забезпечення» / Укладач: Д. Л. Кирийчук. – Херсон: видавництво ФОП Вишемирський В.С., 2018. – 51 с.

Р 65

ISBN 978-617-7573-91-2 (електронне видання)

Укладачі:
Кирийчук Д.Л., к.т.н., доцент кафедри Програмних засобів і технологій.

Рецензенти:

Шарко О.В., д.т.н, професор кафедри транспортних технологій Херсонської державної морської академії.

Львов М.С., д.ф-м.н, професор, завідувач кафедри інформатики, програмної інженерії та економічної кібернетики Херсонського державного університету

Розглянуто на засіданні кафедри Програмних засобів і технологій,
Протокол №11 від 23 травня 2017 року.

Розглянуто на засіданні Вченої ради Херсонського національного технічного університету,
Протокол №9 від 2 червня 2017 року.

УДК 004.4

ISBN 978-617-7573-91-2

Зміст

1 Вступ

Предметом навчальної дисципліни є мережеві технології та протоколи передачі даних.

Мета дисципліни

Метою викладання навчальної дисципліни «Мережеві технології» є систематичне викладення базових понять та принципів побудови, налаштування і оптимізації комп'ютерних мереж, формування у студентів уявлення про теоретичні основи передачі даних, технології побудови сучасних комп'ютерних мереж. Набуття практичних навичок роботи з компонентами мережної інфраструктури, методи, що застосовуються при розробці комп'ютерних ігор; особливості мережевої взаємодії у RTS (Real-time strategy) та TBS (Turn-Based Strategy) іграх.

Очікувані результати

В результаті вивчення дисципліни студенти повинні знати:
– основи комп'ютерних мереж та систем передачі даних;
– основи мережних стеків OSI та TCP/IP;
– основи використання мережних програмних засобів;
– методи та алгоритми роботи мережних протоколів;
– технології організації локальних комп'ютерних мереж та використання засобів інтеграції з мережею Internet;
– технічні прилади канального та мережного рівня передачі даних;
– особливості клієнт-серверної взаємодії в мережевих комп'ютерних іграх;
– методи налаштування та оптимізації засобів транспортного рівня та протоколу TCP/IP.
В результаті вивчення дисципліни студенти повинні вміти:
– аналізувати архітектуру та продуктивність комп'ютерних мереж;
– організовувати локальні комп'ютерні мережі на базі стандартного мережного обладнання та програмних засобів;
– налаштовувати firewall;
– забезпечувати інтеграцію локальних комп'ютерних мереж та окремих комп'ютерів з глобальними мережами (Internet);
– виконувати налаштування параметрів протоколу TCP/IP на різних вузлах комп'ютерних мереж;
– використовувати мережні технології при розробці комп'ютерних ігор;
– оптимізувати мережі під різні типи сервісів.

Змістовний модуль «Розробка мережевих комп'ютерних ігор мовою Java». Представлений матеріал зорієнтовано на студентів вищих навчальних закладів ІТ – спеціальностей, які вивчають такі мови програмування як Java.

Зміст навчального модулю передбачає отримання студентами системних знань в галузі розробки комп'ютерних ігор та формування системи професійних умінь і навичок щодо застосування мережевих технологій та методів клієнт-серверної взаємодії при створені ігрових проектів за допомогою кросплатформної мови програмування Java.

Першу половину навчального модулю присвячено знайомству з особливостями розробки TBS ігор, базовими класами пакета java.net та розробці ігрового протоколу.

Другу половину модулю присвячено знайомству з особливостями розробки ігрового протоколу для RTS ігор.

Наприкінці вивчення даного модулю студент буде вміти створювати прості мережеві ігрові додатки з клієнт-серверною архітектурою на мові програмування Java, та розробляти мережеві ігрові протоколи для TBS і RTS ігор.

2 Опис модуля

Галузь знань: 12 «Інформаційні технології».
Спеціальність: 121 «Інженерія програмного забезпечення»
Рівень: бакалавр.
Назва дисципліни: Мережеві технології
Назва змістовного модуля: Розробка мережевих комп'ютерних ігор мовою Java
Семестр: 4
Кількість кредитних одиниць: дисципліна - 5,0, модуль - 2,0
Орієнтовна кількість часів: дисципліна - 150, модуль - 60
Викладач: к.т.н., доцент Кирийчук Д.Л.

3 Перелік компетентностей та результати навчання*

Загальні (універсальні) компетентності

ЗК-1 Здатність до абстрактного мислення, аналізу та синтезу.
ЗК-2 Здатність застосовувати знання в практичних ситуаціях.
ЗК-3 Здатність вчитися й оволодівати сучасними знаннями, здійснювати

* Освітньо-професійна програма першого (бакалаврського) рівня вищої освіти 2017 року. Спеціальність 121 Інженерія програмного забезпечення, спеціалізація Програмна інженерія.

пошук, оброблення й аналіз інформації з різних джерел.

ЗК-4 Здатність генерувати нові ідеї (креативність), працювати в команді, бути критичним і самокритичним, розробляти проекти, приймати обґрунтовані рішення.

ЗК-5 Здатність оцінювати та забезпечувати якість виконуваних робіт.

ЗК-6 Здатність застосовувати математичний апарат, а також теоретичні, методичні й алгоритмічні основи інформаційних технологій під час вирішення прикладних і наукових завдань в області інформаційних систем і технологій.

Спеціальні (фахові) компетентності

ФК-1 Здатність опановувати сучасні технології математичного моделювання об’єктів, процесів і явищ, розробляти обчислювальні моделі та алгоритми чисельного розв’язання задач математичного моделювання з урахуванням похибок наближеного чисельного розв’язання професійних задач; здійснювати формалізований опис задач дослідження операцій в організаційно-технічних і соціально-економічних системах різного призначення, визначати їх оптимальні рішення, будувати моделі оптимального вибору управління з урахуванням змін параметрів економічної ситуації, оптимізувати процеси управління в системах різного призначення та рівня ієрархії.

ФК-2 Здатність реалізовувати багаторівневі обчислювальні моделі на основі архітектури клієнт-сервер (включаючи сховища, бази та банки даних і знань) для забезпечення обчислювальних потреб багатьох користувачів.

ФК-3 Здатність формулювати та забезпечувати вимоги щодо якості програмного забезпечення у відповідності з вимогами, технічним завданням та стандартами.

ФК-4 Здатність здійснювати аналіз і функціональне моделювання процесів, побудову та застосування функціональних моделей систем; застосування методів та інструментальних засобів для управління процесами життєвого циклу систем відповідно до вимог замовника.

ФК-5 Здатність опановувати та комплексно застосовувати базові загальні знання в області програмування (у тому числі, структурного, функціонального, логічного, об'єктно-орієнтованого, паралельного) та візуального проектування системного та прикладного програмного забезпечення; володіти алгоритмічним мисленням; проектувати та розробляти програмне забезпечення на основі інтеграції провідних сучасних технологій (із застосуванням відповідних моделей, методів та алгоритмів обчислень, структур даних); застосовувати об'єктно-орієнтований підхід під час проектування складних програмних систем методами програмної інженерії для реалізації програмного забезпечення з урахуванням

вимог до його якості, надійності, виробничих характеристик.

ФК-6 Здатність опановувати та комплексно застосовувати базові знання в області принципів, методів і алгоритмів комп'ютерної графіки під час розробки графічних інтерфейсів взаємодії людини з комп'ютером.

ФК-7 Здатність здійснювати процес інтеграції системи, застосовувати стандарти і процедури управління змінами для підтримки цілісності загальної функціональності і надійності програмного забезпечення.

Програмні результати навчання

ПРН-1 Здатність аналізувати проблеми щодо створення програмного забезпечення.

ПРН-2 Здатність, аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки.

ПРН-3 Здатність використовувати знання щодо методів та засобів збору, формулювання та аналізу вимог до програмного забезпечення.

ПРН-4 Здатність застосовувати знання ефективних підходів щодо проектування програмного забезпечення.

ПРН-5 Здатність розуміти основні процеси, фази та ітерації життєвого циклу програмного забезпечення.

ПРН-6 Здатність розуміти і застосовувати знання сучасних підходів щодо оцінки та забезпечення якості програмного забезпечення.

ПРН-7 Здатність розуміти і застосовувати знання щодо відповідних математичних понять, методів доменного, системного і об’єктно-орієнтованого аналізів та математичного моделювання для розробки програмного забезпечення.

ПРН-8 Здатність проводити розрахунок економічної ефективності програмних систем.

ПРН-9 Здатність застосовувати на практиці фундаментальні концепції і основні принципи функціонування мовних, інструментальних і обчислювальних засобів інженерії програмного забезпечення.

ПРН-10 Здатність застосовувати знання методів компонентної розробки програмного забезпечення, виділяючи інтерфейси і реалізації та взаємодію між модулями, підсистемами і компонентами.

Перелік компетентностей та результати навчання
Модуль «Розробка мережевих комп’ютерних ігор мовою Java»

Загальні (універсальні) компетентності	*ЗК-1, ЗК-2, ЗК-6*
Спеціальні (фахові) компетентності	*ФК-2, ФК-5, ФК-6*
Програмні результати навчання	*ПРН-1, ПРН-4, ПРН-7, ПРН-10*

4 Міждисциплінарні зв’язки

Для засвоєння матеріалу використовується такий перелік забезпечуючих дисциплін:

- Вища математика;
- Основи програмування;
- Алгоритми та структури даних;
- Комп'ютерна схемотехніка;
- Архітектура комп’ютерів;
- Комп’ютерна графіка.

5 Мета та передбачувані результати вивчання модуля

5.1 Мета модуля

Метою даного модулю є формування у студентів професійних умінь і навичок щодо розробки комп’ютерних ігор та створення ігрового контенту за допомогою кросплатформної мови програмування Java. Студенти повинні ознайомитись з мережевими протоколами передачі даних і клієнт-серверною технологією передачі даних. Розробити свої протоколи взаємодії для TBS і RTS ігор. Вміти використовувати отриманні знання при створенні комп’ютерних ігор незалежно від платформи.

5.2 Результати навчання

Знання та їх використання

У разі успішного оволодіння матеріалами модуля студент буде вміти організовувати локальні комп’ютерні мережі на базі стандартного мережного обладнання та програмних засобів, виконувати налаштування параметрів протоколу TCP/IP на різних вузлах комп’ютерних мереж, а також налаштування мережних файрволів; розробляти прикладні протоколи для дейтаграмного та потокового режиму передачі даних; вміти писати свої мережні програмні додатки, що використовують стандартні транспортні протоколи передачі даних.

Дослідницькі навички

У разі успішного вивчення модуля студент буде вміти комплексно застосовувати технології та методи розробки кросплатформного програмного забезпечення при проектуванні комп’ютерних ігор; вміти комплексно застосовувати технології та методи мережевої взаємодії при розробці комп’ютерних ігор; вміти застосовувати мову програмування Java при

розробці ігрових додатків; вміти здійснювати апробацію отриманих результатів, приймаючи участь у відповідних формах організації наукових заходів (семінарах, конференціях, конгресах, симпозіумах тощо); вміти впроваджувати отримані результати у практичну діяльність підприємств та організацій.

Спеціальні вміння

У разі успішного вивчення модуля студент буде вміти:

– використовувати відповідні транспортні протоколи для проектування мережевих ігор на мові Java;

– розробляти мережеві протоколи обміну клієнт-серверних додатків;

– використовувати відповідні інструменти мови java для розробки мережевих додатків;

– використовувати можливості інтегрованого середовища розробки, яке поєднує в собі функції текстового редактора з додатковими можливостями для налагодження і виконання мережевих ігрових додатків.

Соціальні вміння

У разі успішного вивчення модуля студент буде вміти працювати в складі професійної проектної команди, що виконує дослідження та розробку в сфері мережевого програмного забезпечення.

Особисті якості

У разі успішного вивчення модуля студент буде вміти:

– обробляти та систематизувати професійні знання щодо створення мережевого програмного забезпечення;

– розробляти, реалізовувати і координувати процеси у мережевих програмних додатках;

– обґрунтовано обирати та освоювати інструментарій з розробки та супроводження мережевого програмного забезпечення;

– аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки.

6 Календарний план семестру і структура модуля

6.1 Місце модуля в структурі дисципліни

Номер	Змістовний модуль	Тиждень вивчання

1	Введення. Основи комп'ютерних мереж.	1-4
2	Глобальні комп'ютерні мережі.	5-6
3	Організація локальної комп'ютерної мережі.	7-9
4	**Розробка мережевих комп'ютерних ігор мовою Java.**	**10-16**

6.2 Інформаційне наповнення змістовного модуля 4

Номер тижня	Зміст
10	Вступ до модуля. Основні поняття та визначення комп'ютерних мережевих ігор.
11-12	Реалізація клієнт-серверної взаємодії у комп'ютерних іграх.
13-14	Особливості використання транспортних протоколів в мережевих іграх.
15	Особливості інформаційного обміну в локальній комп'ютерній мережі при розробці комп'ютерних ігор.
16	Використання глобальної комп'ютерної мережі Інтернет як ігрового середовища.

7 Форми навчання

Навчальний процес здійснюється у таких формах: навчальні аудиторні заняття (лекції, лабораторні, консультації), виконання індивідуальних завдань, самостійна робота, практична підготовка, контрольні заходи.

Аудиторна робота включає 4 лекції та 3 лабораторні роботи.

Лекція - основна форма проведення навчальних занять, яка призначена для засвоєння теоретичного матеріалу.

Лабораторні заняття - форма навчальних занять, на яких студенти поглиблюють теоретичні знання з навчального модулю та набувають практичних навичок роботи в локальній або глобальній мережі на мові Java.

Консультації - це навчальні заняття, на яких студент отримує відповіді викладача на конкретні запитання або пояснення певних теоретичних положень чи практичних аспектів роботи в мережі.

Індивідуальні завдання передбачають проектування та розробку власного ігрового мережевого клієнт-серверного додатку на мові Java. Індивідуальні завдання виконуються студентом самостійно з консультацією викладача.

Самостійна робота здійснюється у вільний від аудиторних навчальних занять час. Вона спрямована на оволодіння студентом навчальним матеріалом, практичними навичками, передбачає отримання нових знань та самостійне вирішення завдань.

8 Порядок проведення атестації

У модулі передбачено виконання лабораторних робіт, модульної контрольної роботи та індивідуального завдання. Підсумкова рейтингова оцінка включає оцінку за виконання і захист лабораторних робіт, модульної контрольної роботи та індивідуального завдання.

Модуль оцінюється за 20 бальною шкалою. Максимальна сумарна оцінка за усі лабораторні роботи в межах модуля складає 12 балів. У кожній роботі 30% нараховується за виконання практичної частини і її відповідність поставленій задачі і 70% – за захист роботи (аргументоване пояснення ходу роботи і відповіді на теоретичні питання).

Модульна контрольна робота оцінюється у 3 бали. Робота складається з трьох питань теоретичного та практичного характеру. Бали нараховуються за коректні відповіді на теоретичні питання та за фрагменти програм, що реалізують практичні завдання.

Виконання та презентація фінального індивідуального завдання в кінці навчання за модулем оцінюється у 5 балів.

Графік проведення поточного оцінювання модуля 3

Номер тижня	Оцінювання
10-11	Оцінка виконання лабораторної роботи 1
12-13	Оцінка виконання лабораторної роботи 2
14-15	Оцінка виконання лабораторної роботи 3
16	Оцінка виконання модульної контрольної роботи Оцінка виконання індивідуального завдання

Представлення звіту щодо виконання лабораторних робіт

Лабораторні роботи виконуються протягом аудиторного заняття. У кінці кожного заняття студент зобов'язаний продемонструвати виконану частину завдання та протягом наступного тижня здати повний проект.

У межах двох тижнів студент зобов'язаний самостійно оформити звіт до попередньої лабораторної роботи, де повинні бути вказані основні кроки, що були виконані у ході даної роботи, описані протоколи мережевої взаємодії та наведені результати роботи програми.

Протягом аудиторного заняття, відведеного для виконання наступної лабораторної роботи студент має захистити попередню роботу, відповівши на питання щодо теоретичного матеріалу та надавши пояснення про хід і результати лабораторної роботи.

Оцінювання лабораторних робіт:

− за кожну лабораторну роботу студент отримує 4 бали (максимальна сумарна оцінка за усі лабораторні роботи в межах модуля складає 12 балів);

− якщо робота виконана невчасно, то знімається 1-2 бали (кількість залежить від терміну запізнення);

− якщо робота виконана не самостійно, то знімається 50% від максимальної кількості балів.

За кожен тиждень запізнення захисту індивідуального завдання нараховується штрафний - 1 бал.

Метод оцінки змістовного модуля 4

Кількість балів в загальній оцінці змістовного модулю відповідає наступному:

Виконання лабораторної роботи 1	максимально 4 бали.
Виконання лабораторної роботи 2	максимально 4 бали.
Виконання лабораторної роботи 3	максимально 4 бали.
Виконання модульної контрольної роботи	максимально 3 бали.
Виконання індивідуального завдання	максимально 5 балів.

Усі набрані бали додаються (максимально 20 балів), штрафні бали за запізнення захисту в представленні звіту з індивідуального завдання (максимально мінус 3 бали) віднімаються. Сумарна оцінка (до 20 балів) є індивідуальна оцінка студента освоєння змістовного модуля 4.

Метод оцінки дисципліни в цілому

Оцінки студентів за результатами вивчення змістовних модулів 1 – 4 підсумовуються. Оцінка заліку (максимально 60 балів) додається. Таким чином розраховується сумарна оцінка студента в балах за дисципліною.

Сумарна оцінка в балах переводиться за нижченаведеною шкалою оцінювання в національну та ЄКТС оцінку:

Сума балів за всі види навчальної діяльності	**Оцінка ECTS**	**Оцінка за національною шкалою**
90 – 100	A	Відмінно
82 – 89	B	Добре
74 – 81	C	Добре
64 – 73	D	Задовільно
60 – 63	E	Задовільно
35 – 59	FX	не зараховано з можливістю повторного складання
0 – 34	F	не зараховано з обов’язковим повторним вивченням дисципліни

9 Зворотній зв’язок

Про результати захисту лабораторних робіт студенти дізнаються під час заняття.

Про результати модульної контрольної роботи студенти дізнаються протягом наступного тижня від здачі.

Консультації для студентів проводяться викладачем впродовж семестру один раз на тиждень.

Інформація щодо оцінки змістовного модуля в цілому надається студентам на 4, 10 та 16 тижні навчання.

Контактні дані для on-line допомоги та консультування:
Викладач: к.т.н., доцент Кирийчук Д.Л.,
e-mail: kyryychuk.dmytro@kntu.net.ua

10 Викладацький склад та допоміжні джерела

Обов'язки викладачів

Основні обов'язки викладачів модуля полягають у проведенні лекцій і лабораторних занять згідно навчальної програми та проведенні контролю якості отриманих знань, умінь і навичок.

Обов'язки координатора дисципліни

Головні обов’язки координаторів модуля полягають у розробці та внесенні змін до навчального модуля у відповідності з поточними потребами, навчальними планами тощо; координації і управлінні професорсько-викладацьким складом; координації проведення заліків.

Обов'язки допоміжного персоналу

Допоміжний персонал здійснює підготовку комп’ютерної техніки та спеціалізованого ігрового обладнання до виконання лабораторних робіт студентами та надає технічну підтримку студентів під час виконання лабораторних робіт.

Контактні дані викладача:

к.т.н., доцент Кирийчук Д.Л., e-mail: kyryychuk.dmytro@kntu.net.ua

11 Навчальна програма і матеріали

11.1 Тема 1: Основні поняття та визначення комп'ютерних мережевих ігор.

Анотація

Лекція знайомить з основними поняттями у ігрових мережевих додатків, жанрами та класифікацією комп'ютерних ігор. Програмні особливості клієнт-серверної взаємодії. Прикладні протоколи.

Мета лекції

Ознайомити студентів з основними поняттями ігрових мережевих додатків, жанрами та класифікацією комп'ютерних ігор. Прикладні протоколи та особливості клієнт-серверної взаємодії.

Очікувані результати

Студент оволодіє основними поняттями: комп'ютерна мережева гра, прикладні протоколи. Буде знати основні переваги та недоліки розробки ігрових протоколів на мові Java.

Контрольні запитання

- Дайте визначення поняттям «комп'ютерна мережева гра», «прикладний протокол», «клієнт-серверна взаємодія».
- Назвіть основні переваги розробки ігрових додатків за допомогою мови Java.
- Які модулі Java використовуються для створення мережних ігрових додатків.
- У чому полягають відмінності кросплатформної розробки комп'ютерних ігор.

11.2 Лабораторна робота № 1. Розробка прикладного протоколу для TBS та RTS ігор.

Анотація

Лабораторна робота знайомить студентів з основними видами прикладних протоколів комп'ютерних мережних ігор та програмним інтерфейсом мови Java, методами роботи в локальній мережі.

Мета лабораторної роботи

Розробити мережевий прикладний протокол (інтерфейс) для ігрового додатку.

Очікувані результати

У разі успішного виконання лабораторної роботи студент буде вміти розробляти мережеві прикладні протоколи для ігрового додатку, та знати особливості протоколів для TBS та RTS мережних ігор.

Контрольні запитання

- Назвіть базові ігрові жанри комп'ютерних ігор;
- Поясніть особливості прикладних протоколі TBS та RTS ігор.

Методичні матеріали та вказівки

- Методичні вказівки щодо виконання лабораторної роботи № 1 доступні за посиланням…….[url].

11.3 Тема 2. Реалізація клієнт-серверної взаємодії у комп'ютерних іграх

Анотація

Лекція знайомить з основними мережними компоненти мови Java, які використовуються для роботи мережних додатків. Описує клієнт-серверну взаємодію комп'ютерних іграх.

Мета лекції

Освоїти основи програмування клієнт-серверної взаємодії комп'ютерних ігор в локальній мережі на мові програмування Java.

Очікувані результати

У разі успішного виконання лабораторної роботи студент буде вміти програмувати клієнт-серверні комп'ютерні ігри в локальній мережі, та знати особливості реалізації у TBS та RTS мережних іграх.

Контрольні запитання

- Назвіть модулі та методи мови програмування Java які використовують в програмуванні мережних додатків;
- Опишіть клієнт-серверну взаємодію ігрових мережних додатків.

11.4 Лабораторна робота № 2. Реалізація клієнт-серверної взаємодії у мережних комп'ютерних TBS іграх

Анотація

Лабораторна робота орієнтована на оволодіння студентами навичок роботи з мережею та потоковими протоколами передавання даних.

Мета лабораторної роботи

Розробка клієнт-серверної мережної TBS гри.

Очікувані результати

У разі успішного виконання лабораторної роботи студент буде вміти використовувати потокові протоколи передавання даних та розробляти власний прикладний ігровий протокол. Використовувати стандартні методи мови Java для роботи з мережею.

Контрольні запитання

- Які методи мови Java та у яких модулях необхідні для потокового передавання даних за допомогою мережі;
- У чому особливість мережної TBS гри;
- Які протоколи передавання даних можна використовувати для мережних TBS ігор.

11.5 Тема 3: Особливості використання транспортних протоколів в мережевих іграх.

Анотація

Лекція знайомить з мережною технологією обміну даних. Актуальністю реплікації даних в мережних RTS іграх та особливістю реалізації різних методів реплікації даних.

Мета лекції

Ознайомити студентів з мережною технологією обміну даних у RTS іграх.

Очікувані результати

Сформувати у студентів знання побудови мережних RTS ігор з урахуванням особливостей клієнт-серверної взаємодії. Сформувати уявлення, щодо оптимального використання мережних протоколів у програмуванні ігор.

Контрольні запитання

- Які особливості клієнт-серверної взаємодії мережних RTS ігор?
- У чому полягає різниця транспортних протоколів для TBS та RTS ігор?
- Опишіть процес створення прикладного протоколу для RTS ігор.

11.6 Лабораторна робота №3. Реалізація клієнт-серверної взаємодії у мережних комп'ютерних RTS іграх.

Анотація

Лабораторна робота орієнтована на оволодіння студентами навичок роботи з мережею та дейтаграмними протоколами передавання даних.

Мета лабораторної роботи

Розробка клієнт-серверної мережної RTS гри.

Очікувані результати

У разі успішного виконання лабораторної роботи студент буде вміти використовувати дейтаграмними протоколи передавання даних та розробляти власний прикладний ігровий протокол. Використовувати стандартні методи мови Java для роботи з мережею.

Контрольні запитання

- Які методи мови Java та у яких модулях необхідні для потокового передавання даних за допомогою мережі;
- У чому особливість мережної RTS гри;
- Які протоколи передавання даних можна використовувати для мережних RTS ігор.

11.7 Тема 4: Особливості інформаційного обміну в локальній комп'ютерній мережі при розробці комп'ютерних ігор.

Анотація

Лекція знайомить з можливостями локальних мереж та особливостями які можна використовувати у комп'ютерних мережних іграх.

Мета лекції

Ознайомити студентів з обміном даних в локальній комп'ютерній мережі при розробці комп'ютерних ігор.

Очікувані результати

Сформувати у студентів знання, щодо розробки комп'ютерних мережних ігор у локальній мережі та закріпити знання транспортних протоколів на прикладі локальних мереж.

Контрольні запитання

- Якими способами можна обмінюватись інформацією між вузлами локальної мережі.
- Назвіть компоненти мови Java, що використовуються при програмуванні ігор в локальній мережі.
- Назвіть протоколи канального рівня.

11 Список джерел інформації

1. Gabriel Gambetta. Fast-Paced Multiplayer (Part I): Client-Server Game Architecture. [Электронный ресурс] // режим доступа: http://www.gabrielgambetta.com/client-server-game-architecture.html

2. Lysenko M. Replication in networked games. 2014. - [Электронный ресурс] // режим доступа: https://0fps.net/2014/02/10/replication-in-networked-games-overview-part-1/

3. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы. - СПб.: Питер, 2002. - 672 с.: ил.

4. Сетевые средства Microsoft Windows NT Server 4.0; Перевод с англ. СПб.: - BHV - Санкт-Петербург, 1997. - 527с.: ил.

5. Козлов А.В. Программирование для Интернет в Delphi 5. - М.: ЗАО «Издательство БИНОМ», 2001. - 368 с.: - ил.

Лабораторна робота № 1. Розробка прикладного протоколу для TBS та RTS ігор.

1. Анотація

Лабораторна робота знайомить студентів з основними видами прикладних протоколів комп'ютерних мережних ігор та програмним інтерфейсом мови Java, методами роботи в локальній мережі.

2. Мета лабораторної роботи

Розробити мережевий прикладний протокол (інтерфейс) для ігрового додатку

3. Очікувані результати

У разі успішного виконання лабораторної роботи студент буде вміти розробляти мережеві прикладні протоколи для ігрового додатку, та знати особливості протоколів для TBS та RTS мережних ігор.

4. Постановка завдання

Розробити прикладний протокол, для обміну повідомленнями між ігровими додатками, об'єднаними в локальну мережу. Додатково реалізувати методи обміну даними із перевіркою та гарантуванням цілісності даних, що передаються.

Варіант RTS та TBS гри узгодити із викладачем. Окремо реалізувати серверну та клієнтську частини.

Звіт з лабораторної роботи повинен містити: тему, постановку задачі, короткі теоретичні відомості, відповіді на контрольні питання, текст програми, результати роботи, висновки.

5. Контрольні питання

1. Назвіть базові ігрові жанри комп'ютерних ігор;
2. Поясніть особливості прикладних протоколі TBS та RTS ігор.

Лабораторна робота № 2. Реалізація клієнт-серверної взаємодії у мережних комп'ютерних TBS іграх.

1. Анотація

Лабораторна робота орієнтована на оволодіння студентами навичок роботи з мережею та потоковими протоколами передавання даних.

2. Мета лабораторної роботи

Розробка клієнт-серверної мережної TBS гри.

3. Очікувані результати

У разі успішного виконання лабораторної роботи студент буде вміти використовувати потокові протоколи передавання даних та розробляти власний прикладний ігровий протокол. Використовувати стандартні методи мови Java для роботи з мережею.

4. Постановка завдання

Реалізувати клієнт-серверний програмний продукт, для обміну повідомленнями між комп'ютерами, об'єднаними в локальну мережу, за протоколом TCP на базі функцій бібліотеки java.net. Реалізувати одну із TBS ігрових моделей: шашки, шахи, хрестики-нулики, морський бій або ін. Окремо реалізувати серверну та клієнтську частини. Передбачити підключення декількох клієнтів.

Окремо реалізувати серверну та клієнтську частини. Передбачити підключення декількох клієнтів.

Звіт з лабораторної роботи повинен містити: тему, постановку задачі, короткі теоретичні відомості, відповіді на контрольні питання, текст програми, результати роботи з екранними формами демонстрації роботи програми, висновки.

5. Протокол IP

Протокол IP (англ. Internet Protocol, протокол міжмережевої взаємодії) описаний в RFC 791. Основна функція протоколу IP - передача пакетів між вузлами, що належать до різних підмереж, через проміжні підмережі. Кожен пакет (дейтаграмма, англ. datagram) обробляється незалежно від інших.

Доставка дейтаграмм не гарантується. Можливі втрати дейтаграмм, доставка з помилками, дублювання і порушення порядку дотримання. Друга функція протоколу IP - виконання фрагментації пакетів при передачі їх між мережами з різним максимально допустимим розміром поля даних кадру (англ. Maximum Transfer Unit, MTU).

Істотна властивість протоколу IP полягає в нетрадиційному порядку передачі бітів : байт передається, починаючи із старшого біта. Крім того, нумерація бітів в байті також починається із старшого: самий старший біт має номер 0, самий молодший - номер 7.

Дейтаграмма (IP -пакет) складається із заголовка і поля даних. Формат заголовка приведений на рис 2.1.

<table>
<tr><th colspan="8">1 байт</th><th colspan="8">2 байт</th><th colspan="8">3 байт</th><th colspan="8">4 байт</th></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td colspan="4">Версия</td><td colspan="4">IHL</td><td colspan="8">Тип Сервиса</td><td colspan="16">Общая длина</td></tr>
<tr><td colspan="16">Идентификация</td><td colspan="3">Флаги</td><td colspan="13">Смещение фрагмента</td></tr>
<tr><td colspan="8">Время жизни</td><td colspan="8">Протокол</td><td colspan="16">Контрольная сумма заголовка</td></tr>
<tr><td colspan="32">IP-адрес отправителя</td></tr>
<tr><td colspan="32">IP-адрес получателя</td></tr>
<tr><td colspan="24">Опции (переменная длина)</td><td colspan="8">Дополнение до 4 байт</td></tr>
</table>

Рисунок 2.1 - Формат заголовка IP -пакета

Поле Номер версії (англ. Version)[4 біта] - вказує використовуваний формат заголовка. Нині основна використовувана версія має номер 4.

Поле Довжина заголовка (англ. Internet Header Length, IHL)[4 біта] - довжина заголовка в чотирьохбайтових словах, мінімальне допустиме значення

- 5 (відповідає довжині заголовка в 20 байт). Максимальна довжина заголовка -60 байт.

Поле Тип сервісу (англ. Type of Service)[8 біт] - вказує на бажані параметри якості обслуговування. Формат байта Типу сервісу приведений на рис 2.2.

Біти	0	1	2	3	4	5	6	7
Підполя	Пріоритет			D	T	R	C	0

Рисунок 2.2 - Структура поля "Пріоритет" заголовка IP -пакета

Поле Пріоритету (англ. Precedence) для звичайних пакетів дорівнює 0, інші значення (від 1 до 7) використовуються для службових цілей, чим більше значення, тим вище пріоритет.

Поля D (англ. Delay, затримка), T (англ. Throughput, пропускна спроможність) і R (англ. Reliability, надійність) використовуються для вказівки найбільш важливої для передавального вузла параметра якості. Вибір відбувається між малою затримкою, великою пропускною спроможністю і високою надійністю. Відповідний біт (біти) встановлюється

в 1, інші - в 0. Як правило, поліпшення одного з параметрів пов'язане з погіршенням інших.

Поле Загальна довжина (англ. Total Length)[16 біт] - довжина дейтаграмми (заголовка і даних) в байтах. Хоча розмір поля дозволяє створювати дейтаграмми завдовжки до 65535 байт, стандарт вимагає, щоб будь-який вузол міг приймати, як мінімум, 576-байтові дейтаграмми (як цілком, так і частями), а відправляти дейтаграмми більшої довжини тільки, будучи упевненим, що одержувач може їх прийняти. Значення 576 вибрано, щоб можна було передавати в одному IP -пакете 512 байт даних ("блок даних розумного розміру", як написано в стандарті; відмітимо, що 512 байт - це типовий розмір сектора жорсткого або гнучкого диска), залишаючи 64 байти під заголовки протоколів : IP -заголовок займає від 20 до 60 байт.

Поле Ідентифікація (англ. Identification)[16 біт] - значення, однакове для усіх дейтаграмм, що містять фрагменти одного пакету ("великого пакету").

Поле Прапори (англ. Flags)[3 біта] - прапори, що управляють фрагментацією, :

0 біт - зарезервований, має дорівнювати 0;

1 біт (DF, англ. Don't Fragment) - "0" = можна фрагментувати, "1" = не можна фрагментувати;

2 біт (MF, англ. More Fragments) - "0" = останній фрагмент, "1" = ще будуть фрагменти.

Поле Зміщення фрагмента (англ. Fragment Offset)[13 біт] - вказує на місце в "великому пакеті", з якого починаються дані поточної дейтаграммы. Вимірюється в 64-бітових (8-байтових) словах. Наприклад, Зміщення фрагмента, рівне двом, означає, що дані поточної дейтаграмми повинні знаходитися в "великому пакеті", починаючи з 16-го байта. Перший фрагмент має нульове зміщення.

Поле Час життя (англ. Time to Live, TTL)[8 біт] - максимальний час, який дейтаграмма може знаходитися в мережі. Кожен маршрутизатор повинен зменшувати це значення на одиницю, і відкидати дейтаграмми зі значенням TTL = 0, передаючи при цьому відправникові відповідне ICMP - повідомлення. Наявність цього поля забезпечує знищення дейтаграмм, що "зациклилися" або "заблукали". Поле TTL також дозволяє обмежити дальність поширення дейтаграмми (це зручно, наприклад, при одночасній передачі безлічі абонентів) і є основою для роботи утиліти traceroute.

Протокол (англ. Protocol)[8 біт] - вказує, дані якого протоколу верхнього рівня передаються в дейтаграмме. Можливі значення цього поля стандартизованы (RFC "Assigned Numbers"), приведемо деякі з них: 1 - ICMP, 4 - IP, 6 - TCP, 17 - UDP, 89 - OSPF.

Поле Контрольна сума заголовка (англ. Header Checksum)[16 біт] - контрольна сума усіх полів заголовка, що обчислюється як доповнення суми усіх 16-бітових слів заголовка (з нульовими бітами в полі контрольної суми). Оскільки деякі поля заголовка (наприклад, час життя) змінюються при

передачі дейтаграмми через мережу, контрольна сума перераховується кожним маршрутизатором. Якщо отримана дейтаграмма з невірною контрольною сумою, така дейтаграмма відкидається.

Поле Адреса відправника (англ. Source IP Address)[32 біта] - IP -адрес відправника дейтаграммы.

Поле Адреса одержувача (англ. Destination IP Address)[32 біта] - IP - адрес одержувача дейтаграммы.

Поле Опції (англ. Options)[змінна довжина] - необов'язкове поле, може містити дані про безпеку, маршрут дейтаграмми (при маршрутизації від джерела) і так далі. У одній дейтаграмме може бути декілька опцій, кожна з яких складається з коду опції (1 байт), довжини опції (1 байт) і байтів даних опції. Якщо для опції не потрібні додаткові дані, вона складається з одного байта - коду опції.

Опції нині практично не використовуються.

Поле Доповнення (англ. Padding) - нульові байти в такій кількості, щоб розмір заголовка був кратний 4 байтам.

6. Функції протоколу транспортного рівня

Відправником і одержувачем даних, передаваних через мережу, з точки зору транспортного рівня, являється застосування (процес). Як будь-яка програма, процеси створюються і знищуються, на кожному вузлі може виконуватися декілька процесів, а кожен процес може мати декілька точок підключення до мережі. Такі логічні точки (програмно організовувані, як правило, у вигляді черг повідомлень) називаються портами (англ. port). Номер порту однозначно ідентифікує процес. Коли вузол отримує дейтаграмму транспортного рівня, він направляє її прикладному процесу, використовуючи номер порту, заданий при встановленні зв'язку.

Порти нумеруються позитивними цілими 16-бітовими числами. Різні протоколи транспортного рівня нумерую свої порти незалежно, тобто, наприклад, порт 20 протоколу TCP і порт 20 протоколу UDP абсолютно не пов'язані один з одним.

Деякі номери портів задані стандартами. Ці номери виділяються організацією IANA (англ. Internet Assigned Numbers Authority). Нині під стандартні порти відведений діапазон від 0 до 1023 (раніше - до 255). Інші порти можуть вільно використовуватися прикладними процесами. Порти в діапазоні від 1024 до 5000 називаються тимчасовими (англ. ephemeral). Призначення цих портів не стандартизовано, але IANA підтримує інформацію про їх використання.

Пара "порт - IP - адреса" називається (у термінології TCP/IP) гніздом або сокетом (англ. socket) і однозначно вказує програмний процес, що виконується на одному з вузлів в мережі.

6.1. Протокол TCP

Протокол TCP (англ. Transmission Control Protocol, Протокол управління передачею) описаний в RFC 793. Він забезпечує надійну передачу потоку даних, використовуючи сервіс передачі дейтаграмм протоколу IP. Пакети, що передаються протоколом TCP, називаються сегментами. Кожен TCP -сегмент розміщується в одному IP -пакете (а в кожному IP -пакете може знаходитися тільки один TCP -сегмент). Надійність передачі забезпечується за допомогою нумерації байтів потоку і підтверджень прийому. Усі байти початкового потоку даних нумеруються (цей номер називається номером в послідовності (англ. sequence number)), і з кожним сегментом передається номер в послідовності його першого байта.

Оскільки два вузли можуть передавати два зустрічні потоки дані по одному TCP -соединению, для передачі підтверджень одного потоку використовуються сегменти зустрічного потоку. У кожному сегменті передається номер в послідовності байта, який збирається прийняти цей вузол.

Після того, як модуль TCP передасть сегмент модулю IP, він записує його копію в чергу на повторну передачу і запускає таймер для цього сегменту. Коли поступить підтвердження прийому сегменту (тобто буде прийнятий сегмент, в якому буде заявлено, що та сторона готова прийняти байт з номером, великим усіх номерів байтів сегменту, що чекає повторної передачі), сегмент віддаляється з черги. Якщо підтвердження не поступає до спрацьовування таймера, сегмент вирушає повторно.

Сегмент складається із заголовка і поля даних. Формат заголовка сегменту TCP приведений на рис 2.3.

<table>
<tr><th colspan="8">1 байт</th><th colspan="8">2 байт</th><th colspan="8">3 байт</th><th colspan="8">4 байт</th></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td colspan="16">Source port</td><td colspan="16">Destination port</td></tr>
<tr><td colspan="32">Sequence number</td></tr>
<tr><td colspan="32">Acknowledgement number</td></tr>
<tr><td colspan="4">Data offset</td><td colspan="6">Reserved</td><td colspan="6">Control bits</td><td colspan="16">Window</td></tr>
<tr><td colspan="16">Checksum</td><td colspan="16">Urgent pointer</td></tr>
<tr><td colspan="24">Options (змінна довжина)</td><td colspan="8">Padding</td></tr>
</table>

Рисунок 2.3 - Формат заголовка TCP -сегмента

Поле Порт відправника (англ. Source port) і Порт одержувача (англ. Destination port)[16 біт] - номери портів на вузлах.

Поле Номер в послідовності (англ. Sequence Number)[32 біта] - номер в потоці першого байта даних цього сегменту. Якщо встановлений керівник біт SYN, то в цьому полі знаходиться початковий номер в послідовності (англ. Initial Sequence Number, ISN) і перший байт даних сегменту має номер в потоці ISN+1.

Поле Номер підтвердження (англ. Acknowledgement number)[32 біта] - номер байта в потоці, очікуваного відправником цього сегменту. При цьому має бути встановлений керівник біт ACK.

Поле Зміщення даних (англ. Data offset)[4 біта] - кількість 32-бітових слів в заголовку TCP -сегмента. Мінімальне значення поля - 5 (20-байтовий заголовок).

Поле Резерв (англ. Reserved)[6 біт] - мають бути заповнені нулями.

Поле Біти управління (англ. Control bits)[6 біт] - від старшого до молодшого:

- URG (англ. Urgent Pointer field significant) - брати до уваги поле "Покажчик терміновості";
- ACK (англ. Acknowledgement field significant) - брати до уваги поле "Номер підтвердження";
- PSH (англ. Push function) - сегмент був примусово відправлений не чекаючи заповнення даними;
- RST (англ. Reset the connection) - перервати зв'язок;
- SYN (англ. Synchronize sequence numbers) - синхронізувати номери байтів в потоці;
- FIN (англ. No more data from sender) - відправник більше не передаватиме дані.

Поле Вікно (англ. Window)[16 біт] - розмір вікна - кількість байтів даних, починаючи з вказаного в полі "Номер підтвердження", які відправник цього сегменту готовий прийняти.

Поле Контрольна сума (англ. Checksum)[16 біт] - контрольна сума усього сегменту (заголовка і даних), обчислюється по алгоритму протоколу IP. Як і в UDP, перед обчисленням контрольної суми до сегменту приписується псевдозаголовок (тієї ж структури, що і в UDP).

Поле Покажчик терміновості (англ. Urgent Pointer)[16 біт] - містить номер першого байта, що має звичайний статус терміновості. При цьому має бути встановлений керівник біт URG.

Поле Опції (англ. Options)[змінний розмір] - додаткова службова інформація. Подібно до опцій заголовка IP -дейтаграммы, мають змінну довжину і можуть бути взагалі відсутніми.

Поле Вирівнювання (англ. Padding) - поле, використовуване для доведення розміру заголовка до цілого числа 32-бітових слів.

Встановлення з'єднання

Перш, ніж процеси зможуть обмінюватися даними по TCP, необхідно встановити TCP -соединение (англ. session); при цьому використовується процедура трьохетапного вітання (англ. three - way handshake) :

- Клієнт генерує випадкове число, яке буде використано як ISN (тобто клієнт нумеруватиме байти, що відправляються, починаючи з цього числа), і відправляє сегмент зі встановленим керівником бітом SYN і ISN в полі "Номер в послідовності".

- Сервер, отримавши сегмент з керівником бітом SYN, зберігає той, що прийшов в нім ISN клієнта, генерує свій ISN як випадкове число, починаючи з якого він нумеруватиме байти, що відправляються, і відправляє сегмент зі встановленими бітами SYN і ACK, що управляють, своїм ISN в полі "Номер в послідовності" і отриманим від клієнта ISN+1 в полі "Номер підтвердження".

- Клієнт, отримавши сегмент з бітами SYN і ACK, що управляють, зберігає той, що прийшов в нім ISN сервера і відправляє сегмент зі встановленим керівником бітами ACK, своїм ISN в полі "Номер в послідовності" і отриманим від сервера ISN+1 в полі "Номер підтвердження".

Після того, як кожна із сторін отримала ISN іншої сторони і підтвердила його, по з'єднанню можна передавати дані.

Відмітимо, що на SYN -сегмент витрачається одне значення номера в послідовності (рівне ISN), хоча в нім не передаються байти призначених для користувача даних.

Розривши з'єднання

Оскільки TCP -соединение по суті є два протилежно спрямованих каналу передачі даних, для коректного розриву з'єднання необхідно і достатньо закрити обидва ці канали.

Якщо одна із сторін більше не збирається передавати дані по з'єднанню, вона повинна передати сегмент зі встановленим керівником бітом FIN. Отримавши такий сегмент, друга сторона повинна підтвердити його отримання сегментом зі встановленим керівником бітом ACK. Після цього один з пари каналів (той, по якому передавався FIN -сегмент) вважається закритим. Другий канал закривається аналогічно (FIN -сегмент в один бік, ACK -сегмент у відповідь), але за ініціативою іншої сторони. З'єднання може бути наполовину закритим скільки завгодно довго.

Таким чином, для коректного розриву з'єднання треба передати чотири сегменти. Відмітимо, що, як і при встановленні з'єднання, на кожного FIN -сегмент витрачається одне значення номера в послідовності, хоча в нім не передаються байти призначених для користувача даних.

Якщо один з комп'ютерів, що встановили з'єднання, відключається від мережі, не виконавши коректного розриву з'єднання (наприклад, при збої ОС або при порушенні роботи каналу зв'язку), то друга сторона з'єднання розірве з'єднання не відразу, а тільки після витікання досить тривалого проміжку часу. Це дає можливість першому комп'ютеру продовжити обмін даними по цьому з'єднанню (наприклад, після відновлення працездатності каналу зв'язку).

Розмір сегменту

При формуванні TCP -сегментов бажано, щоб вони мали такий розмір, щоб що несуть їх IP -пакеты не треба було фрагментувати. Це означає, що максимальний розмір сегменту повинен залежати від того, по мережах яких технологій пролягає шлях між двома вузлами, TCP, що встановили, - з'єднання: якщо на цьому шляху зустрічаються мережі з маленьким розміром

кадру (наприклад, мережі X.25 з розміром кадру 512 байт), максимальний розмір даних поля сегменту має бути таким, щоб розмір IP -пакета (тобто розмір даних + розмір TCP -заголовка (зазвичай 20 байт) + розмір IP - заголовка (зазвичай 20 байт)), в якому він розміщений, не перевершував мінімальний розмір кадру проміжних мереж.

У будь-якому випадку, максимальний розмір IP -пакета, TCP, що несе, - сегмент, не повинен перевищувати максимального розміру поля даних кадру мережі, до якої безпосередньо підключений вузол. Наприклад, для мереж Ethernet з максимальним розміром поля даних кадру, рівним 1500 байт (для кадрів Ethernet II/DIX і 802.3), максимальний розмір сегменту (англ. MSS, Maximum Segment Size) не повинен перевищувати 1460 байт.

Якщо є можливість збільшення MSS (наприклад, відомо, що обмін даними по TCP відбувається тільки в локальній мережі Ethernet), то нею не варто нехтувати: чим більше MSS, тим більше даних переноситься в одному кадрі, тим менше накладні витрати і тим ефективніше передача даних.

За умовчанням (якщо в налаштуваннях TCP/IP не вказано інше значення) розмір сегменту дорівнює 536 байт (оскільки максимальний розмір IP -пакета за умовчанням дорівнює 576 байт).

Вікна прийому

Протокол TCP має засоби управління потоком даних : якщо що приймає дані сторона не устигає їх обробляти, відправник уповільнює або припиняє передачу.

В процесі установки з'єднання кожна із сторін виділяє пам'ять під вхідною і вихідною буфери і передає (у полі "Вікно" того ж SYN -сегмента, в якому передається ISN) другій стороні кількість байт, які вона готова прийняти (не більше, ніж розмір свого вхідного буфера); це число в TCP називається вікном прийому (англ. receive window). Набувши значення вікна прийому, друга сторона зберігає його в блоці управління передачею, пов'язаному із з'єднанням, і в процесі передачі даних стежить за тим, щоб об'єм відправлених, але непідтверджених даних ніколи не перевищував розмір вікна прийому.

В процесі передачі даних сторони можуть змінювати свої вікна прийому (розмір вікна прийому передається в кожному сегменті в полі "Вікно") : наприклад, якщо приймач переобтяжений, він може передати сегмент з полем "Вікно", рівним нулю (щоб відправник припинив передачу), а коли перевантаження приймача закінчиться, передати сегмент з нормальним значенням в полі "Вікно".

На початку передачі даних, поки приймач і передавач не упевнені як канали зв'язку, що сполучають їх, зазвичай обидві сторони встановлюють невелике вікно прийому (розміром в один MSS). У міру того, як приходять підтвердження, розмір вікна збільшується експоненціально (після першого підтвердження - два MSS, після другого - чотири MSS, після третього - вісім MSS і так далі) до деякого порогу, після якого росте лінійно (з кожним новим підтвердженням вікно збільшується на один MSS). Якщо хотябы один

сегмент не підтверджений (чи прийшов сегмент із запитом повторної передачі), розмір вікна зменшується удвічі з кожним непідтвердженим або перезапрошенным сегментом, поки сегменти не почнуть знову підтверджуватися. Далі вікно прийому знову починає рости.

7. Контрольні питання

1. Чи потрібні символьні адреси (забезпечувані, наприклад, серверами DNS) для обміну даними між вузлами TCP/IP -сети?
2. Для чого IP адреса складається з двох полів - номери підмережі і номера вузла в підмережі?
3. Для чого використовується IP адреса 255.255.255.255?
4. Навіщо потрібні маски підмережі?
5. Як протокол IP забезпечує надійну доставку дейтаграмм?

Лабораторна робота № 3. Реалізація клієнт-серверної взаємодії у мережних комп'ютерних RTS іграх.

1. Анотація

Лабораторна робота орієнтована на оволодіння студентами навичок роботи з мережею та дейтаграмними протоколами передавання даних.

2. Мета лабораторної роботи

Розробка клієнт-серверної мережної RTS гри.

3. Очікувані результати

У разі успішного виконання лабораторної роботи студент буде вміти використовувати дейтаграмними протоколи передавання даних та розробляти власний прикладний ігровий протокол. Використовувати стандартні методи мови Java для роботи з мережею.

4. Постановка завдання

Реалізувати клієнт-серверний програмний продукт, для обміну повідомленнями між комп'ютерами, об'єднаними в локальну мережу, за допомогою протоколу UDP на базі функцій бібліотеки java.net. Додатково реалізувати методи обміну даними із перевіркою та гарантуванням цілісності даних, що передаються.

Варіант RTS гри узгодити із викладачем. Окремо реалізувати серверну та клієнтську частини. Передбачити підключення декількох клієнтів.

Звіт з лабораторної роботи повинен містити: тему, постановку задачі, короткі теоретичні відомості, відповіді на контрольні питання, текст програми, результати роботи з екранними формами демонстрації роботи програми, висновки.

5. Коротка теоретична довідка

Бібліотека сокетів використовувалась в інститутах і урядових установах. Тому при розробці бібліотеки враховувалися інтереси й бажання кінцевого користувача. У результаті вийшов потужний інструмент для мережного програмування. З його допомогою можна писати додатки, що не залежать від використовуваного протоколу. Завдяки функціям бібліотеки,

програмування ведеться не машинною мовою, а на прикладному рівні. Крім усього іншого, за допомогою сокетів можна писати власні мережні протоколи. Незважаючи на гадану легкість і простоту програмування на базі сокетов, іноді виникають ситуації, коли відбувається втрата пакетів. Виявити й виправити причину дозволяють мережні утиліти ОС Windows. Також за допомогою мережних утиліт можливе пересилання текстових повідомлень, керування загальними ресурсами, діагностика мережних підключень.

На базі наведених відомостей можливе рішення таких завдань, як розробка багатофункціонального чата, ігрових програм для мережі, організація зв'язку між програмами, що працюють на різних станціях у мережі без звертання до файлу-сервера й т.д. Наприклад:

- написати чат, використовуючи API-функції;
- написати мережну гру;
- реалізувати програмний продукт, що імітує роботу стільникової станції;
- написати програму, що виконує моніторинг мережі;
- сканування мережі й визначення топології;
- написати компонент для обміну даними по одному із протоколів TCP/IP, UDP, IPX/SPX і т.д.;
- реалізувати систему віддаленого адміністрування робочих станцій;
- интернет-пейджер.

WinSock або Windows socket - це інтерфейс прикладного програмування (API), створений для реалізації додатків у мережі на основі протоколу TCP/IP. Для роботи використовується WSOCK32.DLL. Ця бібліотека перебуває в папці \System32 системного каталогу Windows. Існують дві версії WinSock:

WinSock 1.1 - підтримує тільки протокол TCP/IP;

WinSock 2.0 - підтримує додаткове програмне забезпечення.

WinSock 1.1 дав поштовх до розвитку World Wide Web і дозволив одержати доступ в Internet звичайному користувачеві ПК під Windows. Якщо ціль версії 1.1 складалася в рішенні проблеми, то ціль WinSock 2.0 - зробити мережне середовище краще, швидше й надійніше. В WinSock 2.0 додана підтримка інших транспортних протоколів і нові функціональні можливості забезпечення надійності мережного обміну інформацією. WinSock 2.0 дозволяє створювати незалежні від транспортних протоколів додатки, що працюють із TCP/IP (Transmission Control Protocol/Internet Protocol), UDP (User Datagram Protocol), IPX/SPX (Internetwork Packet Exchange/Sequenced Packet Exchange), NetBEUI (NetBios Extended User Interface). Більша ефективність таких додатків досягається за рахунок поєднаного вводу/виводу і сокетів, що розділяються. Специфікація WinSock розділяє функції на три типи:

- блокуючі і не блокуючі (функції Берклі);

- інформаційні (одержання інформації про найменування доменів, служби, протоколи Internet);
- ініціалізації й деініціалізації бібліотеки.

Блокуюча – це функція, що припиняє роботу програми до свого завершення; не блокуюча – це функція, що виконується паралельно із програмою. Список основних функцій, необхідних для створення додатка, наведений у таблицях 3.1-3.3.

Таблиця 3.1 - Блокуючі функції, (функції Берклі)

Назва функції	Призначення
Accept	Створює новий сокет і підключає його до віддаленого комп'ютера
Closesocket	Закриває одну зі сторін з'єднання
Connect	Ініціалізує з'єднання з боку зазначеного сокета
Recv	Приймає дані від підключеного сокета
Recfrom	Приймає дані від підключеного або непідключеного сокета
Send	Посилає дані підключеному сокету
Sendto	Посилає дані підключеному або непідключеному сокету
Bind	Зв'язує віртуальний сокет з фізичним
Inetaddr	Конвертує рядок у значення, яке можна використати в структурі т addr:
Ioctlsocket	Управляє параметрами сокета
Назва функції	Призначення
Listen	Переводить сокет у режим прослуховування порту.
Socket	Створює точку з'єднання

Таблиця 3.2 - Функції ініціалізації й деініціалізації бібліотеки WinSock

Назва функції	Призначення
WSACleanup	Припиняє роботу з WinSock DLL
WSAGetLastError	Одержує інформацію про останню помилку
WSASetLastError	Встановлює інформацію про помилку
WSAStartup	Ініціалізує WinSock DLL

6. Схема клієнт-серверної взаємодії

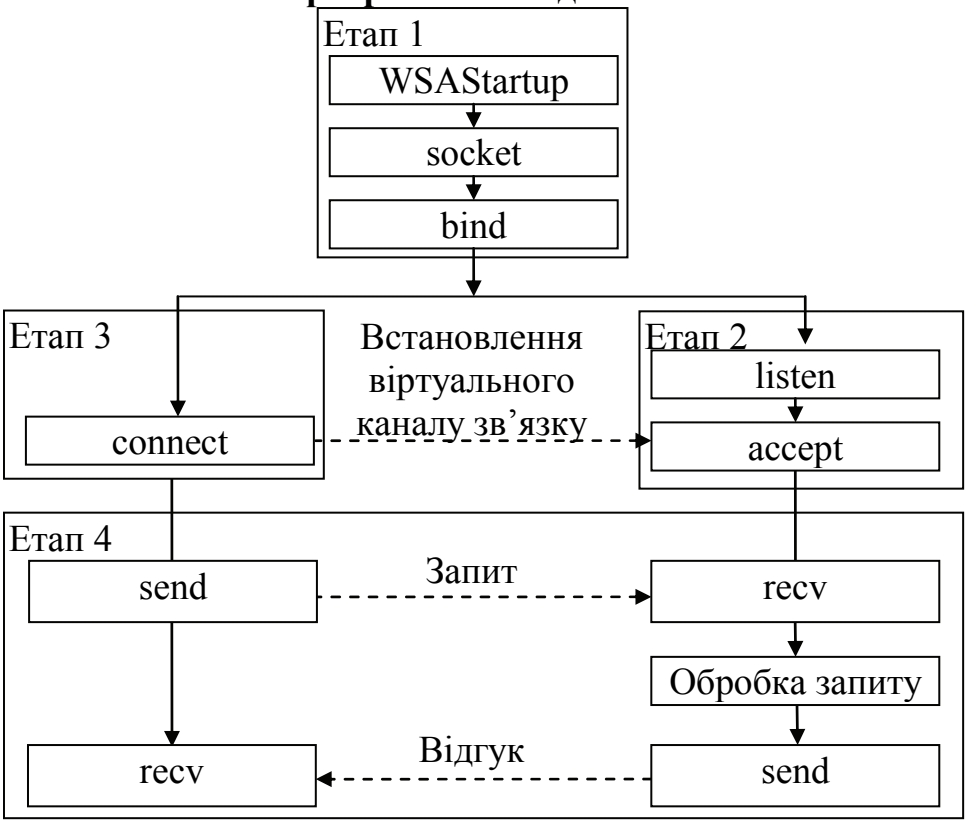

Рисунок 3.1 - Схема взаємодії функцій WinSock

Для реалізації поставленого завдання необхідно створити клієнтський і серверний додаток. Вони різні по організації, але є загальні дії, необхідні як для клієнтської, так і для серверної частини. Схема взаємодії функцій WinSock відображена на рисунку 3. У блоці 1 малюнка відображені загальні дії сервера й клієнта, у блоках 2 і 3, відповідно, дії сервера й клієнта, а в блоці 4 - їхня взаємодія.

Для того, щоб використати функції WinSock, необхідно завантажити бібліотеку WSock32.dll. Як видно із блоку 1 малюнка 3, це здійснюється функцією WSAStartUp [див. 3.7.1]. При вдалому завантаженні бібліотеки, потрібно створити сокет, використовуючи функцію Socket [див. 3.7.2] і асоціювати сокет з адресною структурою SockAddrIn [див. 3.7.3], що містить інформацію про протокол з'єднання, IP-адресу й порт ПК.

6.1 Реалізація клієнтської частини

Після створення клієнтського сокета, він надсилає запит на підключення до сервера, використовуючи функцію connect, указавши в якості одного з параметрів IP-адресу сервера. Ця функція є блокуючою, тобто виконання програми призупиниться доти, доки не прийде відповідь від сервера. При позитивній відповіді сокет підключається до сервера й може з ним взаємодіяти. Реалізація клієнтської частини представлена в блоці 3 рисунку 1.

6.2 Реалізація серверної частини

Після загальних дій, відповідно до блоку 2 рисунку 1, сервер прослуховує порт функцією listen, тобто перевіряє його на предмет запиту від клієнта. Після надходження запиту, сервер обробляє його функцією accept, тобто сервер приєднує клієнта, створюючи новий сокет. Створений сокет виступає як посередник між клієнтом і сервером, тобто "спілкування" відбувається через новий сокет і навпаки.

6.3 Реалізація обміну даними

У блоці 4 рисунку 3 відбитий процес взаємодії між клієнтом і сервером, що зводиться до відправлення й прийому повідомлень. Для відправлення повідомлень використовується функція send або sendto [див. 3.7. 6]. Їхня відмінність полягає в тому, що для функції send необхідне з'єднання (connect, accept), для sendto воно необов'язкове. Для прийому повідомлень застосовують функції recv або recvfrom [див. 3.7. 6]. Для функції recvfrom з'єднання (connect, accept) також необов'язкове.

7. Контрольні запитання

1. Чи підтримувала WinSock 1.1 протокол UDP?
2. Чи вірно твердження, що функція accept є не блокуючою?
3. Що виконує функція bind?
4. Виправте помилку: Socket(MySock, IPPROTOIP, SOCKSTREAM); .
5. Опишіть загальні дії сервера й клієнта.
6. Що означає наступний запис: Connect(Socket1, Addr1, SizeOf(Addr1));?
7. Що необхідно зробити, щоб створити серверний сокет?
8. Для чого використаються потоки?

Лекція 1. Огляд і архітектура обчислювальних мереж

Тема 1. Основні визначення й терміни

Мережа – це сукупність об'єктів, утворених пристроями передачі й обробки даних. Міжнародна організація по стандартизації визначила обчислювальну мережу *як послідовну биту-биту-орієнтовану передачу інформації між зв'язаними один з одним незалежними пристроями.*

Мережі звичайно перебуває в приватному веденні користувача й займають деяку територію й по територіальній ознаці розділяються на:

Локальні обчислювальні мережі (ЛВС) або Local Area Network (LAN), розташовані в одному або декількох близько розташованих будинках. ЛВС звичайно розміщаються в рамках якої-небудь організації (корпорації, установи), тому їх називають корпоративними.

Розподілені комп'ютерні мережі, глобальні або Wide Area Network (WAN), розташовані в різних будинках, містах і країнах, які бувають територіальними, змішаними й глобальними. Залежно від цього глобальні мережі бувають чотирьох основних видів: міські, регіональні, національні й транснаціональні. Як приклади розподілених мереж дуже великого масштабу можна назвати: Internet, EUNET, Relcom, FIDO.

До складу мережі в загальному випадку включаються наступні елементи:

- мережні комп'ютери (оснащені мережним адаптером);
- канали зв'язку (кабельні, супутникові, телефонні, цифрові, волоконно-оптичні, радіоканали й ін.);
- різного роду перетворювачі сигналів;
- мережне встаткування.

Розрізняють два поняття мережі: *комунікаційна мережа* й *інформаційна мережа* (рис. 1.1).

Комунікаційна мережа призначена для передачі даних, також вона виконує завдання, пов'язані з перетворенням даних. Комунікаційні мережі розрізняються по типі використовуваних фізичних засобів з'єднання.

Інформаційна мережа призначена для зберігання інформації й складається з *інформаційних систем*. На базі комунікаційної мережі може бути побудована група інформаційних мереж:

Під *інформаційною системою* варто розуміти систему, що є постачальником або споживачем інформації.

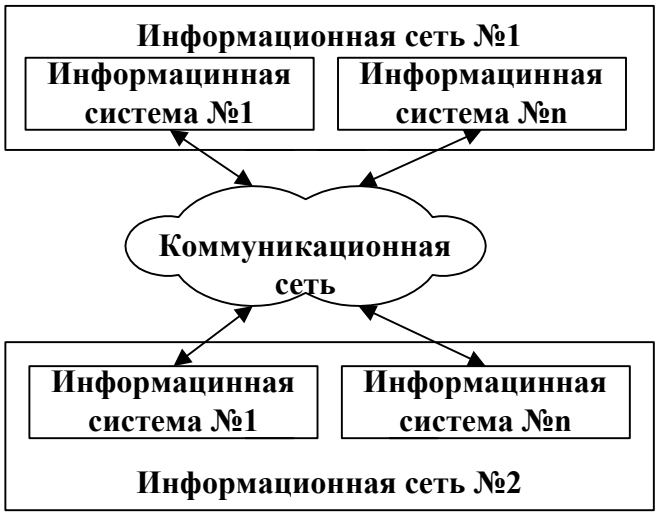

Рисунок 1.1 Інформаційні й комунікаційні мережі

Комп'ютерна мережа складається з *інформаційних систем* і *каналів зв'язку*.

Під *інформаційною системою* варто розуміти об'єкт, здатний здійснювати зберігання, обробку або передачу інформації. До складу *інформаційної системи* входять: комп'ютери, програми, користувачі й інші складові, призначені для процесу обробки й передачі даних. Надалі інформаційна система, призначена для рішення завдань користувача, буде називатися – *робоча станція (client)*. Робоча станція в мережі відрізняється від звичайного персонального комп'ютера (ПК) наявністю *мережної карти* (*мережного адаптера*), каналу для передачі даних і мережного програмного забезпечення.

Под *каналом зв'язку* варто розуміти шлях або засіб, по якому передаються сигнали. Засіб передачі сигналів називають *абонентським,* або *фізичним, каналом*.

Канали зв'язку (data link) створюються по лініях зв'язку за допомогою мережного встаткування й фізичних засобів зв'язку. Фізичні засоби зв'язку побудовані на основі кручених пар, коаксіальних кабелів, оптичних каналів або ефіру. Між взаємодіючими інформаційними системами через фізичні канали комунікаційної мережі й вузли комутації встановлюються *логічні канали.*

Логічний канал – це шлях для передачі даних від однієї системи до іншої. Логічний канал прокладається по маршруті в одному або декількох фізичних каналах. *Логічний канал* можна охарактеризувати, як маршрут, прокладений через фізичні канали й вузли комутації.

Інформація в мережі передається *блоками даних* по процедурах обміну між об'єктами. Ці процедури називають *протоколами передачі даних.*

Протокол – це сукупність правил, що встановлюють формат і процедури обміну інформацією між двома або декількома пристроями.

Завантаження мережі характеризується параметром, називаним *трафиком*. *Трафик (traffic)* – це потік повідомлень у мережі передачі даних.

Під ним розуміють кількісний вимір в обраних крапках мережі числа минаючих *блоків даних* і їхньої довжини, виражене в бітах у секунду.

Істотний вплив на характеристику мережі робить *метод доступу*. *Метод доступу* – це спосіб визначення того, яка з робочих станцій зможе наступної використати канал зв'язку і як управляти доступом до каналу зв'язку (кабелю).

У мережі всі робочі станції фізично з'єднані між собою каналами зв'язку за певною структурою, називаною *топологією*. *Топологія* – це опис фізичних з'єднань у мережі, що вказує які робітники станції можуть зв'язуватися між собою. Тип топології визначає продуктивність, працездатність і надійність експлуатації робочих станцій, а також час звертання до файлового сервера. Залежно від топології мережі використається той або інший метод доступу.

Состав основних елементів у мережі залежить від її архітектури. *Архітектура* – це концепція, що визначає взаємозв'язок, структуру й функції взаємодії робочих станцій у мережі. Вона передбачає логічну, функціональну й фізичну організацію технічних і програмних засобів мережі. Архітектура визначає принципи побудови й функціонування апаратного й програмного забезпечення елементів мережі.

В основному виділяють три види архітектури: архітектура *термінал – головний комп'ютер*, архітектура *клієнт – сервер* і *одноранговa* архітектура.

Сучасні мережі можна класифікувати по різних ознаках: по далекості комп'ютерів, топології, призначенню, переліку надаваних послуг, принципам керування (централізованим і децентралізовані), методам комутації, методам доступу, видам середовища передачі, швидкостям передачі даних і т.д. Всі ці поняття будуть розглянуті більш докладно при подальшому вивченні курсу.

Тема 2. Переваги використання мереж

Комп'ютерні мережі являють собою варіант співробітництва людей і комп'ютерів, що забезпечує прискорення доставки й обробки інформації. Поєднувати комп'ютери в мережі почали більше 30 років тому. Коли можливості комп'ютерів виросли й ПК стали доступні кожному, розвиток мереж значно прискорилося.

З'єднані в мережу комп'ютери обмінюються інформацією й спільно використають периферійне встаткування й пристрої зберігання інформації рис. 1.2.

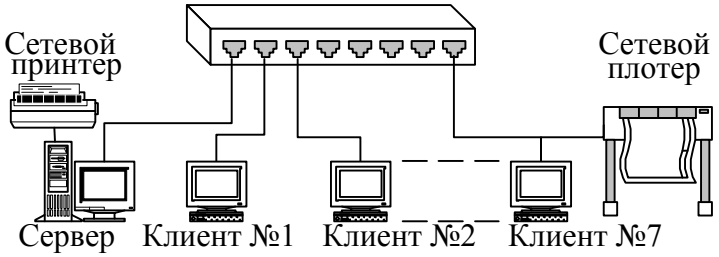

Рисунок 1.2 Використання периферійного встаткування

За допомогою мереж можна розділяти ресурси й інформацію. Нижче перераховані основні завдання, які вирішуються за допомогою робочої станції в мережі, і які важко вирішити за допомогою окремого комп'ютера:

Комп'ютерна мережа дозволить спільно використати периферійні пристрої, включаючи:

- принтери;
- плоттери;
- дискові накопичувачі;
- приводи CD-ROM;
- дисководи;
- стрімери;
- сканери;
- факси-модеми;

Комп'ютерна мережа дозволяє спільно використати інформаційні ресурси:

- каталоги;
- файли;
- прикладні програми;
- гри;
- бази даних;
- текстові процесори.

Комп'ютерна мережа дозволяє працювати із багатокористувальницькими програмами, що забезпечують одночасний доступ всіх користувачів до загальних баз даних із блокуванням файлів і записів, що забезпечує цілісність даних. Будь-які програми, розроблені для стандартних ЛВС, можна використати в інших мережах.

Спільне використання ресурсів забезпечить істотну економію засобів і часу. Наприклад, можна колективно використати один лазерний принтер замість покупки принтера кожному співробітникові або біганини з дискетами до єдиного принтера при відсутності мережі.

Організація електронної пошти. Можна використати *ЛВС* як поштову службу й розсилати службові записки, доповіді й повідомлення іншим користувачам.

Лекція 2. Архітектура мереж

Архітектура мережі визначає основні елементи мережі, характеризує її загальну логічну організацію, технічне забезпечення, програмне забезпечення, описує методи кодування. Архітектура також визначає принципи функціонування й інтерфейс користувача.

У даному курсі буде розглянуто три види архітектур:

архітектура термінал - головний комп'ютер;

одноранговa архітектура;

архітектура клієнт - сервер.

Архітектура термінал - головний комп'ютер

Архітектура термінал - головний комп'ютер (terminal - host computer architecture) - це концепція інформаційної мережі, у якій вся обробка даних здійснюється одним або групою головних комп'ютерів.

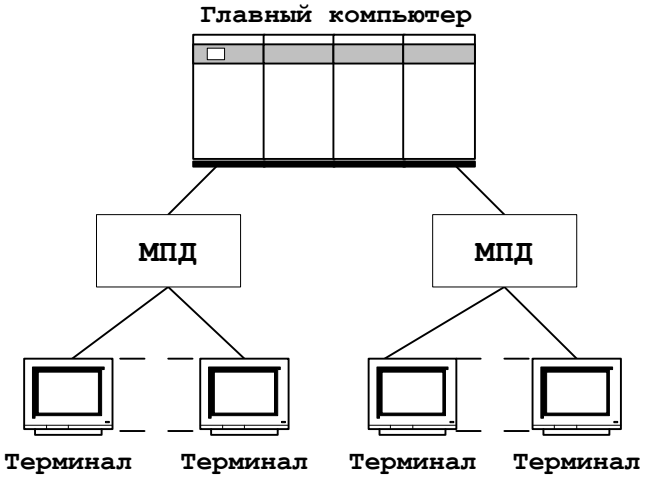

Рисунок 1.3 Архітектура термінал - головний комп'ютер

Розглянута архітектура припускає два типа встаткування:

Головний комп'ютер, де здійснюється керування мережею, зберігання й обробка даних.

Термінали, призначені для передачі головному комп'ютеру команд на організацію сеансів і виконання завдань, уведення даних для виконання завдань і одержання результатів.

Головний комп'ютер через мультиплексори передачі даних (МПД) взаємодіють із терміналами, як представлено на рис. 1.3.

Класичний приклад архітектури мережі з головними комп'ютерами - системна мережна архітектура (System Network Architecture - SNA).

Одноранговa архітектура

Одноранговa архітектура (peer-to-peer architecture) - це концепція інформаційної мережі, у якій її ресурси розосереджені по всіх системах. Дана архітектура характеризується тим, що в ній всі системи рівноправні.

До *однорангових* мереж ставляться малі мережі, де будь-яка робоча станція може виконувати одночасно функції файлового сервера й робочої станції. В *однорангових ЛВС* дисковий простір і файли на будь-якому комп'ютері можуть бути загальними. Щоб ресурс став загальним, його необхідно віддати в загальне користування, використовуючи служби вилученого доступу мережних однорангових операційних систем. Залежно від того, як буде встановлений захист даних, інші користувачі зможуть користуватися файлами відразу ж після їхнього створення. *Однорангові ЛВС* досить гарні тільки для невеликих робочих груп.

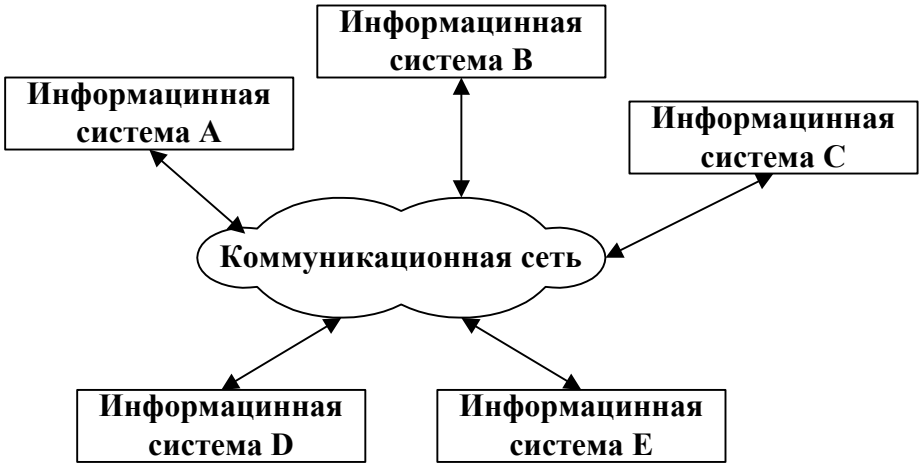

Рисунок 1.4 Однорангова архітектура

Однорангові ЛВС є найбільше простим типом мереж для установлення. Вони на комп'ютері вимагають, крім мережної карти й мережного носія, тільки операційної системи *Windows* 95 або *Windows for Workgroups*. При з'єднанні комп'ютерів, користувачі можуть надавати ресурси й інформацію в спільне користування.

Однорангові мережі мають наступні переваги:

- вони легкі в установці й настроюванні;
- окремі ПК не залежать від виділеного сервера;
- користувачі в стані контролювати свої ресурси;
- мала вартість і легка експлуатація;
- мінімум устаткування й програмного забезпечення;
- немає необхідності в адміністраторі;
- добре підходять для мереж з кількістю користувачів, що не перевищують десяти.

Проблемою однорангової архітектури є ситуація, коли комп'ютери відключаються від мережі. У цих випадках з мережі зникають види *сервісу*, які вони надавали. Мережну безпеку одночасно можна застосувати тільки до одного ресурсу, і користувач повинен пам'ятати стільки паролів, скільки мережних ресурсів. При одержанні доступу до поділюваного ресурсу відчувається падіння продуктивності комп'ютера. Істотним недоліком однорангових мереж є відсутній централізованого адміністрування.

Використання однорангової архітектури не виключає застосування в тій же мережі також архітектури «термінал - головний комп'ютер» або архітектури «клієнт - сервер».

Архітектура клієнт - сервер

Архітектура клієнт – сервер (client-server architecture) – це концепція інформаційної мережі, у якій основна частина її ресурсів зосереджена в серверах, що обслуговують своїх клієнтів (рис. 1.5). Розглянута архітектура визначає два типи компонентів: *сервери й клієнти*.

Сервер - це об'єкт, що надає *сервіс* іншим об'єктам мережі по їхніх запитах. *Сервіс* – це процес обслуговування клієнтів.

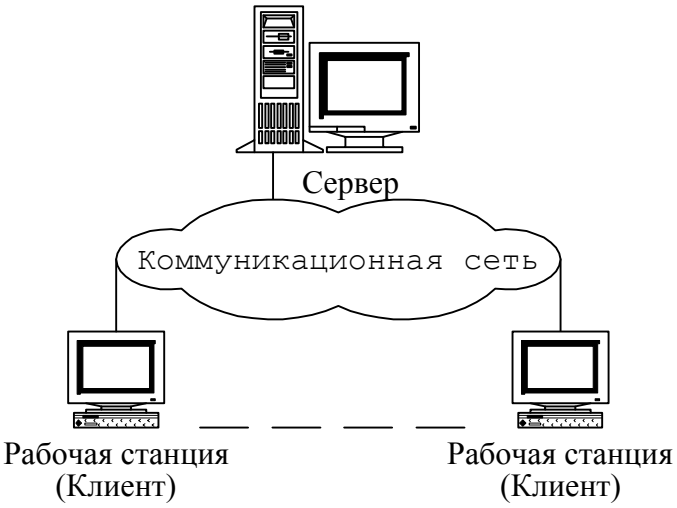

Рисунок 1.5 Архітектура клієнт - сервер

Сервер працює по завданнях клієнтів і керує виконанням їхніх завдань. Після виконання кожного завдання сервер посилає отримані результати клієнтові, що послав це завдання.

Сервісна функція в архітектурі клієнт - сервер описується комплексом прикладних програм, відповідно до якого виконуються різноманітні прикладні процеси.

Процес, що викликає сервісну функцію за допомогою певних операцій, називається *клієнтом*. Їм може бути програма або користувач. На рис. 1.6 наведений перелік сервісів в архітектурі клієнт - сервер.

Клієнти – це робочі станції, які використають ресурси сервера й надають зручні *інтерфейси користувача*. *Інтерфейси користувача* це процедури взаємодії користувача із системою або мережею.

Клієнт є ініціатором і використає електронну пошту або інші сервіси сервера. У цьому процесі клієнт запитує вид обслуговування, установлює сеанс, одержує потрібні йому результати й повідомляє про закінчення роботи.

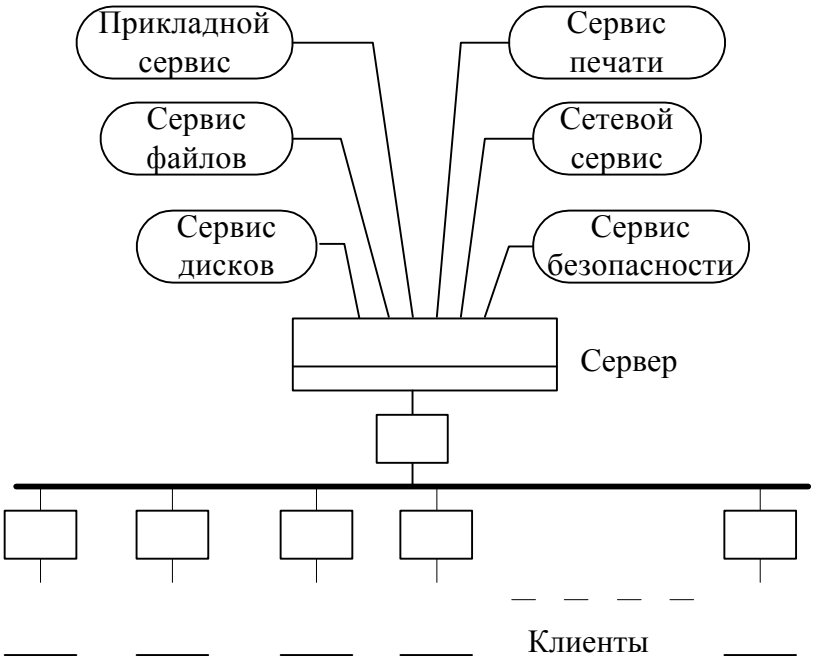

Рисунок 1.6 Модель клієнт-сервер

У *мережах з виділеним файловим сервером* на виділеному автономному *ПК* установлюється серверна мережна операційна система. Цей *ПК* стає *сервером*. Програмне забезпечення (*ПО*), установлене на робочій станції, дозволяє їй обмінюватися даними із сервером. Найпоширеніші мережні операційна системи:

- NetWare фірми Novel;
- Windows NT фірми Microsoft;
- UNIX фірми AT&T;
- Linux.

Крім мережної операційної системи необхідні мережні прикладні програми, що реалізують переваги, надавані мережею.

Мережі на базі серверів мають кращі характеристики й підвищену надійність. Сервер володіє головними ресурсами мережі, до яких звертаються інші робочі станції.

У сучасної клієнт - серверній архітектурі виділяється чотири групи об'єктів: клієнти, сервери, дані й мережні служби. Клієнти розташовуються в системах на робочих місцях користувачів. Дані в основному зберігаються в серверах. Мережні служби є спільно використовуваними серверами й даними. Крім того служби управляють процедурами обробки даних.

Мережі клієнт - серверної архітектури мають наступні переваги:

- дозволяють організовувати мережі з більшою кількістю робочих станцій;
- забезпечують централізоване керування обліковими записами користувачів, безпекою й доступом, що спрощує мережне адміністрування;
- ефективний доступ до мережних ресурсів;

- користувачеві потрібний один пароль для входу в мережу й для одержання доступу до всіх ресурсів, на які поширюються права користувача.
- Поряд з перевагами мережі клієнт - серверної архітектури мають і ряд недоліків:
- несправність сервера може зробити мережа непрацездатної, як мінімум втрату мережних ресурсів;
- вимагають кваліфікованого персоналу для адміністрування;
- мають більше високу вартість мереж і мережного встаткування.

Вибір архітектури мережі

Вибір архітектури мережі залежить від призначення мережі, кількості робочих станцій і від виконуваних на ній дій.

Варто вибрати однорангову мережу, якщо:

- кількість користувачів не перевищує десяти;
- всі машини перебувають близько друг від друга;
- мають місце невеликі фінансові можливості;
- немає необхідності в спеціалізованому сервері, такому як сервер БД, факс-сервер або який-небудь іншої;
- чи є необхідність в централізованому адмініструванні.

Варто вибрати клієнт серверну мережу, якщо:

- кількість користувачів перевищує десяти;
- потрібне централізоване керування, безпека, керування ресурсами або резервне копіювання;
- необхідний спеціалізований сервер;
- потрібний доступ до глобальної мережі;
- потрібно розділяти ресурси на рівні користувачів.

Лекція 3. Семирівнева модель OSI

Для єдиного подання даних у мережах з неоднорідними пристроями й програмним забезпеченням міжнародна організація по стандартах ISO (International Standardization Organization) розробила базову модель зв'язку відкритих систем OSI (Open System Interconnection). Ця модель описує правила й процедури передачі даних у різних мережних середовищах при організації сеансу зв'язку. Основними елементами моделі є рівні, прикладні процеси й фізичні засоби з'єднання. На мал. 2.1 представлена структура базової моделі. Кожний рівень моделі OSI виконує певне завдання в процесі передачі даних по мережі. Базова модель є основою для розробки мережних протоколів. OSI розділяє комунікаційні функції в мережі на сім рівнів, кожний з яких обслуговує різні частини процесу області взаємодії відкритих систем.

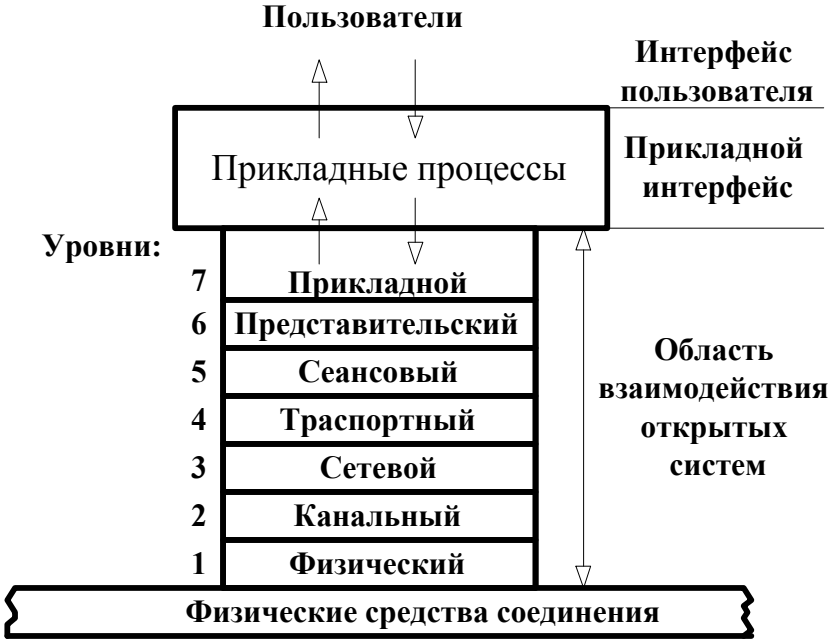

Рисунок 3.1 Модель OSI

Модель OSI описує тільки системні засоби взаємодії, не стосуючись додатків кінцевих користувачів. Додатка реалізують свої власні протоколи взаємодії, звертаючись до системних засобів. Якщо додаток може взяти на себе функції деяких верхніх рівнів моделі OSI, то для обміну даними звертається прямо до системних засобів, що виконують функції нижніх рівнів, що залишилися, моделі OSI.

Тема 1. Взаємодія рівнів моделі OSI

Модель OSI можна розділити на дві різних моделі, як показано на мал.2.2:

1 горизонтальну модель на базі протоколів, що забезпечує механізм взаємодії програм і процесів на різних машинах;

2 вертикальну модель на основі послуг, забезпечуваних сусідніми рівнями один одному на одній машині.

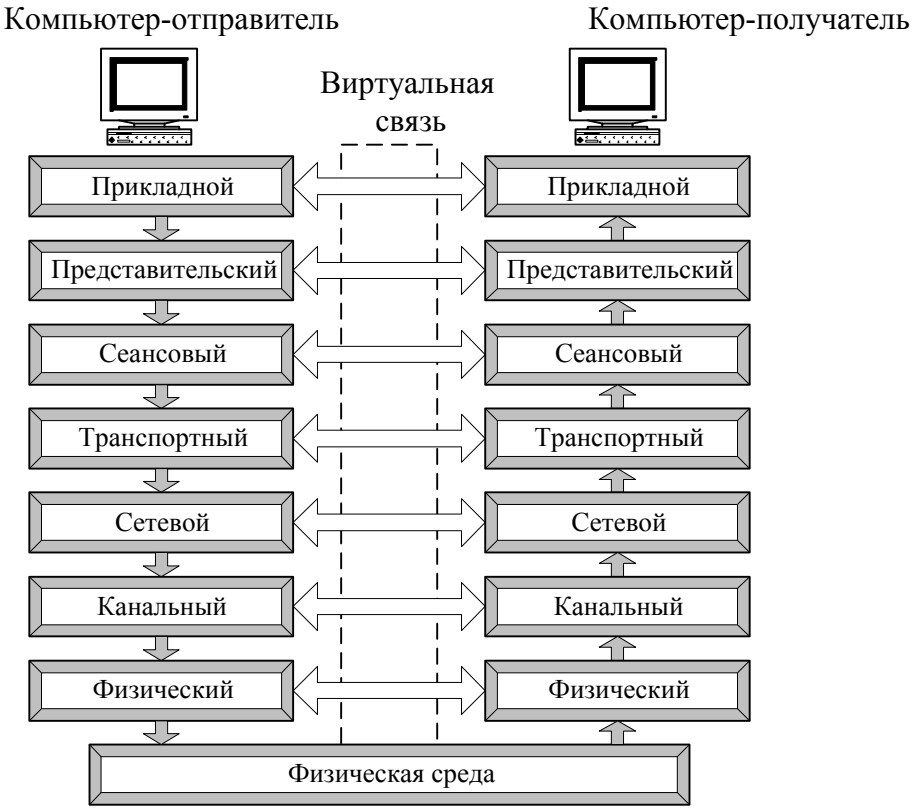

Рисунок 3.2 Схема взаємодії комп'ютерів у базовій еталонній моделі OSI

Кожний рівень комп'ютера-відправника взаємодіє з таким же рівнем комп'ютера-одержувача, начебто він зв'язаний прямо. Такий зв'язок називається логічним або віртуальним зв'язком. У дійсності взаємодія здійснюється між суміжними рівнями одного комп'ютера.

Отже, інформація на комп'ютері-відправнику повинна пройти через всі рівні. Потім вона передається по фізичному середовищу до комп'ютера-одержувача й знову проходить крізь всі шари, поки не доходить до того ж рівня, з якого вона була послана на комп'ютері-відправнику.

У горизонтальній моделі двом програмам потрібен загальний протокол для обміну даними. У вертикальній моделі сусідні рівні обмінюються даними з використанням інтерфейсів прикладних програм API (Application Programming Interface).

Перед подачею в мережу дані розбиваються на пакети. Пакет (packet) – це одиниця інформації, передана між станціями мережі. При відправленні даних пакет проходить послідовно через всі рівні програмного забезпечення. На кожному рівні до пакета додається керуюча інформація даного рівня (заголовок), що необхідний для успішної передачі даних по мережі, як це показано на рис. 2.3, де *Заг* – заголовок пакета, *Кін* – кінець пакета.

На приймаючій стороні пакет проходить через всі рівні у зворотному порядку. На кожному рівні протокол цього рівня читає інформацію пакета, потім видаляє інформацію, додану до пакета на цьому ж рівні стороною, що

відправляє, і передає пакет наступному рівню. Коли пакет дійде до *Прикладного* рівня, вся керуюча інформація буде вилучена з пакета, і дані приймуть свій первісний вид.

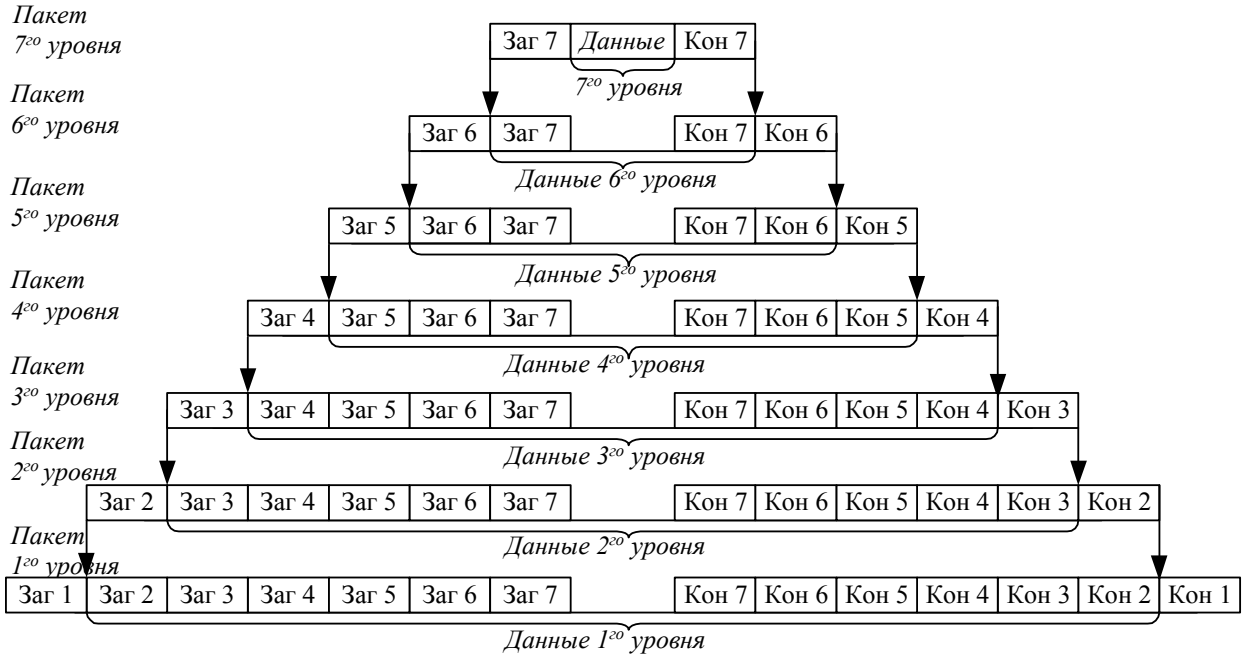

Рисунок 3.3 Формування пакета кожного рівня семирівневої моделі

Кожний рівень моделі виконує свою функцію. Чим вище рівень, тим більше складне завдання він вирішує.

Окремі рівні моделі *OSI* зручно розглядати як *групи програм*, призначених для виконання конкретних *функцій*. Один рівень, приміром, відповідає за забезпечення перетворення даних з *ASCII* в *EBCDIC* і містить *програми* необхідні для виконання цього завдання.

Кожний рівень забезпечує сервіс для вищестоящого рівня, запитуючи у свою чергу, сервіс у нижчестоящого рівня. Верхні рівні запитують сервіс майже однаково: як правило, ця вимога маршрутизації якихось даних з однієї мережі в іншу. Практична реалізація принципів адресації даних покладена на нижні рівні.

Розглянута модель визначає взаємодію відкритих систем різних виробників в одній мережі. Тому вона виконує для них координуючі дії по:

1 взаємодії прикладних процесів;
2 формам подання даних;
3 однаковому зберіганню даних;
4 керуванню мережними ресурсами;
5 безпеки даних і захисті інформації;
6 діагностиці програм і технічних засобів.

На рис. 2.4 наведене короткий опис функцій всіх рівнів.

7. Прикладной представляет набор интерфейсов, позволяющий получить доступ к сетевым службам
6. Представления преобразует данные в общий формат для передачи по сети
5. Сеансовый поддержка взаимодействия (сеанса) между удаленными процессами
4. Транспортный управляет передачей данных по сети, обеспечивает подтверждение передачи
3. Сетевой маршрутизация, управление потоками данных, адресация сообщений для доставки, преобразование логические сетевые адреса и имена в соответствующие им физические
2. Канальный **2.1. Контроль логической связи (LLC):** формирование кадров **2.2. Контроль доступа к среде (MAC):** управление доступом к среде
1. Физический: битовые протоколы передачи информации

Рисунок 2.4 Функції рівнів

Тема 2. Прикладний рівень (Application layer)

Прикладний рівень забезпечує прикладним процесам засобу доступу до області взаємодії, є верхнім (сьомим) рівнем і безпосередньо примикає до прикладних процесів. У дійсності прикладний рівень – це набір різноманітних протоколів, за допомогою яких користувачі мережі одержують доступ до поділюваних ресурсів, таким як файли, принтери або гіпертекстові Web-сторінки, а також організують свою спільну роботу, наприклад за допомогою протоколу електронної пошти [30]. Спеціальні елементи прикладного сервісу забезпечують сервіс для конкретних прикладних програм, таких як програми пересилання файлів і емуляції терміналів. Якщо, наприклад програмі необхідно переслати файли, то обов'язково буде використаний *протокол передачі, доступу й керування файлами* FTAM (File Transfer, Access, and Management). У моделі OSI *прикладна програма*, який потрібно виконати конкретне завдання (наприклад, обновити базу даних на комп'ютері), посилає конкретні дані у вигляді *Дейтаграмми* на *прикладний рівень*. Одне з основних завдань цього рівня - визначити, як варто обробляти

запит прикладної програми, інакше кажучи, який вид повинен прийняти даний запит.

Одиниця даних, який оперує прикладний рівень, звичайно називається повідомленням (message).

Прикладний рівень виконує наступні функції:

Опис форм і методів взаємодії прикладних процесів.

1. Виконання різних видів робіт.
   - передача файлів;
   - керування завданнями;
   - керування системою й т.п.
2. Ідентифікація користувачів по їхніх паролях, адресам, електронним підписам;
3. Визначення функціонуючих абонентів і можливості доступу до нових прикладних процесів;
4. Визначення достатності наявних ресурсів;
5. Організація запитів на з'єднання з іншими прикладними процесами;
6. Передача заявок представницькому рівню на необхідні методи опису інформації;
7. Вибір процедур планованого діалогу процесів;
8. Керування даними, якими обмінюються прикладні процеси й синхронізація взаємодії прикладних процесів;
9. Визначення якості обслуговування (час доставки блоків даних, припустимої частоти помилок);
10. Угода про виправлення помилок і визначенні вірогідності даних;
11. Узгодження обмежень, що накладають на синтаксис (набори символів, структура даних).

Зазначені функції визначають види сервісу, які прикладний рівень надає прикладним процесам. Крім цього, прикладний рівень передає прикладним процесам сервіс, надаваний фізичним, канальним, мережним, транспортним, сеансовим і представницьким рівнями.

На *прикладному рівні* необхідно надати в розпорядження користувачів уже перероблену інформацію. Із цим може впоратися системне й користувальницьке програмне забезпечення.

Прикладний рівень відповідає за доступ додатків у мережу. Завданнями цього рівня є перенос файлів, обмін поштовими повідомленнями й керування мережею.

До числа найпоширеніших протоколів верхніх трьох рівнів ставляться:

1 FTP (File Transfer Protocol) протокол передачі файлів;

2 TFTP (Trivial File Transfer Protocol) найпростіший протокол пересилання файлів;

3 X.400 електронна пошта;

4 Telnet робота з вилученим терміналом;

5 SMTP (Simple Mail Transfer Protocol) простий протокол поштового обміну;

6 CMIP (Common Management Information Protocol) загальний протокол керування інформацією;

7 SLIP **(**Serial Line IP) IP для послідовних ліній. Протокол послідовної посимвольної передачі даних;

8 SNMP (Simple Network Management Protocol) простий протокол мережного керування;

9 FTAM (File Transfer, Access, and Management) протокол передачі, доступу й керування файлами.

Навчальне електронне видання

Кирийчук Д.Л.

ЕЛЕКТРОННИЙ НАВЧАЛЬНИЙ ПОСІБНИК

«РОЗРОБКА МЕРЕЖЕВИХ КОМП'ЮТЕРНИХ ІГОР МОВОЮ JAVA»

Для підготовки студентів
спеціальності 121 «Інженерія програмного забезпечення»,

ISBN 978-617-7573-91-2 (електронне видання)

Підписано до видання 04.04.2019 р. Формат 60×84/8.
Гарнітура Times.
Ум. друк. арк. 5,38. Обл.-вид. арк. 5,79.
Замовлення № 1115.

Книжкове видавництво ФОП Вишемирський В. С.
Свідоцтво про внесення до Державного реєстру суб'єктів
видавничої справи: серія ХС № 48 від 14.04.2005 р.
видано Управлінням у справах преси та інформації
73000, Україна, м. Херсон, вул. Соборна, 2,
тел. (050) 133–10–13, e-mail: printvvs@gmail.com, vish_sveta@rambler.ru