

Міністерство освіти і науки України
ЗНТУ

Кафедра програмних засобів

"Технології розробки мережових додатків"

Конспект лекцій
для студентів спеціальності

121 Інженерія програмного забезпечення

усіх форм навчання

Запоріжжя 2016

Мережні технології: вступ в предмет

У лекції наведені визначення ряду термінів, які використовуються в контексті обчислювальних мереж і мережних технологій. Більшість термінів не мають формального визначення й у різних джерелах можуть приводитися по-різному, але суть буде одна.

Замість передмови

« Головною причиною, чому люди будуть купувати собі додому комп'ютер, стане можливість бути пов'язаними з національною комунікаційною мережею. Ми зараз на самому початку цього етапу, але це буде справжній прорив. Приблизно як телефон.»

Стив Джобс, інтерв'ю для журналу Playboy, 1985 рік

Комунікаційна мережа — система, що складається з об'єктів, що здійснюють функції генерації, перетворення, зберігання й споживання продукту, названих пунктами (вузлами) мережі, і ліній передачі (зв'язків, комунікацій, з'єднань), що здійснюють передачу продукту між пунктами.

Відмінна риса комунікаційної мережі — більші відстані між пунктами в порівнянні з геометричними розмірами ділянок простору, займаних пунктами. У якості продукту можуть фігурувати інформація, енергія, маса, і відповідно розрізняють групи мереж інформаційних, енергетичних, речовинних. У групах мереж можливий поділ на підгрупи. Так, серед речовинних мереж можуть бути виділені мережі транспортні, водопровідні, виробничі й ін. При функціональній проектуванні мереж вирішуються завдання синтезу топології, розподілу продукту по вузлах мережі, а при конструкторській проектуванні виконуються розміщення пунктів у просторі й проведення (трасування) з'єднань.

Інформаційна мережа — комунікаційна мережа, у якій продуктом генерування, переробки, зберігання й використання є інформація.

Обчислювальна (комп'ютерна) мережа — інформаційна мережа, до складу якої входить обчислювальне встаткування. Компонентами обчислювальної мережі можуть бути ЕОМ і периферійні пристрої, що є джерелами й приймачами даних, переданих по мережі. Ці компоненти становлять остаточне встаткування даних (ООД або DTE — Data Terminal Equipment). У якості ООД можуть виступати ЕОМ, принтери, плоттери й інше обчислювальне, вимірювальне й виконавче встаткування автоматичних і автоматизованих систем. Властиво пересилання даних відбувається за допомогою середовищ і засобів, поєднаних за назвою *середовище передачі даних*.

Глобальна обчислювальна мережа поєднує безліч локальних мереж і сотні тисяч — мільйони різнотипних ЕОМ по усьому світу, фізична лінія зв'язку — оптокабель або космічна радіолінія зв'язку.

Локальна обчислювальна мережа (ЛВС) — система зв'язку окремо розташованих ЕОМ на відносно невеликій відстані (звичайно в межах приміщенні й/або поверху будинку); звичайно поєднує до декількох десятків (частіше однотипних) комп'ютерів, фізична лінія зв'язку — «кручена пара» або коаксіальний кабель. У последнее час для зв'язку між вузлами все частіше використовуються беспровідні технології стандартів 802.11.

Корпоративна обчислювальна мережа — локальна обчислювальна мережа (великої) організації, що працює на протоколах Інтернет (стек TCP/IP) що й використовує сервіси Інтернет. При безпосередньому підключенні до глобальної мережі — ще й телекомунікаційне середовище Інтернету.

Робоча група (workgroup) — набір комп'ютерів, об'єднаних одним іменем для роботи із общими сетевыми ресурсами (директорії й файли, принтери й т.п.).

Домен (domain) — певна адміністратором мережі сукупність комп'ютерів, що використовують єдиную базу даних и систему защиты; кожний домен має унікальне ім'я (див. також DNS).

Вузол (host) — підключений до мережі пристрій (звичайно комп'ютер), що ідентифікується власною адресою (наприклад, ір адреса - це унікальне в межах мережі 32 розрядне двійкове число)

Трафік (traffic) — потік повідомлень у поділюванні середовищі передачі даних, часто використовується для грубої оцінки рівня використання передавального середовища (важкий, середній, легкий трафік).

Топологія (topology) мережі — фізична конфігурація машин у мережі.

Маршрутизація — процес визначення (оптимального) шляху доступу до об'єктів (комп'ютерам) мережі.

Пакет, кадр, повідомлення, датаграма — одиниця переданої по мережі інформації, певна кількість байт, згрупована разом і посилаєма одночасно. Те або інше найменування застосовується в контексті опису різних мережевих рівнів.

Мережна архітектура — концепція, що представляє логічну, функціональну й фізичну організацію технічних і програмних засобів мережі й визначальна основні елементи інформаційної мережі, характер і топологію взаємодії цих елементів.

Протокол — це набір семантичних і синтаксичних правил, що визначає поведінку функціональних блоків мережі при передачі даних. Інакше кажучи, протокол — це сукупність угод щодо способу групування даних, що забезпечує їхню передачу в потрібних напрямках і правильну інтерпретацію даних усіма учасниками процесу інформаційного обміну.

Програмне забезпечення комп'ютерних мереж — комплекс програм, що підтримує функції обміну інформацією між окремо розташованими ЕОМ. У цей час програмне забезпечення комп'ютерних мереж звичайно є (іноді опціонально встановлюваної) складовою частиною операційних систем.

Контрольні питання

1. Дайте визначення поняттю «комунікаційна мережа»
2. Дайте визначення поняттю «інформаційна мережа»
3. Дайте визначення поняттю «обчислювальна (комп'ютерна) мережа»
4. Дайте визначення поняттю «локальна обчислювальна мережа (лвс)»
5. Дайте визначення поняттю «корпоративна обчислювальна мережа»
6. Дайте визначення поняттю «глобальна обчислювальна мережа»
7. Дайте визначення поняттю «робоча група»

8. Дайте визначення поняттю «домен (domain)»
9. Дайте визначення поняттю «вузол (host)»
10. Дайте визначення поняттю «топологія (topology)»
11. Дайте визначення поняттю «мережна архітектура»
12. Дайте визначення поняттю «програмне забезпечення комп'ютерних мереж»

Поняття відкритої системи. Модель OSI. Стеки протоколів

Інформаційний обмін — процес багатфункціональний. Родинні функції групуються по призначенню й ці групи називають "рівнями взаємодії". Уніфікація рівнів дозволяє створювати гетерогенні мережі зі складною топологією. В основі уніфікації — поняття еталонної мережної моделі. Модель як така лише описує порядок мережної взаємодії, який реалізується у вигляді стека протоколів.

- Модель OSI
- Стеки протоколів

Модель OSI

Обмін інформацією між комп'ютерами, об'єднаними в мережу, дуже складне завдання. Це пов'язане з тим, що існує багато виробників апаратних і програмних засобів обчислювальних систем. Єдиний вихід — уніфікувати засобу сполучення систем, а саме використовувати *відкриті системи*. Відкрита система взаємодіє з іншими системами на основі єдиних загальнодоступних стандартів і специфікацій.

В 1984 р. Міжнародна Організація по Стандартизації (ISO) представила індустріальний стандарт — *модель взаємодії відкритих систем* (Open System Interconnection Reference Model — OSI/RM, у радянській літературі — ЭМБОС), щоб допомогти постачальникам створювати сумісні мережні апаратні й програмні засоби. Відповідно до цієї моделі виділяються наступні рівні (мал.1):

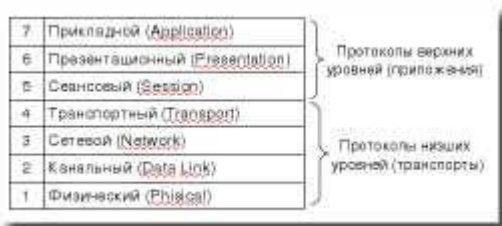


Рис. 1. Еталонна модель OSI

- фізичний (Physical);
- канальний (Data Link);
- мережний (Network);
- транспортний (Transport);
- сеансовий (Session);
- представницький (Presentation) ;
- прикладний (Application).

Відповідно до еталонної моделі OSI ці рівні взаємодіють так, як показано на мал. 2. Таким чином, складне завдання обміну інформацією між комп'ютерами в мережі розбивається на ряд щодо незалежних і менш складних підзадач *взаємодії між суміжними рівнями*.



Рис. 2. Взаємодія між рівнями OSI

Зв'язок між рівнями двох мережних вузлів (*горизонтальна взаємодія*) виконується відповідно до уніфікованих правил — **протоколами взаємодії**

В автономній системі передача даних між рівнями (*вертикальна взаємодія*) реалізується через **інтерфейси API**

Границю між сеансовим і транспортним рівнями можна розглядати як границю між протоколами прикладного рівня й протоколами нижчих рівнів. Якщо прикладний, представницький і сеансовий рівні забезпечують прикладні процеси сеансу взаємодії, то чотири нижчі рівні вирішують проблеми транспортування даних.

Два самі нижчі рівні — фізичний і канальний — реалізуються апаратними й програмними засобами, інші п'ять більш високих рівнів реалізуються, як правило, програмними засобами.

При передачі інформації від прикладного процесу в мережу на фізичний рівень відбувається її обробка, яка полягає в розбивці переданих даних на окремі блоки, перетворенні форми вистави або кодування даних у блоці й додаванні до кожного блоку **заголовка** (header) відповідного рівня (див. приклад). Кожний заголовок характеризує використовуваний протокол обробки даних, причому кожний рівень сприймає як дані весь блок, отриманий від попереднього рівня, включаючи приєднаний заголовок. Така побудова еталонної моделі дозволяє закласти (*інкапсулювати*) у кожний переданий по фізичному середовищу інформаційний блок відомості, необхідні для вибору послідовності протоколів для здійснення зворотних перетворень на стороні, що ухвалює інформацію.

Фізичний рівень

Цей рівень визначає механічні, електричні, процедурні й функціональні характеристики встановлення, підтримки й розмикання фізичного з'єднання між кінцевими системами. Фізичний рівень визначає такі характеристики з'єднання, як рівні напруг, синхронізацію й фізичну швидкість передачі даних, максимальні відстані передачі, конструктивні параметри рознімань і інші аналогічні характеристики. Відомі стандарти RS-232-C, V.24 і IEEE 802.3 (Ethernet).

Канальний рівень

Канальний рівень (рівень ланки даних, інформаційно-канальний рівень) відповідає за надійну передачу даних через фізичний канал, а саме:

- забезпечує фізичну адресацію (на відміну від мережної або логічної адресації);
- забезпечує виявлення помилок у передачі й відновлення даних;
- відслідковує топологію мережі й забезпечує дисципліну використання мережного каналу кінцевою системою;
- забезпечує повідомлення про несправності;
- забезпечує впорядковану доставку блоків даних і керування потоком інформації.

Для ЛВС канальний рівень розбивається на два підуровні:

- LLC (Logical Link Control) — забезпечує керування логічною ланкою, тобто властиво функції канального рівня;
- MAC (Media Access Control) — забезпечує спеціальні методи доступу до середовища поширення.

Мережний рівень

Цей рівень забезпечує можливість з'єднання й вибір маршруту між двома кінцевими системами, підключеними до різних підмереж (сегментам), які можуть бути розділені безліччю підмереж і можуть перебувати в різних географічних пунктах. Протоколи маршрутизації дозволяють мережі з маршрутизаторів вибирати оптимальні маршрути через зв'язані між собою підмереж.

Транспортний рівень

Транспортний рівень забезпечує вищим рівням послуги із транспортування даних, а саме:

- забезпечує надійне транспортування даних через об'єднану мережу;
- забезпечує механізми для установки, підтримки й упорядкованого завершення дії віртуальних каналів;
- забезпечує виявлення й усунення несправностей транспортування;
- стежить за тим, щоб кінцева система не була перевантажена занадто більшою кількістю даних.

Інакше кажучи, транспортний рівень забезпечує інтерфейс між процесами й мережею, установлює логічні канали між процесами й забезпечує передачу по цих каналах інформаційних блоків. Ці логічні канали називаються транспортними.

Сеансовий рівень

Сеансовий рівень реалізує встановлення, підтримку й завершення сеансу взаємодії між прикладними процесами абонентів. Сеансовий рівень синхронізує діалог між об'єктами представницького рівня, визначає крапки синхронізації для проміжного контролю й відновлення при передачі файлів. Цей рівень також дозволяє робити обмін даними в режимі, заданому прикладний програмою, або надає можливість вибору режиму обміну.

Крім основної функції керування діалогом сеансовий рівень надає кошти для вибору класу послуг і повідомлення про виняткові ситуації (проблемах сеансового, представницького й прикладного рівнів).

Представницький рівень

Представницький рівень (рівень представлення даних) визначає синтаксис, формати й структури вистави переданих даних (але не торкається семантики, значення даних). Для того, щоб інформація, що посилає із прикладного рівня однієї системи, була системою, що читається на прикладному рівні іншої, представницький рівень здійснює трансляцію між відомими форматами вистави інформації за рахунок використання уніфікованого формату вистави інформації.

Таким чином, цей рівень забезпечує службові операції, обирає на прикладному рівні, для інтерпретації переданих і одержуваних даних: керування інформаційним обміном, відображення даних і керування структурованими даними. Ці службові дані дозволяють зв'язувати воедино термінали й обчислювальні засоби різних типів. Прикладом протоколу цього рівня є XDR.

Прикладний рівень

На відміну від інших рівнів прикладний рівень — найближчий до користувача рівень OSI — не надає послуги іншим рівням OSI, однак він забезпечує прикладні процеси, що лежать за межами масштабу моделі OSI.

Прикладний рівень забезпечує безпосередню підтримку прикладних процесів і програм кінцевого користувача (СУБД, текстових процесорів, програм банківських терміналів і т.д.) і керування взаємодією цих програм з мережею передачі даних:

- ідентифікує й установлює наявність передбачуваних партнерів для зв'язку;
- синхронізує спільно працюючі прикладні програми;
- установлює угоди по процедурах усунення помилок і керування цілісністю інформації;
- визначає достатність наявних ресурсів для передбачуваного зв'язку.

Модель OSI не є реалізацією, вона лише пропонує порядок організації взаємодії між компонентами системи. Реалізаціями цих правил є **стеки протоколів**.

Стеки протоколів

Стек Netbios/SMB

Фірми Microsoft і IBM спільно працювали над мережними засобами для персональних комп'ютерів, тому стік протоколів Netbios/SMB є їхнім спільним дітищем. Засобу Netbios з'явилися в 1984 році як мережне розширення стандартних функцій базової системи введення/виводу (BIOS) IBM PC для мережної програми PC Network фірми IBM, яка на прикладному рівні (мал. 3) використовувала для реалізації мережних сервісів протокол SMB.

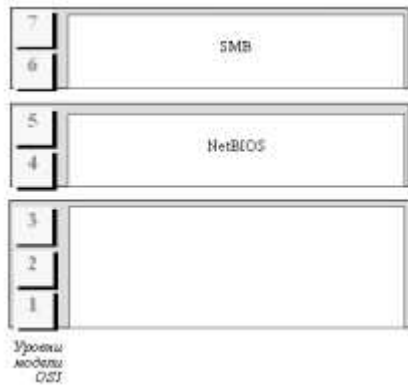


Рис. 3. Стік Netbios/SMB

Протокол **Netbios** працює на трьох рівнях моделі взаємодії відкритих систем: **мережному, транспортному й сеансовому**. Netbios може забезпечити сервіс більш високого рівня, чому протоколи IPX і SPX, однак не має здатність до маршрутизації. Таким чином, Netbios не є мережним протоколом у точному значенні цього слова. Netbios містить багато корисних мережних функцій, які можна віднести до мережного, транспортному й сеансовому рівням, однак з його допомогою неможлива маршрутизація пакетів, тому що в протоколі обміну кадрами Netbios не вводиться таке поняття як мережа. Це обмежує застосування протоколу Netbios локальними мережами, не розділеними на підмережі. Netbios підтримує як дейтаграмний обмін, так і обмін із установленням з'єднань.

Протокол **SMB**, що відповідає прикладному й представницькому рівням моделі OSI, регламентує взаємодія робочої станції із сервером. У функції SMB входять наступні операції:

- Керування сесіями. Створення й розрив логічного каналу між робочою станцією й мережними ресурсами файлового сервера.
- Файловий доступ. Робоча станція може звернутися до файлсерверу із запитом на створення й видалення каталогів, створення, відкриття й закриття файлів, читання й запис у файли, перейменування й видалення файлів, пошук файлів, одержання й установку файлових атрибутів, блокування записів.
- Сервіс печатки. Робоча станція може ставити файли в чергу для печатки на сервері й одержувати інформацію про чергу печатки.
- Сервіс повідомлень. SMB підтримує просту передачу повідомлень із наступними функціями: послати просте повідомлення; послати широкомовне повідомлення; послати початок блоку повідомлень; послати текст блоку повідомлень; послати кінець блоку повідомлень; переслати ім'я користувача; скасувати пересилання; одержати ім'я машини.

Через велику кількість додатків, які використовують функції API, надавані Netbios, у багатьох мережних ОС ці функції реалізовані у вигляді інтерфейсу до своїх транспортних протоколів. В Netware є програма, яка емує функції Netbios на основі протоколу IPX, існують програмні емулятори Netbios для Windows NT і стека TCP/IP.

Стек TCP/IP

Стек TCP/IP, називаний також стеком Dod і стеком Internet, є одним з найбільш популярних стеків комунікаційних протоколів. Стік був розроблений з ініціативи Міністерства оборони США (Department of Defence, Dod) для зв'язку експериментальної мережі Arpanet з іншими сателітними мережами як набір загальних протоколів для різноманітного обчислювального середовища. Мережа ARPA підтримувала розроблювачів і дослідників у військових областях. У мережі ARPA зв'язок між двома комп'ютерами здійснювалася з використанням протоколу Internet Protocol (IP), який і донині є основним у стеці TCP/IP і фігурує в назві стека.

Великий внесок у розвиток стека TCP/IP вніс університет Берклі, реалізувавши протоколи стека у своїй версії ОС UNIX. Широке поширення ОС UNIX привело й до широкого поширення протоколу IP і інших протоколів стека. На цьому ж стеці працює всесвітня інформаційна мережа Internet, чий підрозділ Internet Engineering Task Force (IETF) вносить основний вклад в удосконалювання стандартів стека, публікує їх у формі специфікацій RFC.

Тому що стік TCP/IP був розроблений до появи моделі взаємодії відкритих систем ISO/OSI, те, хоча він також має багаторівневу структуру, відповідність рівнів стека TCP/IP рівням моделі OSI досить умовно.

Структура протоколів TCP/IP наведено на малюнку 4. Протоколи TCP/IP діляться на 4 рівня.

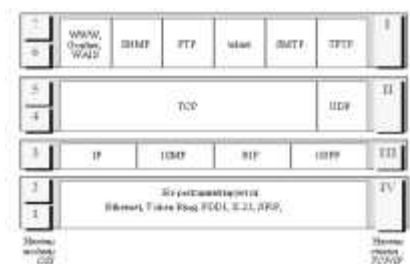


Рис. 4. Стік TCP/IP

Самий нижній (рівень IV) — рівень міжмережних інтерфейсів — відповідає фізичному й каналному рівням моделі OSI. Цей рівень у протоколах TCP/IP не регламентується, але підтримує всі популярні стандарти фізичного й каналного рівня: для локальних каналів це Ethernet, Token Ring, FDDI, для глобальних каналів — власні протоколи роботи на аналогових виділених лініях, що комутуються й, SLIP/PPP, які встановлюють з'єднання типу "крапка — крапка" через послідовні канали глобальних мереж, і протоколи територіальних мереж X.25 і ISDN. Розроблена також спеціальна специфікація, що визначає використання технології ATM у якості транспорту каналного рівня.

Наступний рівень (рівень III) — це рівень міжмережної взаємодії, який займається передачею даних із використанням різних локальних мереж, територіальних мереж X.25, ліній спеціального зв'язку й т.п. У якості основного протоколу мережного рівня (у термінах моделі OSI) у стеці використовується протокол **IP**, який споконвічно проектував як протокол передачі пакетів у складених мережах локальних мереж, що полягають із великої кількості, об'єднаних як локальними, так і глобальними зв'язками. Тому протокол IP добре працює в мережах зі складною топологією, раціонально використовуючи наявність у них підсистем і ощадливо витрачаючи пропускну здатність низькошвидкісних ліній зв'язку. Протокол IP є даним протоколом.

До рівня межмережної взаємодії ставляться й усі протоколи, пов'язані зі складанням і модифікацією таблиць маршрутизації, такі як протоколи збору маршрутної інформації **RIP** (Routing Internet Protocol) і **OSPF** (Open Shortest Path First), а також протокол міжмережними керуючих повідомлень **ICMP** (Internet Control Message Protocol). Останній протокол призначений для обміну інформацією про помилки між маршрутизатором і шлюзом, системою-джерелом і системою-приймачем, тобто для організації зворотного зв'язку. За допомогою спеціальних пакетів ICMP повідомляється про неможливість доставки пакета, про перевищення часу життя або тривалості складання пакета із фрагментів, про аномальні величини параметрів, про зміну маршруту пересилання й типу обслуговування, про стан системи й т.п.

Наступний рівень (рівень II) називається основним. На цьому рівні функціонують протокол керування передачею **TCP** (Transmission Control Protocol) і протокол дейтаграм користувача **UDP** (User Datagram Protocol). Протокол TCP забезпечує стійке віртуальне з'єднання між вилученими прикладними процесами. Протокол UDP забезпечує передачу прикладних пакетів дейтаграмним методом, тобто без установаження віртуального з'єднання, і тому вимагає менших накладних витрат, чому TCP.

Верхній рівень (рівень I) називається прикладним. За довгі роки використання в мережах різних країн і організацій стік TCP/IP нагромадив велику кількість протоколів і сервісів прикладного рівня: протокол копіювання файлів **FTP**, протоколи віддаленого керування telnet і ssh, поштовий протокол **SMTP**, гіпертекстові сервіси доступу до вилученої інформації, такі як **WWW** і багато інші. Коротко зупинимося на деяких із протоколів стека, найбільше тісно пов'язаних з тематикою даного курсу.

Протокол **SNMP** (Simple Network Management Protocol) використовується для організації мережного керування. Проблема керування розділяється тут на два завдання. Перше завдання пов'язана з передачею інформації. Протоколи передачі керуючої інформації визначають процедуру взаємодії сервера із програмою-клієнтом, що працює на хосту адміністратора. Вони визначають формати повідомлень, якими обмінюються клієнти й сервери, а також формати імен і адрес. Друге завдання пов'язана з контрольованими даними. Стандарти регламентують, які дані повинні зберігатися й накопичуватися в шлюзах, імена цих даних і синтаксис цих імен. У стандарті SNMP визначена специфікація інформаційної бази даних керування мережею. Ця специфікація, відома як база даних **MIB** (Management Information Base), визначає ті елементи даних, які хост або шлюз повинен зберігати, і припустимі операції над ними.

Протокол пересилання файлів **FTP** (File Transfer Protocol) реалізує дистанційний доступ до файлу. Для того, щоб забезпечити надійну передачу, FTP використовує в якості транспорту протокол із установаженням з'єднань — TCP. Крім пересилання файлів протокол, FTP пропонує й інші послуги. Так користувачеві надається можливість інтерактивної роботи з вилученою машиною, наприклад, він може роздрукувати вміст її каталогів, FTP дозволяє користувачеві вказувати тип і формат даних, що запам'ятовуються. Нарешті, FTP виконує аутентифікацією користувачів. Перш, ніж одержати доступ до файлу, відповідно до протоколу користувачі повинні повідомити своє ім'я й пароль.

У стеці TCP/IP протокол FTP пропонує найбільш широкий набір послуг для роботи з файлами, однак він є й самим складним для програмування. Додатка, яким не потрібні всі можливості FTP, можуть використовувати іншої, більш економічний протокол — найпростіший протокол пересилання файлів **TFTP** (Trivial File Transfer Protocol). Цей протокол реалізує тільки передачу файлів, причому в якості транспорту використовується більш простий, чому TCP, протокол без установаження з'єднання — UDP.

Протокол **telnet** забезпечує передачу потоку байтів між процесами, а також між процесом і терміналом. Найбільше часто цей протокол використовується для емуляції термінала вилученої EOM.

Контрольні питання

1. Для чого призначена модель OSI?
2. Перелічить рівні моделі OSI
3. Які завдання вирішує прикладний рівень моделі OSI?
4. Які завдання вирішує рівень вистав моделі OSI?
5. Які завдання вирішує транспортний рівень моделі OSI?
6. Які завдання вирішує мережний рівень моделі OSI?
7. Які завдання вирішує каналний рівень моделі OSI?
8. Які завдання вирішує фізичний рівень моделі OSI?
9. Як у моделі OSI проходить обмін даними між рівнями?
10. Що таке «стік протоколів»

Однорангові й клієнт-серверні мережі. Ролі вузлів у мережі.

Основне призначення комп'ютерних мереж — спільне використання ресурсів і здійснення інтерактивного зв'язку між вузлами мережі. Ресурси (resources) — це дані, додатки й периферійні пристрої.

Усі мережі мають деякі загальні компоненти, функції й характеристики. У їхньому числі:

- сервери (server) — комп'ютери, що надають свої ресурси мережним користувачам;
- клієнти (client) — комп'ютери, що здійснюють доступ до мережних ресурсів, надаваним сервером;
- середовище (media) — спосіб з'єднання комп'ютерів;
- спільно використовувані дані — файли, надавані серверами по мережі;
- спільно використовувані периферійні пристрої, наприклад принтери, бібліотеки CD-ROM і т.д., — ресурси, надавані серверами;
- ресурси — файли, принтери й інші елементи, використовувані в мережі.

Незважаючи на певні подібності, мережі розділяються на два типи: однорангові (peer-to-peer) і на основі сервера (server based).

Однорангові мережі

В одноранговій мережі всі комп'ютери рівноправні: немає ієрархії серед комп'ютерів і немає виділеного (dedicated) сервера. Як правило, кожний комп'ютер функціонує і як клієнт, і як сервер; інакше кажучи, немає окремого комп'ютера, відповідального за адміністрування всієї мережі. Усі користувачі самостійно вирішують, які дані на своєму комп'ютері зробити загальнодоступними по мережі

Оскільки в одноранговій мережі кожний комп'ютер функціонує і як клієнт, і як сервер, користувачі повинні мати достатній рівень знань, щоб працювати і як користувачі, і як адміністратори свого комп'ютера.

Мережі на основі сервера

Однорангова мережа, де комп'ютери виступають у ролі й клієнтів, і серверів, може виявитися недостатньо продуктивною. Тому більшість мереж використовує виділені сервери. Виділеним називається такий сервер, який функціонує тільки як сервер (крім функцій клієнта або робочої станції). Вони спеціально оптимізовані для швидкої обробки запитів від мережних клієнтів і для керування захистом файлів і каталогів.

Файл-сервери й принт-сервери управляють доступом користувачів відповідно до файлів і принтерів. Наприклад, щоб працювати з текстовим процесором, Ви насамперед повинні запустити його на своєму комп'ютері. Документ текстового процесора, що зберігається на файл-сервері, завантажується на згадку Вашого комп'ютера, і, таким чином, Ви можете працювати із цим документом на своєму комп'ютері. Інакше кажучи, файл-сервер призначений для зберігання файлів і даних.

На серверах додатків виконуються прикладні частини клієнт-серверних додатків, а також перебувають дані, доступні клієнтам. Наприклад, щоб спростити добування даних, сервери зберігають більші обсяги інформації в структурованому виді. Ці сервери відрізняються від файл- і принт-серверів. В останніх файл або дані цілком копіюються на запитувачий комп'ютер. А в сервері додатків на запитувачий комп'ютер пересилаються тільки результати запиту. Додаток-клієнт на вилученому комп'ютері одержує доступ до даних, збережених на сервері додатків. Однак замість усієї бази даних на Ваш комп'ютер із сервера завантажуються тільки результати запиту.

Поштові сервери управляють передачею електронних повідомлень між користувачами мережі. Комунікаційні сервери управляють потоком даних і поштових повідомлень між цією мережею й іншими мережами або вилученими користувачами.

У розширеній мережі використання серверів різних типів здобуває особливу актуальність. Необхідно тому враховувати всі можливі нюанси, які можуть виявитися при розростанні мережі, для того щоб зміна ролі певного сервера надалі не відбилася на роботі всієї мережі. Сервер спроектований так, щоб надавати доступ до безлічі файлів і принтерів, забезпечуючи при цьому високу продуктивність і захист.

Адміністрування й керування доступом до даних здійснюється централізовано. Ресурси, як правило, розташовані також централізовано, що полегшує їхній пошук і підтримку. Основним аргументом при виборі мережі на основі сервера є, як правило, захист даних. У таких мережах, наприклад, як Windows NT Server, проблемами безпеки може займатися один адміністратор: він формує політику безпеки (security policy) і застосовує її відносно кожного користувача мережі.

Комбіновані мережі

Існують і комбіновані типи мереж, що сполучають кращі якості однорангових мереж і мереж на основі сервера.

Багато адміністраторів вважають, що така мережа найбільше повно задовольняє їхні запити. Операційні системи для мереж на основі сервера, наприклад Microsoft Windows NT Server або Novell® Netware®, у цьому випадку відповідають за спільне використання основних додатків і даних. На комп'ютерах-клієнтах можуть виконуватися операційні системи Microsoft Windows NT Workstation або Windows 95, які будуть управляти доступом до ресурсів виділеного сервера й у той же час надавати в спільне використання свої жорсткі диски, а в міру необхідності дозволяти доступ і до своїх даних.

Комбіновані мережі — найпоширеніший тип мереж, але для їхньої правильної реалізації й надійного захисту необхідні певні знання й навички планування.

Контрольні питання

1. Опишіть особливості однорангових мереж.
2. Опишіть особливості мереж на основі сервера.

Компоненти мережного додатка. Клієнт-серверна взаємодія й ролі серверів.

Як правило комп'ютери й програми, що входять до складу інформаційної системи, не є рівноправними. Деякі з них володіють ресурсами (файлова система, процесор, принтер, база даних і т.д.), інші мають можливість звертатися до цих ресурсів. Комп'ютер (або програму), керуючий ресурсом, називають сервером цього ресурсу (файлсервер, сервер бази даних, обчислювальний сервер...). Клієнт і сервер якого-небудь ресурсу можуть перебувати як на одному комп'ютері, так і на різних комп'ютерах, зв'язаних мережею.

У рамках багаторівневої вистави обчислювальних систем можна виділити три групи функцій, орієнтованих на рішення різних підзадач:

1. функції введення й відображення даних (забезпечують взаємодію з користувачем);
2. прикладні функції, характерні для даної предметної області;
3. функції керування ресурсами (файловою системою, базою даних і т.д.)



Рис.1. Компоненти мережного додатка

Виконання цих функцій в основному забезпечується програмними засобами, які можна представити у вигляді взаємозалежних компонентів (рис. 1), де:

- *компонент вистави* відповідає за користувацький інтерфейс;
- *прикладний компонент* реалізує алгоритм рішення конкретного завдання;
- *компонент керування ресурсом* забезпечує доступ до необхідних ресурсів.

Автономна система (комп'ютер, не підключений до мережі) представляє всі ці компоненти як на різних рівнях (ОС, службове ПО й утиліти, прикладне ПО), так і на рівню додатків (не характерно для сучасних програм). Так само й мережа — вона представляє всі ці компоненти, але, у загальному випадку, розподілені між вузлами. Завдання зводиться до забезпечення мережної взаємодії між цими компонентами.

Архітектура «клієнт-сервер» визначає загальні принципи організації взаємодії в мережі, де є *сервери*, вузливпостачальники деяких специфічних функцій (сервісів) і *клієнти*, споживачі цих функцій.

Практичні реалізації такої архітектури називаються **клієнтсерверними технологіями**. Кожна технологія визначає власні або використовує наявні правила взаємодії між клієнтом і сервером, які називаються *протоколом обміну (протоколом взаємодії)*.

Двухзвенна архітектура

У будь-якій мережі (навіть однорангової), побудованої на сучасних мережних технологіях, присутні елементи клієнтсерверної взаємодії, найчастіше на основі **двухзвенної архітектури**. Двухзвенною (two-tier, 2-tier) вона називається через необхідність розподілу *трьох базових компонентів* між двома вузлами (клієнтом і сервером).

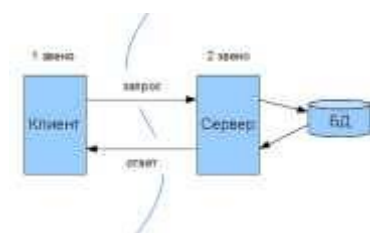


Рис.2. Двухзвенна клієнтсерверна архітектура

Двухзвенна архітектура використовується в клієнтсерверних системах, де сервер відповідає на клієнтські запити прямо й у повному обсязі, при цьому використовуючи тільки власні ресурси. Т.е. сервер не викликає сторонні мережні додатки й не звертається до сторонніх ресурсів для виконання якої-небудь частини запиту (рис. 2)

Розташування компонентів на стороні клієнта або сервера визначає наступні основні моделі їх взаємодії в рамках двухзвенна архітектури:

- **сервер терміналів** — розподілена вистава даних;
- **файлсервер** — доступ до вилученої бази даних і файловим ресурсам;
- **сервер БД** — вилучена вистава даних;
- **сервер додатків** — вилучений додаток.

Перераховані моделі з варіаціями представлені на рис. 3.



Рис.3. Моделі клієнтсерверної взаємодії

Історично першої з'явилася модель розподіленого вистави даних (модель сервер терміналів). Вона реалізовувалася на універсальній ЕОМ (мэйнфреймеі), що виступала в ролі сервера, з підключеними до неї алфавітно-цифровими терміналами. Користувачі виконували введення даних із клавіатури термінала, які потім передавалися на мэйнфрейм і там виконувалася їхня обробка, включаючи формування «картинки» з результатами. Ця «картинка» і поверталася користувачеві на екран термінала.

З появою персональних комп'ютерів і локальних мереж, була реалізована модель файлового сервера, що представляв доступ файловим ресурсам, у т.ч. і до вилученої бази даних. У цьому випадку виділений вузол мережі є файловим сервером, на якому розміщені файли бази даних. На клієнтах виконуються додатки, у яких сполучені компонент вистави й прикладний компонент (СУБД і прикладна програма), що використовують підключену вилучену базу як локальний файл. Протоколи обміну при цьому представляють набір низкорівневих викликів операцій файлової системи.

Така модель показала свою неефективність через те, що при активній роботі з таблицями БД виникає більше навантаження на мережу. Частковим рішенням є підтримка тиражування (реплікації) таблиць і запитів. У цьому випадку, наприклад при зміні даних, оновлюється не вся таблиця, а тільки модифікована її частина.

З появою спеціалізованих СУБД з'явилася можливість реалізації іншої моделі доступу до вилученої бази даних — моделі сервера баз даних. У цьому випадку ядро СУБД функціонує на сервері, прикладна програма на клієнті, а протокол обміну забезпечується за допомогою мови SQL. Такий підхід у порівнянні з файловим сервером веде до зменшення завантаження мережі й уніфікації інтерфейсу «клієнт-сервер». Однак, мережний трафік залишається досить високим, крім того, по колишньому неможливо задовільне адміністрування додатків, оскільки в одній програмі сполучаються різні функції.

З розробкою й впровадженням на рівні серверів баз даних механізму збережених процедур з'явилася концепція *активного сервера БД*. У цьому випадку частина функцій прикладного компонента реалізовані у вигляді збережених процедур, виконуваних на стороні сервера. Інша прикладна логіка виконується на клієнтській стороні. Протокол взаємодії — відповідний діалект мови SQL.

Переваги такого підходу очевидні:

- можливо централізоване адміністрування прикладних функцій;
- зниження вартості володіння системою (ТОС, total cost of ownership) за рахунок оренди сервера, оренди покупки;
- значне зниження мережного трафіка (тому що передаються не SqlpЗапити, а виклики збережених процедур).

Основний недолік — обмеженість засобів розробки збережених процедур у порівнянні з мовами високого рівня.

Реалізація прикладного компонента на стороні сервера представляє наступну модель — сервер додатків. Перенос функцій прикладного компонента на сервер знижує вимоги до конфігурації клієнтів і спрощує адміністрування, але представляє підвищені вимоги до продуктивності, безпеки й надійності сервера.

У цей час намічається тенденція повернення до того, із чого починалася клієнтсерверна архітектура — до централізації обчислень на основі моделі термінала-сервера. У сучасній реінкарнації термінали відрізняються від своїх алфавітно-цифрових предків тим, що маючи

мінімум програмних і апаратних засобів, представляють мультимедійні можливості (у т.ч. *графічний користувацький інтерфейс*). Роботу терміналів забезпечує високопродуктивний сервер, куди винесене всі, аж до віртуальних драйверів пристроїв, включаючи драйвери відеопідсистеми.

Трьохзвенна архітектура

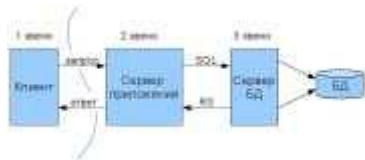


Рис.4. Трьохзвенна клієнтрсерверна архітектура

Ще одна тенденція в клієнтрсерверних технологіях зв'язана із усе більшим використанням розподілених обчислень. Вони реалізуються на основі моделі сервера додатків, де мережний додаток розділений на дві й більш частин, кожна з яких може виконуватися на окремому комп'ютері. Виділені частини додатка взаємодіють один з одним, обмінюючись повідомленнями в заздалегідь погодженому форматі. У цьому випадку двухзвенна клієнтрсерверна архітектура стає **трьохзвенною (three-tier, 3-tier)**.

Як правило, третьою ланкою в трьохзвенній архітектурі стає сервер додатків, тобто компоненти розподіляються в такий спосіб (рис. 4):

1. Вистава даних — на стороні клієнта.
2. Прикладний компонент — на виділеному сервері додатків (як варіант, що виконує функції проміжного ПО).
3. Керування ресурсами — на сервері БД, який і представляє запитувані дані.

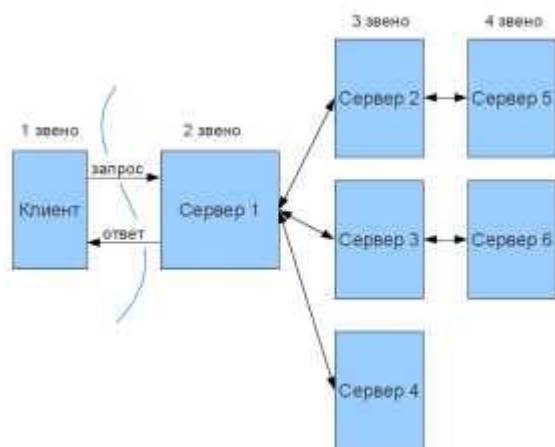


Рис.5. Багатоланкова (N-tier) клієнтрсерверна архітектура

Трьохзвенна архітектура може бути розширена до **багатоланкової (N-tier, Multi-tier)** шляхом виділення додаткових серверів, кожний з яких буде представляти власні сервіси й користуватися послугами інших серверів різного рівня. Абстрактний приклад багатоланкової моделі наведений на рис. 5.

Порівняння архітектур

Двухзвенна архітектура простіше, тому що всі запити обслуговуються одним сервером, але саме через це вона менш надійна й висуває підвищені вимоги до продуктивності сервера.

Трьохзвенна архітектура складніше, але завдяки тому, що функції розподілені між серверами другого й третього рівня, ця архітектура представляє:

1. Високий ступінь гнучкості й масштабованості.
2. Високу безпеку (тому що захист можна визначити для кожного сервісу або рівня).
3. Високу продуктивність (тому що завдання розподілені між серверами).

КлієнтСерверні технології

Архітектура клієнт-сервер застосовується у великій кількості мережних технологій, використовуваних для доступу до різних мережних сервісів. Коротко розглянемо деякі типи таких сервісів (і серверів).

WebвСервери

Споконвічно представляли доступ до гіпертекстових документів по протоколу HTTP (Hyper Text Transfer Protocol). Зараз підтримують розширені можливості, зокрема роботу з бінарними файлами (зображення, мультимедія й т.п.).

Сервери додатків

Призначені для централізованого рішення прикладних завдань у деякій предметній області. Для цього користувачі мають право *запускати серверні програми* на виконання. Використання серверів додатків дозволяє знизити вимоги до конфігурації клієнтів і спрощує загальне керування мережею.

Сервери баз даних

Сервери баз даних використовуються для обробки користувацьких запитів мовою SQL. При цьому СУБД перебуває на сервері, до якого й підключаються клієнтські додатки.

ФайлрСервери

ФайлеСервер *зберігає* інформацію у вигляді файлів і представляє користувачам доступ до неї. Як правило файлесервер забезпечує й певний рівень захисту від несанкційованого доступу.

Проксі-Сервер

По-перше, діє як посередник, допомагаючи користувачам одержати інформацію з Інтернету й при цьому забезпечуючи захист мережі.

По-друге, зберігає часто запитувану інформацію в кеш-пам'яті на локальному диску, швидко доставляючи її користувачам без повторного звертання до Інтернету.

Файрволи (брандмауери)

Міжмережні екрани, що аналізують і фільтрують минаючий мережний трафік, з метою забезпечення безпеки мережі.

Поштові сервери

Представляють послуги з відправлення й одержання електронних поштових повідомлень.

Сервери дистанційного доступу (RAS)

Ці системи забезпечують зв'язок з мережею по лініях, що комутируються. Вилучений співробітник може використовувати ресурси корпоративної ЛВС, підключившись до неї за допомогою звичайного модему.

Це лише кілька типів із усього різноманіття клієнтсерверних технологій, використовуваних як у локальних, так і в глобальних мережах.

Для доступу до тих або інших мережних сервісів використовуються клієнти, можливості яких характеризуються поняттям «товщини». Воно визначає конфігурацію встановлення й програмне забезпечення, наявні в клієнта. Розглянемо можливі граничні значення:

«Тонкий» клієнт

Цей термін визначає клієнта, обчислювальних ресурсів якого досить лише для запуску необхідного мережного додатка через webфінтерфейс. Користувацький інтерфейс такого додатка формується засобами *статичного* HTML (виконання Javascript не передбачається), уся прикладна логіка виконується на сервері. Для роботи тонкого клієнта досить лише забезпечити можливість запуску webбраузера, у вікні якого й здійснюються всі дії. Із цієї причини webбраузер часто називають "універсальним клієнтом".

«Товстий» клієнт

Таким є робоча станція або персональний комп'ютер, що працюють під керуванням власної дискової операційної системи, що й мають необхідний набір програмного забезпечення. До мережних серверів «товсті» клієнти звертаються в основному за додатковими послугами (наприклад, доступ до webсерверу або корпоративній базі даних). Так само під «товстим» клієнтом мається на увазі й клієнтський мережний додаток, запущене під керуванням локальної ОС. Такий додаток сполучає компонент вистави даних (графічний користувацький інтерфейс ОС) і прикладний компонент (обчислювальні потужності клієнтського комп'ютера).

Останнім часом усе частіше використовується ще один термін: «rich»-client. «Rich»-клієнт свого роду компроміс між «товстим» і «тонким» клієнтом. Як і «тонкий» клієнт, «rich»-клієнт також представляє графічний інтерфейс, описуваний уже засобами XML, що й включає деяку функціональність товстих клієнтів (наприклад інтерфейс drag-and-drop, вкладки, множинні вікна, що випадають меню й т.п.)

Прикладна логіка «rich»-клієнта також реалізована на сервері. Дані відправляються в стандартному форматі обміну, на основі того ж XML (протоколи SOAP, XML-RPC) і інтерпретуються клієнтом.

Деякі основні протоколи «rich»-клієнтів на базі XML наведені нижче:

- XAML (extensible Application Markup Language) — розроблений Microsoft, використовується в додатках на платформі .NET;

- XUL (XML User Interface Language) — стандарт, розроблений у рамках проекту Mozilla, використовується, наприклад, у поштовому клієнті Mozilla Thunderbird або браузері Mozilla Firefox;
- Flex — мультимедійна технологія на основі XML, розроблена Macromedia/Adobe.

Висновок

Отже, основна ідея архітектури «клієнт-сервер» полягає в поділі мережного додатка на кілька компонентів, кожний з яких реалізує специфічний набір сервісів. Компоненти такого додатка можуть виконуватися на різних комп'ютерах, виконуючи серверні й/або клієнтські функції. Це дозволяє підвищити надійність, безпеку й продуктивність мережних додатків і мережі в цілому.

Контрольні питання

1. У чому полягає основна ідея ДО-ІЗ взаємодії?
2. У чому відмінності між поняттями «клієнтрсерверна архітектура» і «клієнтрсерверна технологія»?
3. Перелічіть компоненти ДО-ІЗ взаємодії.
4. Які завдання виконує компонент вистави в ДО-З архітектурі?
5. З якою метою засобу доступу до БД представлені у вигляді окремого компонента в ДО-З архітектурі?
6. Для чого бізнес-логіка виділена як окремий компонент у ДО-З архітектурі?
7. Перелічіть моделі клієнтсерверної взаємодії.
8. Опишіть модель «файлсервер».
9. Опишіть модель «сервер БД».
10. Опишіть модель «сервер додатків»
11. Опишіть модель «сервер терміналів»
12. Перелічіть основні типи серверів.

Проміжне ПО. Призначення, функції й види middleware.

Проміжне програмне забезпечення використовується в багатоланкових клієнтрсерверних додатках для інтеграції розподілених компонентів у єдину інформаційну систему.

Еволюція архітектури «клієнт-сервер»

Класична двухзвенна клієнт-серверна модель цілком підходить для невеликих корпоративних підрозділів з обмеженим числом користувачів і невисоким навантаженням на сервер. Однак, у міру впровадження клієнтрсерверних технологій у сферу високочитичних корпоративних додатків, що обслуговують сотні користувачів і обробляють істотні масиви даних, стали очевидні недоліки двухзвених рішень:

- обмежені можливості масштабування;

- необхідність зміни клієнтських додатків при зміні роботи серверної логіки.

Розв'язати ці проблеми дозволяє перехід на три- і багатоланкові моделі, у яких прикладна логіка винесена в окремий рівень (на сервер додатків). Це дозволяє ефективно розподіляти навантаження й забезпечити прозоре нарощування як функціональності сервера (серверів) додатків, так і числа, що обслуговуються користувачів.

Але, для переходу на трьохзвенну архітектуру найчастіше доводиться розв'язати іншу проблему: як забезпечити взаємодію наявних компонентів, від яких не можна відмовитися по тем або іншим причинам, з компонентами нової системи. Т.е. виникає питання *інтеграції додатків*. Відповіддю на це питання є використання спеціалізованого, т.зв. проміжного, програмного забезпечення — **middleware**.

Визначення й призначення проміжного ПО

Проміжне програмне забезпечення (middleware) — це клас програмного забезпечення, призначеного для об'єднання компонентів розподіленого клієнтсерверного додатка або цілих мережних додатків у єдину інформаційну систему. Проміжне ПО представляє набір сервісів, звертання до яких дозволяє різним додаткам, у загальному випадку, що виконуються на різних платформах, взаємодіяти між собою (рис. 1). Загальні прикладні інтерфейси (API) проміжного ПО дозволяють реалізувати взаємодію між додатками, не глибшаючись в інфраструктуру й деталі реалізації гетерогенної мережі, а наступні зміни в структурі й составі такої мережі не зажадають змін у додатках (за умови, що ці зміни не торкаються API middleware).



Рис. 1. Проміжне програмне забезпечення

Термін **middleware** уперше був використаний в 1968 р., але як технологія інтеграції корпоративних додатків, цей тип програмного забезпечення став використовуватися з 800х років ХХ в. для вирішення проблем сумісності й взаємодії нових додатків із застарілими успадкованими системами.

Місце проміжного ПО — в умовній «середині» між мережними додатками або їх компонентами. Цим воно нагадує середню ланку в трьохзвених клієнт-серверних архітектурах, за винятком того, що функціональні частини **middleware** розподілені між додатками й/або їх компонентами в корпоративній мережній середовищі.

Функції middleware

Сервіси **middleware** представляють додаткам різні функції API, які, у порівнянні з функціями операційних систем і мережних служб, забезпечують:

- прозорий доступ до інших мережних сервісів і додаткам;

- незалежність від інших мережних сервісів;
- високу надійність і постійну готовність.

Слід зазначити, що найчастіше відмінності у функціональності операційної системи й проміжного ПО є умовними. Зокрема, деякі можливості, що раніше представляються винятково засобами middleware, тепер реалізуються на рівні ядра операційних систем. Типовим прикладом є стек TCP/IP, підтримка якого включена практично в усі ОС.

Види проміжного ПО

Затребуваність проміжного ПО при розробці й впровадженні складних систем привела до появи різних типів middleware, які відрізняються масштабованістю, надійністю й самою ідеєю організації взаємодії. При цьому весь існуючий на сьогоднішній день спектр проміжного ПО можна умовно розділити на дві основні категорії (по спеціалізації):

1. Програмне забезпечення для міжпрограмної взаємодії (див. також IPC - Inter-Process Communication).
2. Програмне забезпечення доступу до баз даних.

Проміжна ПО міжпрограмної взаємодії

Протоколи й продукти цієї категорії полегшують міжпроцесну взаємодію (IPC - Interprocess Communications) і розподіл об'єктів в інфраструктурі корпоративної інформаційної системи. Вони виконують роль клею, що дозволяє з'єднати багатоланкові додатки. Основними технологіями є наступні: RPC, MOM, TPM і ORB. У готових рішеннях, як правило, використовується один або кілька таких протоколів.

Концепція **виклику вилучених процедур** (*remote procedure call — RPC*) була розроблена й реалізована в компанії XEROX ще початку 80х років XX століття. Загальний зміст RPC можна представити так: програма може виконувати не тільки власні (скомпільовані) процедури й функції, але й звертатися до процедур вилученого сервера (рис. 2).

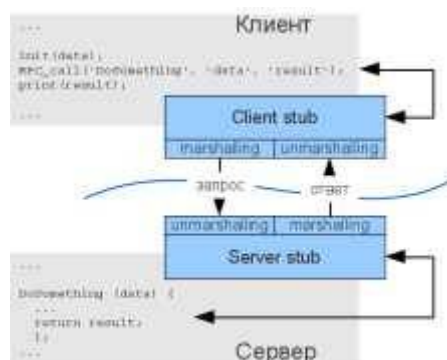


Рис. 2. Виклик вилучених процедур

Для установки зв'язку, передачі виклику й повернення результату клієнтський і серверний процеси звертаються до спеціальних компонентів – «заглушок», відповідно до клієнтської й серверній (client stub і server stub). Ці stub'и не реалізують ніякої прикладної логіки й призначені тільки для організації взаємодії вилучених (у загальному випадку) додатків. Усі RpsОбігу клієнта (імена функцій і списки параметрів) упаковуються клієнтською заглушкою в мережні повідомлення (цей процес називається *marshaling*) і їй же передаються серверній заглушці. Та, у свою чергу, розпаковує отримані дані (*unmarshaling*), передає їхньої реальної функції сервера, а отримані результати впаковує у зворотному порядку.

Клієнтська заглушка ухвалює відповідь, розпаковує його й повертає в додаток. Таким чином, процедури заглушки ізолюють прикладні модулі клієнта й сервера від рівня мережних комунікацій.

Для визначення правил, що задають відносини між клієнтом і сервером RPC, використовується мова опису інтерфейсів IDL. Інтерфейс містить визначення імені функції й повний опис переданих параметрів і результатів виконання. Правила IDL забезпечують незалежність механізму RPC від мов програмування: звертаючись до вилученої процедури, клієнт використовує свої мовні конструкції, а IdlКомпілятор перетворює їх в уніфіковані описи, зрозумілі серверу. На сервері IdlОпису перетворюються в конструкції мови програмування, на якій реалізований серверний процес.

В RPC як засобу організації мережної взаємодії є ряд мінусів, що криються в самій парадигмі структурного програмування:

- Синхронні запити — RPC припиняє виконання додатка доти, поки сервер не поверне необхідні дані.
- Статичні відносини між компонентами — прив'язка клієнтського процесу до серверних заглушок відбувається на етапі компіляції й не може бути змінена під час виконання.

Для усунення цих недоліків були розроблені більш досконалі механізми реалізації RPC:

- асинхронні RPC, що дозволяють уникнути припинення виконання клієнтського додатка за допомогою функцій зворотного виклику (callback functions);
- код заглушок винесений у бібліотеки, що динамічно підключаються.

Протокол взаємодії RPC також може вплинути на роботу системи. Розглянемо гіпотетичний фрагмент коду:

```
for (i = 0, i < 1000, i++)  
    for (j = 0, j < 1000, j++)  
        RPC_call();
```

У цьому випадку виклик вилученої процедури буде виконуватися мільйон раз. На кожній ітерації будуть виконуватися дії не тільки по передачі даних і результатів, але й по керуванню з'єднанням. Це приведе не тільки до підвищення навантаження на мережу, але й зниженню продуктивності системи в цілому. У цьому випадку правильним рішенням є винос функцій установлення й розриву з'єднання за тіло циклів, що суперечить самій ідеї RPC.

Див. також: RFC 1057 Remote Procedure Call Protocol Specification Version 2 (Sun Microsystems), C706 DCE 1.1: Remote Procedure Call (The OpenGroup Distributed Computing Environment (DCE))

Сервіси обробки повідомлень (MOM — *message-oriented middleware*) оце системи, як правило асинхронні, у яких взаємодія між клієнтом і сервером заснована на обміні повідомленнями. *Повідомлення* — це текстові блоки, що полягають із керуючих команд і переданих даних. Для передачі повідомлень використовуються байтоорієнтовані протоколи, такі як HTTP, POP/SMTP і т.п.

Обмін повідомленнями реалізується через API системи MOM. Запити сервісів ставляться в *чергу повідомлень* і обробляються відповідно до пріоритетів і доступності ресурсів (рис.

3).Пріоритети повідомлень дозволяють забезпечити першочергову доставку *важливих повідомлень*, а *відкладена доставка* здійснюється або за розкладом, або з появою адресата в мережі. Відповіді сервера містять інформацію про успішне або неуспішні виконанні операції.

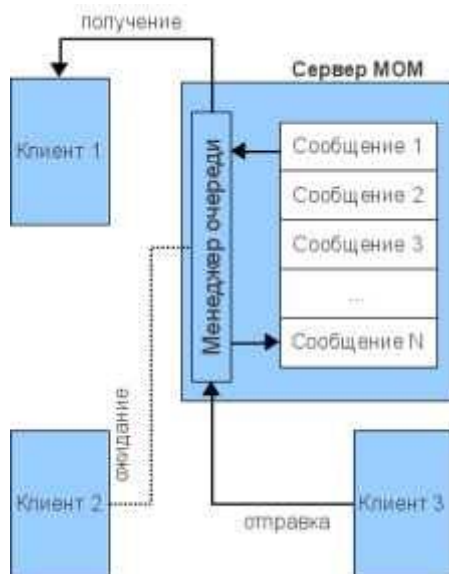


Рис. 3. Проміжне забезпечення, орієнтоване на обробку повідомлень (MOM)

Сервіси MOM добре зарекомендували себе в сильно розподілених додатках, використовуваних у гетерогенній мережі з повільними й ненадійними з'єднаннями. Це досягається завдяки підтримці рівнів «якості обслуговування»:

- надійна доставка повідомлень (reliable message delivery) — система MOM гарантує, що в процесі обміну жодне повідомлення не буде загублено;
- гарантована доставка повідомлень (guaranteed message delivery) — повідомлення доставляється адресатові негайно або через заданий проміжок часу, що не перевищує певного значення (у випадку, якщо мережа в цей момент не доступна);
- застрахована доставка повідомлень (assured message delivery) — кожне повідомлення доставляється тільки один раз.

Черги повідомлень являють собою потужний, гнучкий і в той же час простий механізм міжпрограмної взаємодії.

Крім наведеної, можна сказати класичної, схеми із чергами, розроблені й використовуються сервіси MOM з *безпосередньою передачею повідомлень* і на основі *підписки*.

Системи з безпосередньою передачею повідомлень (message passing) використовують логічне мережне з'єднання для обміну повідомленнями між взаємодіючими додатками. Ця схема зручна в тих випадках, коли клієнти й сервери повідомлень використовуються в сильно зв'язаній мережній інфраструктурі й синхронізовані за часом.

Сервіси MOM, що обслуговують клієнтів по підписці/публікації (publish&subscribe) працюють за принципом, що нагадує поштове розсилення: один додаток публікує інформацію в мережі, а інші підписуються на цю публікацію для одержання необхідних даних. Взаємодіючі таким способом додатки повністю незалежні друг від друга, що представляє можливість динамічної реконфігурації всієї розподіленої системи.

Монітори обробки транзакцій (*Transaction Processing monitors*, Тр-monitors) — це проміжне програмне забезпечення, що забезпечує контроль передачі даних від клієнта при роботі з розподіленими базами даних. Монітор транзакцій забезпечує цілісність даних, стежачи за тим, щоб не було загублених або незавершених транзакцій. *Транзакція* — це закінчений блок звертань до бази даних, для якого гарантується виконання чотирьох умов ACID:

- Атомарність (**Atomicity**) — операції транзакції утворюють неподільний, атомарний блок, який або виконується від початку до кінця, або не виконується взагалі. При неможливості виконання транзакції відбувається відкіт до вихідного стану;
- Погодженість (**Consistency**) — по завершенню транзакції всі задіяні ресурси перебувають у визначеному й погодженому стані;
- Ізольованість (**Isolation**) — одночасний доступ транзакцій різних додатків до поділюваних ресурсів координується таким чином, щоб виключити їхній вплив один на одного;
- Тривалість (**Durability**) — усі модифікації ресурсів у процесі виконання транзакції будуть довгочасними.

Монітор транзакцій забезпечує контроль над виконанням цих умов, виконуючи функції концентрації й преренаправлення запитів до БД у розподіленій середовищі з безліччю баз даних від різних постачальників (рис. 4).



Рис. 4. Монітор транзакцій

Докладний опис відкритої специфікації Distributed TP доступно на сайті Opengroup.

Розподілені об'єктні системи (Distributed object systems) — це проміжне програмне забезпечення, реалізоване у вигляді взаємодіючих один з одним програмних об'єктів. Кожний такий об'єкт унікальним образом ідентифікується в мережі й забезпечує доступ до сервісів, що представляються їм, через публічні властивості й методи. Реалізація об'єкта й платформа, на якій він виконується, повністю прозорі для клієнта.

З погляду розробки, використання розподілених об'єктів оправдано при створенні масштабних об'єктно-орієнтованих систем. Фактично ж, об'єктна «обгортка» таких рішень приховує за собою раніше розглянуті RPC, MOM і засобу керування транзакціями. По-цьому, у першій наближенні загальний принцип взаємодії розподілених об'єктів дуже схожий на RPC (рис. 5). Ця подібність помітна й у засобах опису інтерфейсів — об'єкти використовують усі той же IDL. Однак, у порівнянні з RPC, розподілені об'єкти можуть компонуватися динамічно, тобто не на етапі компіляції, а під час виконання додатків.

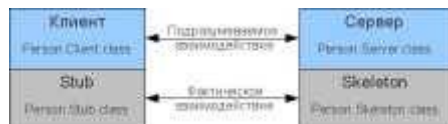


Рис. 5. Розподілені об'єктні системи (stub і skeleton - клієнтська й серверна заглушки)

Архітектура розподілених об'єктних систем стандартизована й найпоширеніші специфікації CORBA, COM/DCOM і EJB.

- **CORBA** (*Common Object Request Broker Architecture*, типова архітектура брокера об'єктних запитів) — відкритий стандарт, розроблений групою Object Management Group (OMG), який визначає інтерфейси між мережними об'єктами, що дозволяють їм працювати спільно. Брокери об'єктних запитів (object request brokers, ORB), створені відповідно до CORBA, представляють інтерфейси для розробки об'єктно-орієнтованих систем «клієнт-сервер».
- Microsoft **COM** (*Component Object Model*, компонентна об'єктна модель) — це сімейство технологій, призначених для організації взаємодії Windows Додатків (см. MSDN: The Component Object Model). У це сімейство входять COM+, DCOM (Distributed COM) і Activex Controls. Microsoft позиціонує COM як платформу для розробки повторно використовуваних (re-usable) компонентів додатків. У випадку DCOM — компонентів розподілених клієнтрсерверних систем.
- **EJB** (Enterprise Javabeans) — технологія, розроблена Sun Microsystems для корпоративних рішень на платформі Java (Java EE/EJB). Специфікація EJB описує архітектуру серверних компонентів і порядок їх використання в клієнтрсерверних додатках. Ця технологія спрощує розробку розподілених систем на основі Java і забезпечує найбільшу переносимість Java Додатків.

Проміжне ПО доступу до баз даних

Розглянемо два основні типи проміжного забезпечення, орієнтованого на роботу з розподіленими даними — це власне проміжне забезпечення СУБД і основне проміжне забезпечення баз даних.

- *Власне проміжне ПО СУБД* — це вбудовані механізми доступу для конкретного сервера баз даних.
- *Основне проміжне ПО баз даних* — до цього типу ставиться, наприклад, інтерфейс Open Database Connectivity (ODBC), який дозволяє програмам «спілкуватися» на різних діалектах SQL через загальні інтерфейси.

Власне проміжне ПО СУБД. До цього виду ставиться, у першу чергу, підтримка стандартів мови SQL. Незважаючи на те, що SQL є міжнародним стандартом, розроблювачі СУБД самостійно визначають, які з можливостей SQL вони будуть підтримувати у своїх системах. У результаті, конкретна СУБД може, з одного боку, не підтримувати або підтримувати частково деякі команди SQL, а з іншого — представляти розроблювачам додатків нестандартні мовні розширення.

Іншим видом власного middleware є вбудовані засоби СУБД, що дозволяють виконувати імпорт даних із зовнішніх джерел (наприклад, з інших СУБД або текстових файлів) і експортувати дані в сторонні формати (наприклад, CSV або XML).

Основне проміжне ПО доступу до БД. До цієї категорії ставляться зовнішні (стосовно СУБД) засоби middleware, спеціально розроблені для звертання до баз даних. Сюди ставляться як технологічні рішення (наприклад, ODBC і JDBC), так і концептуальні, що представляють узагальнену архітектуру інформаційної системи з розподіленими БД (наприклад, EDA або DQB).

- **ODBC** (*Open DataBase Connectivity*, відкрите з'єднання із БД) — це інтерфейс доступу до баз даних, розроблений Microsoft. ODBC представляє можливість звертання до реляційних і нереляційних СУБД у гетерогенній середовищі через уніфіковані виклики API (рис. 6). Звертання до функцій ODBC транслюються в природні діалекти багатьох СУБД за допомогою спеціальних OdbcсДрайверів або прямо, або через вказівку імені джерела даних (рис. 7).

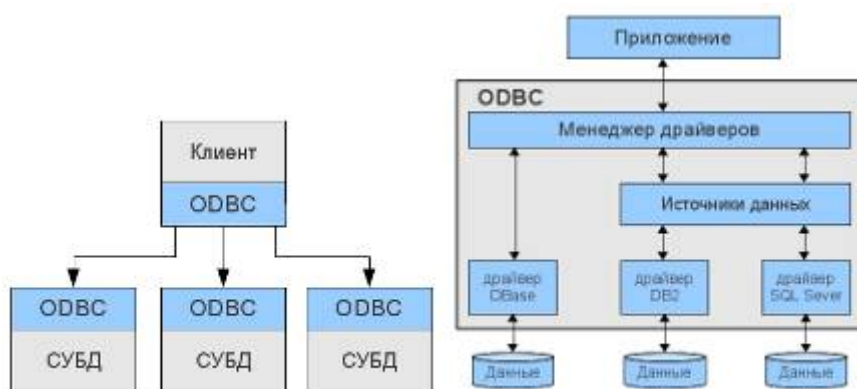


Рис. 6. а) Доступ до БД через ODBC; б) Интерфейс ODBC.

- **JDBC** (*Java DataBase Connectivity*)—це прикладний програмний інтерфейс для звертання до баз даних з Java Додатків. Засіб JDBC є платформозалежним й виконується в будь-якому середовищі під керуванням віртуальної машини Java. У порівнянні з ODBC цей тип проміжного ПО баз даних пропонує більше можливостей по інтеграції додатків і розширює сферу використання БД (наприклад, убік мобільних додатків). Більш того, специфікація JDBC передбачає можливість доступу JavaДодатків до БД через ODBC за допомогою спеціального засобу, т.зв. «моста» (ODBC/Jdbc-bridge).
- **EDA** (*Event-Driven Architecture*, подійно-керована архітектура) — це концепція керування корпоративною інформаційною системою на основі подій, що виникають у бізнесі-процесі. Доступ до БД в EDA базується на використанні централізованого способу звертання до розподілених баз даних. Усі запити клієнтів надходять на вхід SqlлШлюзу, який транслює SqlрЗапити в уніфікований формат і перенаправляє їх до серверів БД. Взаємодія з ними можливо як через набір загальних API, так і через OdbcрИнтерфейс (рис. 8). Використання SqlлШлюзу забезпечує доступ до

даних у різноманітних багатоланкових системах, де безліч додатків паралельно звертаються до джерел даних різного формату.

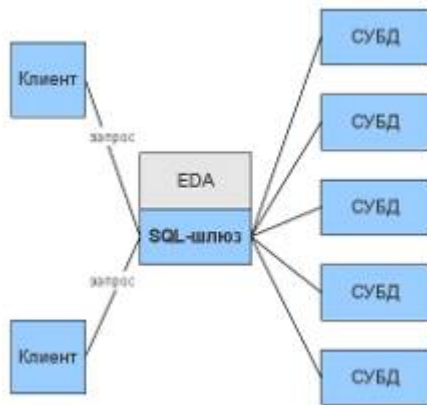


Рис. 8. Принцип доступу до БД в EDA

- **DQB** (*Distributed Query Broker*, брокер розподілених запитів) — децентралізоване (на відміну від EDA) рішення для доступу до БД на основі ORB. Заглушки-Брокери, розміщені на серверах БД, ухвалюють уніфіковані SqlpЗапити й транслюють їх у діалекти конкретних СУБД (рис. 9).

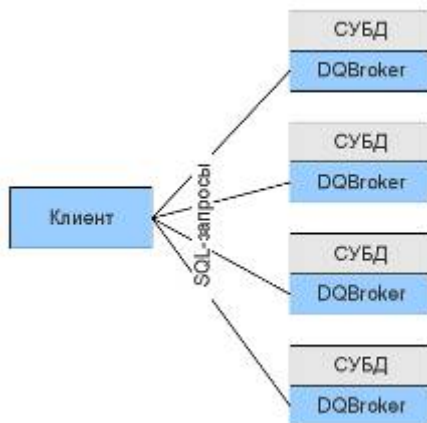


Рис. 9. Брокер розподілених запитів (DQB)

Middleware: Що вибрати?

Стік протоколів TCP/IP давно вже підтримується на рівні операційних систем, засобу RPC також доступні в мережних ОС чи не "за замовчуванням". Цей механізм практично не вимагає від розроблювачів і інтеграторів яких-небудь нових знань і навичок. Не потрібно вивчати специфічні middleware API: виклик вилученої процедури нічим не відрізняється від звертання до локальної підпрограми, а всі процеси перетворення даних і передачі їх по мережі виконуються клієнтської й серверної заглушками. Використання RPC може бути найбільш підходящим рішенням для систем, де можливі проблеми, пов'язані із синхронністю протоколу RPC, не є критичними.

Проміжне ПО обміну повідомленнями пропонує більшу, у порівнянні з RPC, гнучкість, тому що жодне із взаємодіючих додатків не блокується в процесі обміну повідомленнями. Таке рішення краще підійде для автоматизованих систем, компоненти яких слабо зв'язані, працюють у незалежному тимчасовому режимі й використовують різноманітне мережне

середовище. Однак, системи на основі MOM як правило не підтримують автоматичне перетворення форматів (marshalling/unmarshalling), що вимагає від розроблювача рішення завдань по перетворенню форматів даних.

Взаємодія компонентів корпоративних додатків в об'єктно-орієнтованім середовищі зручніше за все реалізовувати за допомогою об'єктних же засобів проміжного ПО. У цьому випадку, розроблювач використовує об'єктну модель, що чітко описує бізнеси-процеси. Високий рівень абстракції об'єктів (будь те ORB або EJB) повністю приховує від користувача деталі реалізації взаємодії між ними. Використання об'єктів дозволяє будувати корпоративні додатки немов лего город, з уніфікованих «цеглинок» об'єктів. Це справедливо, якщо вся система є об'єктно-орієнтованою. Утруднення з використанням розподілених об'єктів з'являються, коли компонентами автоматизованої системи є успадковані додатки.

Монітори транзакцій орієнтовані на складні й високочитичні корпоративні системи. Вони забезпечують необхідний рівень сервісу для таких систем, включаючи керування процесами, балансування навантаження на сервери, засоби резервування відновлення даних. Для завдань автоматизації невеликого підприємства можливості, що представляються ТраМоніторами можуть бути надлишкові. До того ж, продукти цієї категорії мають високу вартість, що може бути економічно не вигідно.

Розглянуті категорії проміжного ПО усе рідше й рідше використовуються в «чистому» виду. Напроти, як і в більшості мережних технологій, розвиток middleware іде у бік об'єднання можливостей різних його категорій і для різного роду завдань завжди можна підібрати найбільш підходяще рішення.

Контрольні питання

1. Опишіть призначення проміжного ПО.
2. Перелічіть основні способи організації міжпрограмної взаємодії (типи проміжного ПО).
3. Опишіть принцип роботи проміжного ПО типу «сервіс обміну повідомленнями».
4. Опишіть принцип роботи сервісу вилученого виклику процедур (RPC).
5. Опишіть принцип роботи проміжного ПО типу «монітор транзакцій».
6. Опишіть принцип міжпрограмної взаємодії на основі об'єктної моделі ORB.

Централізоване керування ресурсами. Служби каталогів

Централізоване керування ресурсами мережі має на увазі концентрацію даних про ці ресурси в базі даних служби каталогів. Каталог зберігає відомості про користувачів, комп'ютери, принтери, мережних сервісах і т.п.

Що таке служба каталогів?

Служба каталогів — це мережний сервіс, що представляє централізовані засоби керування ресурсами автоматизованої системи. Під ресурсами маються на увазі все компоненти мережної інфраструктури, які використовуються для виконання функцій АСУ: користувачі, файли й каталоги, пристрої, мережні сервіси і т.д. (рис.1).



Рис. 1. Служба каталогів

Як правило, служба каталогів складається з бази даних, у якій розміщені відомості про мережні ресурси й серверного ПО доступу, що представляє механізми, до цієї бази. Як база даних сервісу каталогів, так і її керуюча програма можуть бути розподілені на кілька серверів (рис. 2).

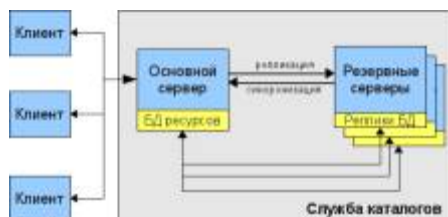


Рис. 2. Розподілена служба каталогів

Основними функціями служби каталогів є наступні:

- Керування користувачами й групами (створення/видалення, налаштування прав доступу).
- Керування ресурсами (вистава в загальний доступ, установка обмежень, вилучене адміністрування й т.п.).
- Розмежування прав доступу (як правило, на рівні користувачів, груп і окремих ресурсів).

Серед додаткових функцій сервісу каталогів можна вказати, наприклад, такі:

- пошук ресурсів;
- поширення мережних політик;
- інтеграція з іншими сервісами.

Мережна політика — сукупність правил, що визначають методи й засоби взаємодії із загальними ресурсами в корпоративній мережі.

Мережні ресурси в службі каталогів звичайно представлені у вигляді ієрархічної структури. Такий спосіб найбільш близький до реальної організаційної моделі переважної більшості підприємств і організацій. Корінь ієрархії описує підприємство в цілому, нижчележачі рівні — підрозділи й окремі елементи. Для однакового звертання до будь-якого елемента ієрархії протокол взаємодії представляє уніфіковану схему адресації — або власну, або сумісну зі стандартними схемами.

Розглянемо основні особливості деяких сервісів каталогів, різних по можливостях, що представляються.

Популярні служби каталогів

Network Information Service (NIS/YP)

NIS (Мережна Інформаційна Служба) — служба каталогів, розроблена й реалізована Sun Microsystems для систем на основі UNIX. NIS спочатку називалися Yellow Pages (YP), але через проблеми з торговельним знаком Sun змінила цю назву. Стара назва (yp) використовується в назвах утиліт NIS.

NIS - це ієрархічна система, у якій існує три типи хостів: *основні (master) сервери*, *вторинні (slave) сервери* й клієнтські машини (рис. 3). У якості джерел даних для клієнтів використовуються системні файли (у термінах NIS — *карти ресурсів*) основного сервера NIS: passwd, hosts, group, services та інші. У мережі може бути один основний сервер NIS і жодного або більш вторинних серверів. Вторинні сервери зберігають копії загальних файлів основного сервера для забезпечення надмірності. Клієнти можуть бути настроєні як на роботу з основним сервером NIS, так і із вторинними. Щоб одержати доступ до запитуваної інформації клієнт повинен бути членом домена NIS.

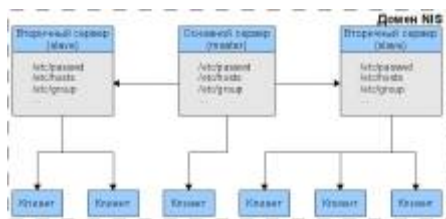


Рис. 3. Структура Network Information Service (NIS)

Домен NIS — це сукупність довірених ресурсів з унікальним у межах мережі іменем. Ім'я домена NIS і спосіб іменування ресурсів нагадує адресацію в систему доменних імен (DNS), але ніякого відношення до DNS не має. Інформація про домене зберігається на основному сервері NIS і реплікується на вторинні сервера нарівні з іншими ресурсами. Один основний сервер може вести бази декількох доменів NIS.

Загальний доступ реалізується за наступною схемою: коли клієнтському додатку потрібен доступ до якогось ресурсу, те локальні обіги (наприклад, до файлу hosts) транслюються у виклики віддалених процедур сервера NIS, з яким зв'язаний клієнт. Оскільки NIS використовує RPC, те NisвСервери працюють на тих портах, які їм представляє сервіс RPC (тобто в служби NIS немає виділеного номера порту, на відміну від, наприклад, ftp-сервера або веб-сервера).

Установлення зв'язку із сервером NIS виконується шляхом широкомовного розсилання запитів. Якщо в мережі є кілька серверів (основний і трохи вторинних), то клієнт NIS (звичайно це програма yrbind) буде використовувати адресу першого, що відповів і направляти всі свої запити на цей сервер. Час від часу yrbind буде перевіряти доступність сервера. Якщо той не відповість за розумний час, то клієнт знову почне процедуру опитування мережі в пошуках «живого» сервера.

Сервіс NIS підтримується більшістю Unix-Систем загального призначення й представляє прості й зручні засоби для організації централізованого керування мережею. NIS добре інтегрується з такими мережними сервісами, як наприклад DHCP або NFS.

Серед основних проблем NIS виділимо дві:

- використанні RPC, що може привести до непрацездатності клієнтів при неприступності NisвСервера;
- передача карт у відкритому виді, що може привести до порушення інформаційної безпеки.

Через другий недолік NIS не рекомендується застосовувати в публічних мережах. Для усунення недоліків NIS були розроблені вдосконалені специфікації протоколу (NYS і NIS+), але вони не настільки популярні.

Вихідну специфікацію NIS можна одержати на сервері Sun Microsystems (тепер Oracle), у розділі документації (Network Information Service), а практичні ради по плануванню й розгортанню цієї служби — наприклад, на сервері Linux Documentation Project (NIS-HOWTO).

LDAP

Модель OSI визначає самі різні аспекти взаємодії відкритих систем. Є в ній і специфікація X.500, що описує службу каталогів. X.500 — це серія рекомендацій розроблених ITU-T в 1993 р., що й одержали статус міжнародного стандарту (ISO/IEC 9594-1). Специфікація X.511 описує протокол DAP для доступу до каталогів X.500. DAP, прийнятий у якості міжнародного стандарту, виявився надлишковим і складним у реалізації й на його основі були розроблені адаптовані версії, серед яких NDS і LDAP.

Протокол LDAP (*Lightweight Directory Access Protocol*, спрощений протокол доступу до каталогу) — бінарний клієнтсерверний протокол прикладного рівня, призначений для доступу до розподіленої служби каталогів через Інтернет. Цей протокол може використовуватися як у якості шлюзу до будь-яких X.500ісумісним каталогам (рис. 4), так і в якості основного сервісу каталогів (рис. 5). Специфікація поточної версії (LDAP v3) наведена в RFC 4510.

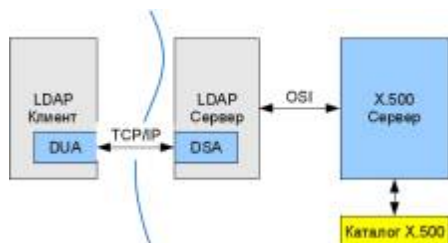


Рис. 4. LDAP у ролі шлюзу до каталогу X.500

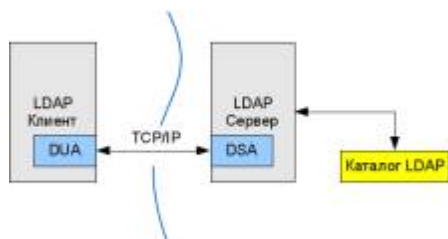


Рис. 5. LDAP у якості самостійного сервісу

У термінах X.500 каталог являє собою «сукупність відкритих систем, що спільно надають служби каталогів». Простіше говорячи, це розподілена клієнтсерверна база даних, доступ до якої можливий через уніфіковані інтерфейси. Користувач каталогу, який може бути людиною або іншим якимсь об'єктом, одержує доступ до каталогу (Directory) за допомогою

клієнтського додатка (DUA). Клієнт взаємодіє з одним або більш серверами через агентів системи каталогів (DSA) (рис. 4).

Ldapустановлює порядок взаємодії між клієнтом і сервером на основі обміну повідомленнями. Повідомлення визначають запитувані клієнтом операції, відповіді сервера й формат переданих даних. LDAP — це сесійний протокол, що вимагає встановлення, підтримки й розриву з'єднання між клієнтом і сервером, тому основним транспортом для нього є TCP. Для сервісу LDAP є стандартний порт 389 (TCP/UDP, див. опис «гарне відомих» (well-known) портів у локальному файлі /etc/services).

Основною одиницею інформації, збереженої в каталозі, є окремий запис (entry). Кожний запис представляє який-небудь реальний об'єкт: людину, комп'ютер, організацію і т.д. Запис описує об'єкт через набір властивих йому атрибутів. Атрибути являють собою пари виду «ім'я — значення». Фактичне значення атрибута залежить від його типу (рис. 6).

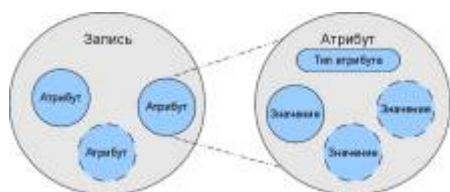


Рис. 6. Записи й атрибути LDAP

Записи зберігаються в ієрархічній структурі, називаній «Інформаційне дерево каталогу» (Directory Information Tree, DIT). Звертання до записів здійснюється по їхнім унікальним іменам (DN, distinguished name). DN включає повний шлях до запису від кореня DIT і цим нагадує шлях до файлу у файловій системі. Крім DN використовується й відносне унікальне ім'я (RDN, relative distinguished name).

Якщо порівнювати DN і RDN з іменами файлів, то DN можна представити так:

```
/home/user/documents/somefile.txt
```

а, відповідно, RDN так:

```
somefile.txt
```

По суті ж, DN є ланцюжок з Rdn'ов елементів структури різного рівня.

В LDAP використовується уніфікована схема іменування й набір обов'язкових і опціональних атрибутів визначених типів, які мають короткі аліаси. Деякі типи атрибутів записів наведені в [табл. 1](#). Простий приклад DIT з декількома записами наведений на [рис. 7](#).

Таблиця 1. Атрибути записів LDAP

Тип атрибута	Короткий (алиас)	запис	Пояснення
Commonname	cn		Загальне ім'я
Stateorprovincename	st		Географічна назва регіону
Organizationname	o		Назва організації
Organizationalunitname	ou		Назва структурного підрозділу
Countryname	c		Країна

Streetaddress	street	Вулиця (як частина адреси)	
domaincomponent	dc	Елемент домена	
userid	uid	Унікальний користувача	ідентифікатор

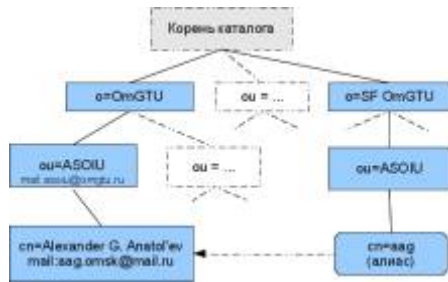


Рис. 7. Приклад (і не більш того!) інформаційного дерева каталогу (DIT) в LDAP

LDAP споконвічно орієнтований на використання в сильно розподіленій середовищі, тому каталог або його частини можуть бути розміщені на різних серверах. Клієнт може звертатися до кожного із серверів LDAP. Доступ до них може здійснюватися по реферальному (рис. 8) або ланцюговому (рис. 9) принципам.

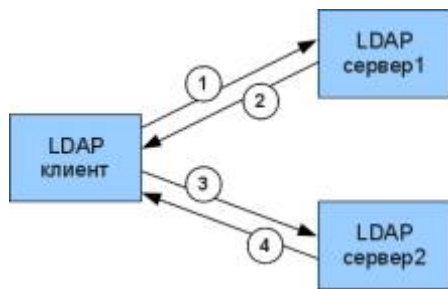


Рис. 8. Реферальне опитування LdapСерверів.

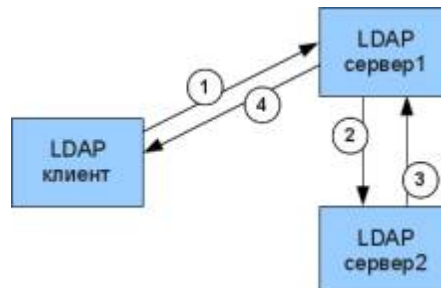


Рис. 9. Опитування « по ланцюжку»

У випадку реферальної обробки запитів LdapСервер повертає клієнтові, замість відповіді, інформацію про те, на якому із серверів можна одержати потрібні дані. При ланцюговій обробці LdapСервер сам опитує інші сервера в пошуках відповіді на клієнтський запит.

Реплікація каталогу LDAP може виконуватися по наступних сценаріях:

- *multi-master* — усі LdapСервери рівноправні, у мережі відсутні чітко означений сервер, керуючий даними;
- *master-slave* — головний (master) сервер управляє всіма змінами каталогу й розсилає їх на підлеглі (slave) сервери. Цей сценарій може бути розширений до делегування функцій синхронізації одному з підлеглих серверів, який і буде реплікувати дані на інші slave-сервери.

Найбільш відомою (але не єдиною) відкритою реалізацією протоколу LDAP є проект Open LDAP.

Active Directory

Починаючи з версії Windows 2000 Server Microsoft стала використовувати власну, LdapтСумісну службу каталогів Active Directory. Про особливості цієї системи централізованого керування ви можете почитати [здесь](#) і [здесь](#).

Дистанційне керування комп'ютером. Термінали й протоколи вилученого керування.

Режим дистанційного керування дозволяє клієнтам управляти роботою вилученого сервера або робочої станції по мережі. Цей режим звичайно використовується сисадмінами для адміністрування виділених серверів і технічної підтримки користувачів.

Вилучений комп'ютер (вилучений хост) — сервер, фізично розміщений на деякій відстані від клієнта й доступ до якого (для клієнтів) можливий тільки по мережі. Для підключення до вилученого хосту використовуються спеціалізовані *протоколи вилученого керування*. Ці протоколи, що реалізують модель « терміналасервер», дозволяють працювати з ресурсами сервера так само, як і з локальними ресурсами: виконувати команди, працювати з файловою системою, запускати додатка й т.п. З поширенням персональних ЕОМ термінальний доступ став використовуватися в основному для рішення завдань вилученого адміністрування.

Термінали

Із середини 19709х до середини 19809х років уся робота за комп'ютером виконувалася через спеціальні пристрої введеннявиводу — термінали. За допомогою терміналів користувачі підключалися до сервера (мэйнфрейму) і використовували його системні ресурси для рішення прикладних завдань.

Споконвічно, **термінал** — це мережний пристрій, підключене до обчислювальної системи й призначений для введення й виводу даних. Команди, прийняті із пристрою введення термінала (клавіатури), передаються на вилучений сервер, де й виконуються. Результати обробки вертаються й відображаються на пристрої виводу термінала (дисплеї). Згодом були розроблені емуляторы терминалов — спеціальні програми, що виконують ті ж завдання.

Таким чином, можна виділити два основні типи терміналів:

1. Реальний, або *фізичний термінал* — як правило має на увазі пристрій, обчислювальні здатності якого обмежені можливістю відображати те, що йому передане з мережі (як максимум — повноекрану графіку).
2. *Віртуальний термінал* — мережний додаток (програма), що виконує функції фізичного термінала. З боку вилученого хоста така програма, нічим не відрізняється від реального термінала.

Однак термінали різняться не тільки за поданням, але й по можливостях. Залежно від них розрізняють:

- *текстові термінали;*
- *графічні термінали;*

- «інтелектуальні» термінали.

Текстові термінали

Текстовий термінал, також відомий як "послідовний термінал", " AsciiТермінал", "символьний термінал" і "відеодисплейний термінал" (VDT, video display terminal) — це пристрій (або програма) введення/виводу, що працює в дуплексному режимі, що й використовує символноорієнтовані (орієнтований^орієнтовані-байт-орієнтовані) протоколи передачі даних, такі як telnet. Потоки даних між терміналом і сервером представляють AsciiіСимволи. Спеціальні символи (або послідовності символів) представляють команди керування сервером і відповіді сервера. Існують і службові команди, що управляють виводом (переклад рядка, напівжирний/похилий текст, звуковий сигнал і т.п.) і сеансом зв'язку (узгодження версій протоколу й ін.).

Текстові термінали відрізняються мінімальними вимогами до смуги пропускання мережі. Це дозволяє дистанційно управляти вилученими серверами (наприклад, у мережі Інтернет), використовуючи повільні з'єднання, що комутируються (модемний зв'язок через порт RS-232C).

«Просунуті» моделі текстових терміналів підтримують т.зв. «блоковий режим», заснований на буферизації даних, що вводяться. Символи, які були отримані із клавіатури, зберігаються в пам'яті термінала (і їх можна відредагувати вбудованим строковим редактором) до натискання клавіші введення/передачі (Enter або щось подібне). Таким чином, серверу передається цілий рядок (блок символів).

«Тупі» (dumb) термінали

Є безліч різних визначень " dumbнтерміналів", які, суть, характеризують максимальні можливостей термінала. Найчастіше під dumbатерміналами мають на увазі текстові термінали, які здатні відображати на екрані тільки алфавітно-цифрові символи.

Графічні можливості текстових терміналів обмежені спеціальними символами ASCII, які можуть бути використані для малювання рамок таблиць, стрілок, горизонтальних і вертикальних ліній, заливань різної щільності і т.д. Використання VGA палітри дозволяє задавати колір тла й тексту. Але все це зводиться до псевдографіки, тому що текстові термінали не управляють виводом на рівні окремих пікселів (Рис. 1).

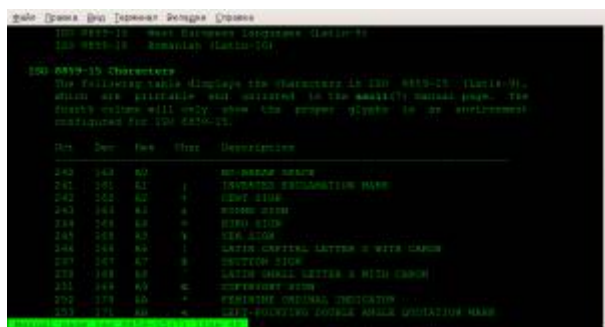


Рис. 1. Емуляція текстового термінала

Графічні термінали

Графічні термінали, на відміну від текстових, представляють можливості роботи з повнобарвною растровою й векторною графікою при введенні й відображенні даних. Такі термінали застосовуються в системах, що підтримують графічний користувацький інтерфейс (GUI, Graphic User Interface). Це дозволяє одержувати дані від користувача не тільки із клавіатури, але й від крапкових пристроїв введення ("миша", тачпад, сенсорний екран і т.п.)

Для графічних терміналів застосовуються два типи дисплеїв:

- *Растрові* - де зображення будується порядково електронним променем (плтДисплеї) або попиксельно (ЖкДисплеї). Растрова графіка використовується в переважній більшості випадків, у силу універсальності й прийнятної вартості подібних пристроїв. Прикладами протоколів графічних терміналів є RDP, ICA, VNC.
- *Векторні* - використовують інтелектуальну електроніку, щоб переміщати електронний промінь не тільки порядково, але й у будь-якому напрямку. Це дозволяє будувати високоякісні зображення без викривлень на пристроях з різною розв'язною здатністю. Основний недолік векторних терміналів — їх дорожнеча, через що вони мало поширені.

Вимоги до пропускної здатності лінії зв'язку при використанні графічних терміналів, у загальному випадку, суттєво вище в порівнянні з текстовими, т. до передавати доводиться не коди символів, а, фактично, скриншоти. Для зниження навантаження на мережу використовуються різні способи: компресія, обмеження на глибину кольору й дозвіл екрана, передача областей, що тільки змінилися, і ін. На рис. 2 - емуляція графічного терміналу на основі X-Window.



Рис. 2. Терминальный доступ по протоколу X11 (X-Window).

Текстовый режим работы также поддерживается графическими терминалами, але реалізований інакше — усередині основного протоколу.

«Інтелектуальні» термінали

Під терміном «інтелектуальний термінал» звичайно мають на увазі апаратно-програмний комплекс, ресурсів якого досить не тільки для виводу повноекранної графіки, але й для виконання деяких обчислень. Принциповою відмінністю інтелектуального терміналу від звичайного ПК є те, що здійсненні програми повинні завантажуватися із сервера, а

виконуватися на клієнтові (самому терміналі). У наведеном трактуванні типовим прикладом інтелектуального терміналу є бездисковая рабочая станция — комп'ютер, у якого відсутній жорсткий диск (а найчастіше й змінні накопичувачі). Завантаження операційної системи й додатків для такого комп'ютера виконується по мережі.

«Тонкі клієнти»

«Тонкий клієнт» (thin client) — різновид інтелектуального терміналу. Вірніше - його програмне забезпечення, яке дозволяє завантажувати й виконувати мережні додатки (клієнтські скрипти, JavaАплети, Activex-додатка й т.п.) у деякому програмнім оточенні (наприклад, у браузері або віртуальній машині).

Протоколи вилученого керування

telnet

Служба вилученого керування **telnet** (Teletype Network) — одна із самих старих мережних технологій Internet (перша специфікація — RFC 158 від 19.05.1971 г.). Поточна специфікація telnet — RFC 854/STD 8 – Telnet Protocol Specification. За замовчуванням сервер telnet ухвалює вхідні підключення на 23-ий порт TCP (див. Сетевые сервисы).

Протокол telnet споконвічно розроблявся для використання в гетерогенних мережах. У його основі — концепція мережного віртуального терміналу (Network Virtual Terminal, NVT) — механізму абстрагування від специфіки введення/виводу різних апаратних і програмних платформ. Так, наприклад, в UNIX у якості символу переходу на інший рядок використовується LF (код 13), у той час як в MS-DOS і Windows — пари символів CR-LF (коди 10 і 13). Мережний віртуальний термінал NVT пропонує використання уніфікованого набору символів, який використовується для перетворення кодів клієнта й сервера.

Хоча в сесії telnet виділяють клієнтську й серверну сторону, протокол є симетричним і обидві сторони взаємодіють через NVT, обмінюючись даними двох типів:

- *Прикладними даними* (тобто даними, які йдуть від користувача до серверного додатка й назад);
- *Опціями* протоколу telnet, службовцями для з'ясування можливостей і переваг сторін.

Прикладні дані проходять через протокол без змін тобто на виході другого віртуального терміналу ми бачимо саме те, що було введено на вхід першого. З погляду протоколу ці дані - просто послідовність байтів (октетів), за замовчуванням приналежних набору ASCII. Якщо ж установлена опція *Binary* — то послідовність будь-яких даних у двійковій виставі.

Усі значення октетів прикладних даних крім 7 (десятькове 255) передаються по транспорті як є. Октет 7 передається послідовністю 77 із двох октетів. Це пов'язане з тим, що октет 7 використовується для кодування опцій протоколу.

Telnet підтримує чотири режими передачі:

- **Напівдуплексний режим.** NVT за замовчуванням цей напівдуплексний пристрій, який вимагає виконання спеціальної команди (GO AHEAD, GA) від сервера, перед тем як буде прийнято введення від користувача. Уведення користувача відображається локальною луною від NVT

клавіатури на NVT принтер, таким чином, від клієнта до сервера посилають тільки повні рядки.

- **Посимвольний.** Кожний символ, що вводиться, відправляється серверу окремо від інших. Сервер відбиває луною (опція ECHO — RFC 857) більшість символів і ці символи відбиваються на екрані клієнта.
- **Блоковий.** Відправлення даних проводиться за принципом "рядок за один раз".
- **Лінійний режим (linemode).** У цьому випадку цей термін означає реальну опцію linemode (RFC 1184). Ця опція обговорюється клієнтом і сервером і коректує всі недоліки в режимі рядок за один раз.

З погляду безпеки протокол telnet уразливий для будь-якого виду атак, до яких уразливий його транспорт, тобто протокол TCP. У протоколі не передбачене використання ні шифрування, ні перевірки дійсності даних. Тому telnet можна використовувати в повністю контрольованій мережі або із застосуванням захисту на мережному рівні (різні реалізації віртуальних приватних мереж). Через ненадійність від telnet як засобу керування операційними системами давно відмовилися.

Іноді клієнти telnet використовуються для доступу до інших символьноорієнтованим протоколам на основі транспорту TCP (HTTP, IRC, SMTP, POP3 і прочим). Однак, використання клієнта telnet у цій якості може привести до помилок, пов'язаних з неправильним трактуванням сервером опцій протоколу (лістинг 3).

Лістинг 1. Використання telnet у якості веб-браузера

```
aag@stilo.asoiu:~> telnet 4stud.asoiu 80
Trying 194.188.210.1...
Connected to 4stud.asoiu.
Escape character is '^['.

HTTP/1.1 400 Bad Request
Date: Wed, 03 Oct 2012 06:01:43 GMT
Server: Apache/2.2.17 (Linux/SUSE)
Content-Length: 307
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.2.17 (Linux/SUSE) Server at 4stud.asoiu Port 80</address>
</body></html>
Connection closed by foreign host.
```

rlogin

Протокол rlogin (англ. *remote login* — вилучений вхід у систему, RFC 1282, порт сервера - 513) дозволяє користувачам робочих станцій UNIX підключатися до вилучених серверів UNIX через мережу Internet і працювати так само, як при прямій підключенні термінала до машини. Крім rlogin існують ще кілька подібних протоколів: rsh (remote shell), rcp (remote copy). Утиліти rlogin, rsh і rcp часто поєднують під загальною назвою rкоманд.

Що вперше з'явився в складі 4.2BSD UNIX, програма rlogin (і інші гдкоманди) деякий час була винятково популярною в цьому середовищі. У якості засобу термінального доступу rlogin дуже схожа на telnet, але через тісну інтеграцію з ОС знайшла обмежене застосування в інших системах.

В rlogin відсутні багато опцій, властиві telnet, зокрема режим узгодження параметрів між клієнтом і сервером. Однак rlogin передбачає хоча б мінімальні засоби захисту на основі довірчих відносин між хостами: на сервері rlogin у спеціальних системних файлах (звичайно /etc/hosts.equiv і \$HOME/.rhosts) адміністратор може перелічити комп'ютери, доступ з яких до даного сервера буде дозволений без пароля. Користувачі інших комп'ютерів (не перерахованих у цих файлах) можуть увійти на сервер лише після введення пароля (який як і в telnet передається у відкритому виді). Але, довірчі відносини між хостами теж не панацея, і можуть установлюватися лише в ізольованих мережах. Справа в тому, що такі техніки несанкційованого доступу, як підміна ІрсАдрес (Ip-spoofing) і доменних імен (Dns-spoofing), роблять гссервіси в Інтернет незахищеними. Тому сучасні дистрибутиви UnixoПодібних ОС включають не фактичні гдкоманди, а їх, що заміщають посилання на, SshлАналоги: scp, sftp і ін.(див. man rlogin в Ubuntu, що виводить посібник з Openssh).

SSH (Secure Shell)

SSH (англ. Secure Shell - безпечна оболонка) — безпечний протокол дистанційного керування комп'ютером. Основна специфікація - RFC 4251. The Secure Shell (SSH) Protocol Architecture, вхідний порт - 22. У порівнянні із протоколами telnet і rlogin, ssh відрізняється, у першу чергу, високою захищеністю за рахунок підтримки криптостійких алгоритмів шифрування й рядом додаткових можливостей.

SSH дозволяє не тільки використовувати захищену оболонку на вилученому хості, але й тунелювати графічний інтерфейс (X11 forwarding), перетворюючи клієнтів, що використовують X-Window System, у графічні термінали. SSH також здатний (при належному конфігуруванні) передавати через безпечний канал будь-який інший мережний протокол - port forwarding. Криптографічний захист протоколу SSH не фіксована, можливий вибір різних алгоритмів шифрування.

Згідно з Указом Президента Російської Федерації № 334 від 03.04.95 фізичним особам і організаціям, включаючи державні, частки й акціонерні, заборонена експлуатація систем криптографії, що не пройшли сертифікації у ФАПСИ. А SSH (а йже з ним PGP) є саме такою системою.

Не варто думати, що нам намагаються заборонити захищати конфіденційну інформацію: організації не тільки можуть, але й зобов'язано захищати важливу інформацію. Тільки для цього вони повинні застосовувати сертифіковані засоби. Застосовуючи вільне, але несертифіковане ПО на основі ssh, SSL, PGP і т.п. слід пам'ятати, що його використання чревате розглядом з боку спецслужб.

Сервіс SSH використовує за замовчуванням порт 22 і поєднує протоколи трьох рівнів:

- **Протокол аутентифікації** (The Secure Shell (SSH) Authentication Protocol, RFC 4252) — цей рівень надає кілька механізмів для аутентифікації користувача: тут може використовуватися традиційна парольна аутентифікація, аутентифікація, заснована на публічному ключі і т.д.
- **Протокол транспортного рівня** (The Secure Shell (SSH) Transport Layer Protocol, RFC 4253) — забезпечує взаємодія алгоритму й обмін ключами. Обмін ключами дозволяє аутентифікувати сервер і створити

криптографічно захищене з'єднання: забезпечуючи цілісність, конфіденційність і додатковий стиск.

- **Протокол з'єднання** (The Secure Shell (SSH) Connection Protocol, RFC 4254) — створює безпечний (шифруємий) канал, представляючи його у вигляді декількох логічних каналів, які використовуються для різних цілей (різних видів служб).

Підтримка SSH реалізована у всіх UNIX системах і на більшості з них у типову поставку включене як клієнтське, так і серверне ПО (див. [man ssh](#)). Для не-UNIX систем є як платне, так і безкоштовне клієнтське ПО (Securecr, putty). Серверне ПО - платне.

X-Window System

[актуальну інформацію див. в [man Xorg](#)]

Практично всі завдання керування UNIX можуть виконуватися в текстовому режимі, однак адміністратори нерідко віддають перевагу графічному інтерфейсу, як більш зручному. До того ж, деякими додатками UNIX можна управляти тільки в графічній середовищі. Програмне забезпечення X-server, відповідальне за вивід графічної інформації, є для безлічі платформ, включаючи DOS, Windows, Macintosh, UNIX і т.д.

Слід мати у виді, що застосування X Window System припускає наявність достатня великої пропускної здатності мережі. Система прекрасно працює в локальних мережах, але дуже повільно — по глобальних каналах. Тому при використанні X Window System на домашньому комп'ютері адміністратора керування краще здійснювати через термінальні утиліти на зразок xterm, а не за допомогою графічних утиліт.

При підключенні до сервера UNIX (на якому запускаються клієнти X11) аутентифікація може здійснюватися двома методами: через термінальні утиліти (telnet, rlogin і т.п.) і через менеджер дисплеїв X (X Display Manager, xdm). У першому варіанті передачі пароля у відкритому виді можна уникнути, застосовуючи замість telnet і rlogin уже згадувані програми ssh і OTP. У випадку X Display Manager паролі за замовчуванням передаються у відкритому виді. Тому при вилученні керування сервером UNIX по загальнодоступних мережах xdm користуватися не варто.

Дуже обережно адміністратори повинні підходити уводити, увести до ладу питанню використання сервера UNIX у якості сервера X (тобто, говорячи зрозумілою мовою, до запуску графічної оболонки X11 на сервері UNIX). X Window System улаштована так, що користувач може зі своєї машини запустити клієнта X на вилученому сервері X і перехоплювати на ньому введення/вивід інформації. У результаті зловмисник одержує можливість зчитувати конфіденційну інформацію із сервера X, включаючи паролі, що вводяться користувачем на сервері X (хоча емулятор терміналу xterm дозволяє блокувати перехоплення пароля, цією можливістю рідко хто користується).

На серверах X застосовуються дві схеми аутентифікації клієнтів: по імені хоста й за допомогою «магічних булочок» (MIT-MAGIC-COOKIE-1). При аутентифікації по імені хоста на сервері X створюються системні файли, де перелічуються хости, звідки дозволено запускати клієнтські програми X на даному сервері X. Але подібний захист ніяк не назвеш достатньої, тому що за допомогою підміни ІрсАдрес або доменних імен зловмисник може провести атаку на X11. При використанні ж схеми «магічних булочок» (їх підтримка вбудована до протоколу XDMCP, на основі якого функціонує X Display Manager) аутентифікація здійснюється на підставі облікових записів користувачів. Щоб мати право запустити клієнта на сервері X, користувач у своєму домашньому каталозі машиниклієнта

X11 повинен мати системний файл із записаним секретним кодом сервера X. Цей секретний код і називається магічною булочкою. Лихо тільки в тому, що булочка передається по мережі у відкритому виді, тому даний метод також чи навряд можна вважати безпечним.

В X Window System 11 Release 5 додані ще дві схеми (XDM-AUTHORIZATION-1 і SUN-DES-1), що нагадують схему MIT-MAGIC-COOKIE-1, але, що використовують алгоритм шифрування DES. Однак через експортні обмеження такі схеми в комплект поставки X Window System не включають. Виходячи з вищенаведених міркувань, запускати серверне ПО X11 на сервері UNIX можна лише в тому випадку, коли заборонений доступ клієнтів X11 з інших комп'ютерів.

Remote Desktop Protocol *

RDP (англ. *Remote Desktop Protocol*, протокол вилученого робочого стола) — відкритий протокол прикладного рівня, розроблений Microsoft і споконвічно призначений для підключення графічних терміналів до Windows Terminal Server. Клієнти існують практично для всіх версій Windows (включаючи Windows CE і Mobile), Linux, Free BSD, Mac OS X. По-умовчання використовується порт TCP 3389. Офіційна назва Майкрософт для клієнтського ПО - Remote Desktop Connection або Terminal Services Client (TSC), зокрема, клієнт в Windows XP/2003/vista називається mstsc.exe.

RdприКлієнт, - Remote Desktop - дозволяє вам, перебуваючи на одному комп'ютері, віддалено управляти іншим. Наприклад, якщо вам потрібно зайти у свій комп'ютер, що перебуває в офісі, з будинку (припустимо, що ви завчасно настроїли свій робочий комп'ютер), те ви можете за допомогою інструмента Remote Desktop одержати доступ до всіх даних, що перебувають на робочому комп'ютері, включаючи файли, додатки й мережні з'єднання. Ви навіть можете чути звук, з яким відкриваються файли (запитується: навіщо? прим. aag).

Фактично Remote Desktop дозволяє не тільки одержувати доступ до файлів вилученого комп'ютера, але й насправді бачити робочий стіл таким, який він є на вилученому комп'ютері. Більше того, якщо вилучений комп'ютер працює в операційній системі Windows 2000 або .NET Server, те на ньому можуть віддалено працювати декілька користувачів одночасно.

Технологія Terminal Services є основою для роботи вилученого помічника, який дозволяє вашому другові або працівникові технічної підтримки встановлювати з'єднання з вашим комп'ютером, бачити ваш робочий стіл і управляти комп'ютером.

Для з'єднання інструмент Remote Desktop використовує LAN, віртуальну приватну мережу (VPN) або інтернет-з'єднання. Робота вилученого робочого стола сильно залежить від швидкості встановленого з'єднання.

Remote Desktop підтримує роботу у двох режимах:

- Remote Desktop (Вилучений робітник стіл) - підходить для використання в локальній мережі й вимагає установки програмного забезпечення на комп'ютері-клієнтові.
- Remote Desktop Web Connection (ІнтернетПідключення до вилученого робочого стола) - вимагає на клієнтській машині тільки наявності браузера Internet Explorer, але на сервері для неї необхідно встановити й настроїти більшу кількість програм.

Remote Desktop підтримує 24-бітні кольори - це дозволяє варіювати якість картинки в широких межах.

Переадресація ресурсів дозволяє, наприклад, використовувати файлову систему вилученого комп'ютера як мережного ресурсу загального доступу.

Переадресація звуків дозволяє комп'ютеру-клієнтові відтворювати звуки, які генеруються на комп'ютері-сервері. При програванні звуків Remote Desktop також ураховує пропускну здатність смуги частот. Замість того щоб перевантажувати з'єднання звуковим сигналом при зміні пропускну здатності, Remote Desktop знижує якість звуку.

Remote Desktop Сервер і Remote Desktop Клієнт користуються загальним буфером. Це дозволяє їм вільно обмінюватися інформацією.

Веб-технології вилученого керування

Всесвітня Павутина (World Wide Web) виявляє все більший вплив на засоби керування мережним середовищем. Уже зараз багато маршрутизаторів, комутатори, мережні принтери допускають керування через web-браузери. Але список цей далеко не вичерпується ними й Web вторгається й у сферу керування мережними ОС. Спочатку з Web можна було управляти лише серверами HTTP і FTP, але цей список постійно розширюється й охоплює тепер СУБД (наприклад, phpmyadmin як веб-інтерфейс до СУБД Mysql), файлові системи, міжмережні екрани, мережні служби DNS, DHCP і багато чого іншого (приклад - Webmin для керування Unix-подібними системами). Незважаючи на все, що збільшується кількість подібних рішень, до повнофункціональних спеціалізованих адміністративних систем вони не доросли. Основна проблема тут у тому, що для багатьох додатків і, особливо, мережних пристроїв пароль по HTTP передається у відкритому виді.

Бездискові робочі станції. Выддалене подання.

Бездискова робоча станція — це персональний комп'ютер, позбавлений незнімних засобів для довгочасного зберігання даних. Існують два основні сценарії використання бездискових робочих станцій:

- Операційна система й додатки завантажуються по мережі із сервера й потім виконуються локально на робочій станції. Результати роботи (наприклад, документи, з якими працює користувач) зберігаються також на сервері, або на якому-небудь знімному пристрої. Замість сервера для завантаження робочої станції може використовуватися знімний носій, такий, як компакт-диск (так, наприклад, працює Knoppix).
- Операційна система й додатки виконуються на сервері, а результати їх роботи (користувацький інтерфейс додатків) передаються на робочу станцію й відображаються так само, як якби користувач працював із цими додатками прямо. Такі робочі станції називаються *терміналами*, а сервери, на яких виконуються ОС і додатки — *серверами терміналів*. Термінали вимагають мінімальної обчислювальної потужності, тому що самі вони не обробляють дані, а лише відображають інтерфейс. З іншого боку, підвищена продуктивність потрібно від серверів терміналів.

Основними перевагами використання бездискових робочих станцій є централізоване зберігання всіх даних, що дозволяє легко управляти ними, робити резервне копіювання і т.д. З іншого боку, якщо бездискова станція завантажується із сервера (або є терміналом), то вона непрацездатна без справно працюючого мережного підключення й сервера.

Основна маса рішень на користь використання бездискових станцій, приймається іт - персоналом, з економічної сторони поставленого завдання, оскільки апаратні вимоги як правило низки й можна збирати станції практично з "утилю" 5-7 літньої давнини. Більшість великих виробників серверних рішень роблять термінали (тонкі клієнти) на сучасних, але низькопродуктивних комплектуючих. Тонкі клієнти як правило споживають менше енергії, підтримують більшість сучасних інтерфейсів і не мають, що рухаються елементів (вентиляторів, жорстких дисків), що дозволяє збільшити строк роботи тонких клієнтів.

Сучасні операційні системи використовують свопінг. У випадку бездискового комп'ютера своп-файл доводиться розміщати на сервері, у результаті мережа сильно навантажується додатковим трафіком. Тому бездискові робочі станції були популярні в епоху MS-DOS, яка не використовує свопінг; у цей час їх звичайне використання у якості терміналів (у цьому випадку на них запускається тільки одна програма, що практично не звертається до дискової пам'яті). Іноді бездискові станції все-таки постачають жорстким диском невеликої ємності для розміщення тільки своп-файла. Останнім часом достатнє поширення одержали т.зв. апаратні реалізації термінальних протоколів як термінальні рішення. Автономний пристрій(коробочка) до якого підключені пристрої введення/виводу й організований доступ до термінального сервера по мережі.

PXE

Схема завантаження з використанням PXE

PXE (англ. *Preboot Execution Environment*, вимовляється *пiксі*) -- середовище для завантаження комп'ютерів за допомогою мережної карти без використання жорстких дисків, компакт-дисків і інших пристроїв, застосовуваних при завантаженні операційної системи. Для організації завантаження системи в PXE використовуються протоколи IP, UDP, DHCP і TFTP. Рхе-Код, прописаний у мережній карті, одержує завантажник з мережі, після чого передає йому керування.

Завантаження Linux Системи з використанням PXE

Завантаження Linux системи за допомогою PXE можна виконати за допомогою завантажника PXELINUX, створеного на базі SYSLINUX. PXELINUX завантажується за допомогою РхекКоду в мережній карті, а потім йому передається керування. Код PXELINUX не потрібно прописувати в чип на мережній карті (якщо потрібно саме це, див. проект etherboot).

- **рхе - Завантаження РхечЗавантажника.** Прошиванням PXE, вбудованої в мережну карту, виконується завантаження завантажника **pxelinux**.
- **dhcр1 - Одержання ІрeАдреси й адреси TftpрСервера.** Завантажник pxelinux робить DhсpЗапит і з його допомогою одержує ІрpАдресу й адреса TftpрСервера. За замовчуванням адреса TftpрСервера дорівнює адресі DhсppСервера.

- **tftp** - Звертання до TftpServerу для одержання ядра Linux. Завантажник rхelinux звертається до TftpServerу й запитує в нього ядро Linux (і, при необхідності, образ initrd).
- **kernel** - Запуск ядра Linux. Після того як ядро Linux завантажено, керування передається йому.
- **dhcp2** - Одержання ІрeАдреси ДНСР. Ядро Linux робить запит ДНСР, за допомогою якого одержує свій ІррАдресу, адресу NfsvСервера, на яким перебуває коренева файлова система, а також шлях до місця розташування цієї файлової системи на диску.
- **nfs** - Монтування кореневої файлової системи. Коренева файлова система монтується.
- **init** - Виклик процесу init. На кореневій файловій системі перебуває файл /sbin/init, якому передається керування.

Термінальний сервер

Термінальний сервер — від англ. terminal server. Він же — сервер терміналів.

Термінальний сервер — сервер, що надає клієнтам обчислювальні ресурси (процесорний час, пам'ять, дисковий простір) для рішення завдань. Технічно, **термінальний сервер** являє собою дуже потужний комп'ютер (або кластер), з'єднаний по мережі з термінальними клієнтами — які, як правило, являють собою малопотужні або застарілі робочі станції.

Процес роботи

Термінальний клієнт після встановлення зв'язку з термінальним сервером пересилає на останній, що вводяться дані (натискання клавіш, переміщення миші) і, можливо, надає доступ до локальних ресурсам (CD-ROM, флорідиск, принтер, жорсткий диск). Термінальний сервер надає середовище для роботи (термінальна сесія), у якій можна виконувати ресурсомісткі додатки. Результат роботи сервера передається на клієнта, як правило, це зображення для монітора.

Переваги термінального сервера

- Істотна економія на програмнім і апаратнім забезпеченні для робочих станцій
- Зниження тимчасових витрат на адміністрування
- Підвищення безпеки — зниження ризику інсайдерських зломів

Види термінальних серверів

- Microsoft Windows Terminal Server (поставляється в Microsoft Windows Server)
- Citrix Metaframe
- X-Window
- 2X Terminalserver (англ.)

Термінальний режим роботи

Термінальний режим роботи - організація мережної роботи інформаційної системи (ІС) за допомогою розміщення всіх користувацьких додатків і даних на центральному сервері (серверах), доступ до яких здійснюється з машинитерміналів (см. також робоча станція, робочий термінал, "тонкий клієнт"), виготовлених у спрощенім виконанні й, як наслідок, більш дешев, що займають мінімум місця, безшумних і практично не потребуючих обслуговування.

Термінальний режим роботи крім зниження вартості ІС, програмного забезпечення й витрат на обслуговування, дозволяє забезпечити більш високий рівень безпеки й простоту масштабування ІС.

Тонкий клієнт

У комп'ютерних технологіях **тонкий клієнт** (англ. *thin client*) — це комп'ютерклієнт мережі із клієнтр серверною архітектурою, який переносить більшість завдань по обробці інформації на сервер. Таким чином, сервер необхідний для нормальної роботи тонкого клієнта. Цим тонкий клієнт відрізняється від товстого клієнта, який, навпаки, робить обробку інформації незалежно від сервера, використовуючи останній в основному лише для зберігання даних. Прикладом тонкого клієнта може служити комп'ютер із браузером, що використовується для роботи із застосуваннями-прикладеннями-веб-додатками.

Тонкий клієнт у більшості випадків має мінімальну апаратну конфігурацію, іноді — без жорсткого диска. У деяких конфігураціях системи **тонкий клієнт** завантажує операційну систему по мережі із сервера, використовуючи протоколи PXE, BOOTP, DHCP, TFTP і Remote Installation Services (RIS).

Протоколи, використовувані тонкими клієнтами

- X11 — використовується в Unix
- Telnet — мультиплатформний
- SSH — мультиплатформний захищений аналог Telnet
- **NX technology** — протокол X11 зі стиском даних
- **Virtual Network Computing**
- **Citrix ICA**
- *Remote Desktop Protocol (RDP)*, протокол для вилученої роботи з використанням графічного інтерфейсу користувача для Microsoft Windows

Приклади тонких клієнтів

- Thinstation
- LTSP
- Openthinclient
- Windows CE

Специфікації

- RFC 1350 - опис протоколу TFTP (англ.)
- RFC 2131 - опис протоколу DHCP (англ.)
- RFC 854 - опис протоколу telnet (англ.)

- RFC 951 і RFC 1542 - опис протоколу DHCP (англ.)

Remote Desktop Protocol

Специфікація ICA

Interapplication Communication Architecture – Архітектура міжпрограмних зв'язків, специфікація ICA. Специфікація, що передбачає організацію ефективної роботи мережних комп'ютерів і тонких клієнтів; 2. Independent Computing Architecture – архітектура незалежних обчислень, зв'язувальний протокол ICA — протокол взаємодії з тонкими клієнтами (см. також RDP, thin client); 3. Intelligent Console Architecture – архітектура інтелектуальних консолей, протокол ICA [фірми Citrix] — дозволяє виконувати програми Windows на вилученому сервері, у те час як їхній інтерфейс користувача відображається на будь-якому не Windows комп'ютері, у якого є клієнт ICA. Це можуть бути машини Unix і Macintosh, багато інтелектуальних терміналів, мережні комп'ютери й навіть кишенькові ПК. Подібно системі X Windows, для відображення інтерфейсу користувача в ICA на термінал передається потік компактних команд, а не растр цілого екрана, завдяки чому можна працювати по відносно низькошвидкісним каналам зв'язку, наприклад 10 Мбіт/с Ethernet або навіть по модемним з'єднанню.

Тонкі клієнти й графічний інтерфейс

Мода на схуднення, здається, докотилася й до мережних технологій. Останнім часом багато уваги стало приділятися перспективам ринку тонких клієнтів взагалі й мережних комп'ютерів (МК) зокрема. Однак під час обговорення можливостей тих або інших видів тонких клієнтів основний акцент робиться на зниження вартості програмно-апаратних засобів і зменшення витрат на адміністрування. При цьому специфічні вимоги корпоративних і індивідуальних замовників часто відходять на другий план.

У даному огляді мова йтиме про особливості багатокористувацького графічного інтерфейсу в середовищі тонких клієнтів.

БАГАТОКОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС

Багатокористувацький інтерфейс завжди був характерною рисою систем старшого й середнього класу (мэйнфреймів, суперкомп'ютерів, міні-комп'ютерів, серверів корпоративного рівня). По-перше, він дозволяє використовувати обчислювальну потужність системи одночасно декільком користувачам, і, по-друге, користувачі можуть працювати із центральною машиною безпосередньо на своєму робочім місці через настільний ПК або термінал.

Графічний інтерфейс додає нові функціональні можливості. Насамперед це відноситься до додатків САПР, програмам інженерних і наукових розрахунків, геоінформаційних систем та ін.

Але багатокористувацький графічний інтерфейс уже давно перестав бути приналежністю тільки центральних обчислювальних машин. Збільшення потужності серверів на базі ПК, поява для них потужних 32-розрядних операційних систем, розвиток Web-Технологій зробили проблему багатокористувацького графічного інтерфейсу актуальною й на ринку ПК. Виконання багатьох офісних додатків можна перенести із ПК на сервер, спрощуючи й здешевлюючи в такий спосіб клієнтські місця.

Багатокористувацький графічний інтерфейс важливий і для системних адміністраторів. Він дозволяє контролювати й управляти серверами з робочого місця адміністратора. Зрозуміло, управляти серверами можна, і не використовуючи такий інтерфейс, за допомогою спеціалізованих систем керування. Однак подібні системи мають занадто багато обмежень і не можуть іти в порівняння з роботою безпосередньо в середовищі конкретного сервера.

Усі серверні операційні системи (Windows NT Server, UNIX, Netware, VMS, OS/400, OS/390 і ін.) є багатокористувальними. Правда, випадок з Netware особливий. Ця ОС не є системою загального призначення й у силу своєї специфіки погано підходить для ролі сервера додатків. Багатокористувацький графічний інтерфейс для доступу до сервера Netware, по великому рахунку, не потрібний.

Що стосується Windows NT Server, те це ще один особливий випадок. Хоча вона повністю багатокористувальна, вбудований багатокористувацький інтерфейс в NT відсутній. Проте продукти третіх фірм дозволяють організувати багатокористувацький доступ до сервера, у тому числі за допомогою тонких клієнтів. Більше того, починаючи з NT версії 5.0 Microsoft обіцяє реалізувати вбудоване багатокористувальне середовище.

ТОНКІ КЛІЄНТИ

Тонкими клієнтами звичайно називають комп'ютери з обмеженими ресурсами, наприклад комп'ютери, у яких відсутні вінчестери (бездисківні станції), з невеликим обсягом оперативної пам'яті, з малопотужним процесором і т.д. Як виразився один комп'ютерний фахівець, тонкий клієнт - це недороблений комп'ютер. Відмінна риса тонкого клієнта в тому, що для своїх потреб він задіє ресурси інших машин (серверів).

Сам термін "тонкий клієнт" з'явився порівняно недавно, хоча подібні пристрої відомі давно. До тонких клієнтів відносять символічні (алфавітно-цифрові) термінали, ХлТермінали, бездисківні станції, мережні комп'ютери, WindowseТермінали й клієнти Winframe, а також комп'ютери, що задовольняють специфікації Netpc компаній Intel і Microsoft.

Усі ці клієнти різняться товщиною, тобто набором програмно-апаратних засобів - чому їх більше, тем товстіше клієнт.

Деякі типи тонких клієнтів не можуть працювати із графічною інформацією, зокрема це ставиться до символічних терміналів. Такі пристрої домінували в епоху панування мейнфреймів і міні-комп'ютерів, але зараз попит на них досить обмежений.

У цій статті ми розглянемо більш докладно окремі види тонких клієнтів, їх відмінні риси, переваги й недоліки при доступі до графічної інформації (див. Таблицю 1).

ТАБЛИЦЯ 1 - ХАРАКТЕРИСТИКИ ТОНКИХ КЛІЄНТІВ

Тип	ХаТермінал	Бездисківна станція Netware	СК	Netpc	Клієнт Winframe
Продуктивність мережі	10 Мбіт/з	2 Мбіт/з	2 Мбіт/з	2 Мбіт/з	20 Кбіт/з
Оперативна пам'ять	4 Мбайт	4 Мбайт (Windows)	4 Мбайт	16 Мбайт	640 Кбайт
Ціна, доларів	2000	500	700	1000	400

X ТЕРМІНАЛИ Й X WINDOW SYSTEM

ХлТермінали - це мережні пристрої, що працюють у середовищі X Window System. Дану систему розробили в Массачусетському технологічному інституті в 1984 р. В 1987 р. вийшла остання, 11, версія X Window System (позначається X11), у якій були усунуто багато обмежень попередніх версій.

X Window System є мережною графічною системою, що використовує власний протокол X. X11 являє собою розподілену систему, що працює в архітектурі клієнт-сервер і не залежну ні від платформи, ні від операційної системи.

У розріз із загальноприйнятою термінологією в X11 машина, на якій виконується ХлДодаток, називається ХоКлієнтом, а комп'ютер, з якого здійснюється введення/вивід інформації, - ХоСервером. Таким чином, ХаТермінал виконує роль ХрСервера. X Window System (ХнКлієнт і Х-сервер) являє собою надбудову над певної ОС, якої може бути UNIX, Windows, Mac OS і навіть MS-DOS. Для ХлТерміналу в якості ОС часто використовують урізану версію UNIX. ХнКлієнт і Х-сервер можуть виконуватися як на одній машині, так і на різних комп'ютерах із зовсім несумісними ОС. Більше того, на одному ХрСервері може відображатися (і вводиться) інформація з декількох ХоКлієнтів, що виконуються одночасно на різних комп'ютерах. І навпаки, один ХнКлієнт може відображатися відразу на декілька ХаСерверів.

Як уже було сказано, X11 використовує власний протокол X, який є протоколом прикладного рівня. У якості транспорту він може задіяти фактично будь-який мережний протокол, зокрема TCP/IP, Decnet, IPX/SPX і ін. Інтерфейсом до ХлПротоколу з боку ХтКлієнта на прикладному рівні служить бібліотека Xlib.

З боку ХрСервера взаємодія з ХоПротоколом забезпечують драйвери пристроїв ХрСервера.

На Малюнку 1 представлена схема роботи X Window System. ХтКлієнти й Х-сервери спілкуються один з одним, пересилаючи пакети інформації з ХлПротоколу: запити, відгуки, події й помилки. Запити генеруються бібліотекою Xlib і передаються на ХеСервер. Запити можуть містити широкий спектр інформації, наприклад пропозиція намалювати лінію, змінити колір тла або перемістити вікно. У відповідь на запити ХеСервер генерує відгуки, що направляються в Xlib і потім ХтКлієнтам. У відгуку ХеСервер повідомляє, яку операцію він виконав.

(1x1)

Малюнок

1.

Взаємодія X Клієнтів і Х-серверів. На одній машині може бути запущено трохи X Додатків і один X Сервер.

Одним з найважливіших типів пакетів є події. Найчастіше події представляють виконуваний користувачем дії, як-от: натискання клавіш на клавіатурі або миші й переміщення миші.

Події передаються від ХрСервера до Х-клієнта, при одержанні яких останній уживає відповідні дії, зокрема посилає нові запити. У відповідь на некоректні запити ХоСервером генеруються пакети помилок, що направляються в Xlib і далі ХтКлієнтові.

X11 - повністю відкрита система, доступна у вихідних кодах. Крім того, X11 є розширюваною системою. Саме дана обставина робить X Window System дуже гнучкою в мережнім середовищі. Розширення X11 являють собою модулі, що підключаються до Xlib функціональні можливості, що й додають нові, X11 або враховуючі специфічні особливості ХрСервера. Наприклад, у стандартній поставці X Window System відсутня підтримка тривимірної графіки або підтримка шрифтів Postscript, однак численні розширення дозволяють це зробити.

З погляду виводу графічної інформації X11 являє собою потужну систему. ХеСервер використовує для графічної обробки свої обчислювальні можливості, у тому числі графічну систему, і ніяк щодо цього не пов'язаний з машиною, де виконується ХнКлієнт. Так, якщо ХеСервер одержав запит на відрисовку еліпса, те це проблеми винятково ХрСервера. Простіше говорячи, машина, що виконує ХнКлієнт, займається обчисленнями й обробкою подій, а ХеСервер відповідає за вивід графіки. Мабуть, це один із самих класичних прикладів архітектури клієнт-сервер.

У силу своєї специфіки X11 пред'являє досить високі вимоги до пропускної здатності мережі. Її стихія - локальні мережі зі швидкістю не менш 10 Мбіт/с. На менших швидкостях ХнКлієнт і Х-сервер починають погано ладити один з одним. Особливо це помітно при роботі зі складною графікою. Справедливості заради варто сказати, що останній реліз X Window System Version 11 Release 6 значно послабив вимоги до пропускної здатності мережі за рахунок оптимізації й буферизації інформації, переданої між Х Клієнтом і Х-сервером.

Крім того, оскільки X11 являє собою надбудову над ОС, вимоги до обсягу оперативної пам'яті досить високі. Типовий ХеСервер на ПК використовує не менш 4 Мбайт пам'яті. Правда, зараз дані вимоги не виглядають надмірними, але в 800х роках це було, безумовно, так.

X11 добре підходить для додатків, зайнятих інтенсивною обробкою графічної інформації, наприклад, у завданнях САПР. ХлТермінали - це звичайно бездисківні станції з більшими дисплеями (17-19 дюймів) і із запрограмованим ПЗУ (флешпам'яттю). Обсяг оперативної пам'яті становить від 4 до 64 Мбайт. Ціни на ХлТермінали залежать від розміру дисплея, типу графічного процесора й обсягу пам'яті. Діапазон цін ХлТерміналу коливається від 1500 до 5000 доларів.

Х Термінал вважається товстим клієнтом, мабуть, самим товстим серед тонких. ХлТермінали й X Window System знайшли застосування, головним чином, у системах старшого й середнього класу, особливо в середовищі UNIX і VMS.

На жаль, ні Microsoft, ні IBM, законодавці мод на ринку персональних комп'ютерів в 80е роки, не виявили необхідної уваги до технології X11. І проблема не тільки в підвищених вимогах X11 до комп'ютерних ресурсів. Кожна із цих компаній прагла очолювати на ринку ПК і не зважала на інших гравців. Характерно, що й Windows, і OS/2 дуже схожі на X Window System, але функціонально набагато бідніше. Вони не надають навіть мінімальних можливостей багатокористувацького мережного середовища. Проте безліч продуктів незалежних компаній дозволяють реалізувати X Window System у середовищі Windows і OS/2. Однак усі такі додатки мають той недолік, що віконна система X11 розміщується поверх іншої віконної системи (Windows або OS/2). Через це страждає продуктивність і накладаються численні обмеження на роботу X11.

Слабкістю X11 є її недостатня захищеність при роботі в мережі, у всякому разі, що погано відповідає сучасним вимогам до безпеки.

По даним деяких досліджень, попит на ХлТермінали останнім часом залишається незмінним, хоча ринок Unix-Машин збільшується. Причина тут та ж, що й у випадку інших тонких клієнтів. Мало хто з користувачів прагне мати в себе на столі "неповноцінний" комп'ютер.

БЕЗДИСКОВІ СТАНЦІЇ

Звичайно коли говорять про бездискові станції, то мають на увазі не їх функціональні особливості, а властиве їм апаратне обмеження - відсутність вінчестера. До бездискових клієнтів можуть бути віднесено багато типів тонких клієнтів: ХлТермінали, Windows-термінали, мережні комп'ютери, бездискові станції UNIX і т.п. Найчастіше операційна система в бездискову станцію завантажується віддалено, хоча іноді вона може втримуватися у флешпам'яті або прошита в ПЗУ.

Яскравим прикладом бездискової станції служить бездисковий ПК у середовищі Netware. Такі ПК дуже популярні в банківській сфері, оскільки забезпечують високий ступінь безпеки й простіше в адмініструванні, чому звичайні комп'ютери. Такі ПК працюють у такий спосіб. У мережній платі бездискового ПК встановлене ПЗУ вилученого завантаження. При включенні комп'ютера із ПЗУ мережної плати ініціалізується код, за допомогою якого посилає запит по мережі на завантаження ОС. Сервер Netware у відповідь на це пересилає завантажувальний образ операційної системи, після чого підключаються мережні диски Netware.

Такі бездискові станції можуть працювати з MS-DOS/Windows 3.x або Windows 95. Усі додатки, виконувані на ПК, і дані, які вони використовують, спочатку пересилаються по мережі в оперативну пам'ять цього ПК. З погляду логіки обробки графічної інформації немає принципової різниці між звичайним ПК і бездисковою станцією Netware, за винятком того, що додатка й дані зберігаються не на локальному диску, а на мережному.

Для бездискових станцій Netware найкраще підходить Ethernet, але годиться й Arcnet (2 Мбіт/с). У цей час бездискові станції Netware не дуже популярні, оскільки обсяг інформації, оброблюваної настільними комп'ютерами, багаторазово зріс. Бездискові станції Netware створюють занадто більше навантаження на мережу й сервери.

МЕРЕЖНІ КОМП'ЮТЕРИ

Мережні комп'ютери зобов'язані своєю популярністю розвитку Web-Технологій. Хоча зараз багато називають мережними комп'ютерами зовсім різні пристрої, зважуючи в одну купу й WindowseТермінали, і машини Netpc, і навіть ХлТермінали, у класичнім визначенні СК - це бездисковий комп'ютер, призначений для виконання Java-Додатків у середовищі Internet/Intranet, і вартістю порядку 500 доларів. Сам термін "мережний комп'ютер" з'явився з легкої руки глави Oracle Ларрі Еллісона, а концепція СК зустріла підтримку основних супротивників Microsoft. Але до розчарування багатьох створити СК вартістю в 500 доларів так і не вдалося. Звичайно маєтись на увазі, що на СК встановлена Java-Сумісна ОС, яка або завантажується із сервера, або перебуває у флеш-пам'яті. До типових СК можна віднести Javastation компанії Sun, що працює з операційною системою Java OS. І операційна система й Web Браузер Hotjava тут завантажуються із сервера. Після цього через Web Браузер можна здійснювати доступ до Java Додатків у мережі (див. Малюнок 2).

(1x1)

Малюнок 2. Робота мережного комп'ютера.

СК працює аналогічно бездискової станції Netware, але між ними є принципова різниця. По-перше, JavaЖДодатки є незалежними від платформи, тому їх можна запускати на будь-якому сервері Internet/Intra-net. По-друге, доступ додатків до даних може здійснюватися не тільки локально (шляхом їхнього пересилання на СК у форматі HTML), але й удаленно, через відповідне програмне забезпечення серверів Web. Це дозволяє значно знизити мережний трафік.

СК добре працює в локальній мережі, однак у глобальних мережах низька пропускна здатність приводить до більших затримок при завантаженні JavaЖДодатків. Стихія СК - мережі із пропускною здатністю більше 2 Мбит/с.

Основна функціональна гідність СК у вбудованій підтримці технології Java і, таким чином, у незалежності від операційної системи сервера. Як наслідок, СК прекрасно підходить для гетерогенного мережного середовища. Ще одна перевага СК - відносно невисока ціна. Звичайно, ні про які 500 долари зараз уже мова не ведеться, але однаково вартість СК - одна з найнижчих серед тонких клієнтів.

Недоліки СК є продовженням їх гідностей. Залежність від Java можна трактувати і як обмеження. Насамперед, JavaЖДодатків зараз на ринку зовсім небагато, особливо якщо мова йде про серйозні офісні пакети. До того ж швидкість виконання JavaЖДодатків у край низка, і, як не дивно, особливо в СК. По тестах швидкість виконання JavaЖДодатків на Javastation на порядок нижче, чим на ПК із процесором Pentium, хоча й на процесорах Pentium вона далеко не блискуча. Це викликано принциповими обмеженнями технології Java (Java - інтерпретируемый мова), які не вдається подолати навіть хитруваннями типу компіляція на лету (Just-In-Time). Проте незабаром на ринку повинні з'явитися спеціалізовані процесори Java, які дозволять кардинально підвищити продуктивність СК.

Сама по собі мова Java погано підходить для обробки графічної інформації, і СК не можуть змагатися з ХмТерміналами по можливостях роботи із графікою. Однак у більшості випадків мережні комп'ютери непогано справляються зі звичайними офісними додатками типу текстових редакторів, а також із завданнями вилученого керування серверами.

Після оголошення специфікації СК багато виробників програмно-апаратного забезпечення підтримали цю ідею. Багато, але далеко не все. Microsoft сприйняла ідею СК у багнети, оскільки бачила в ньому погрозу для свого домінуючого положення на ринку настільних систем. І, треба сказати, заперечення Білла Гейтса не позбавлені логіки. Хто захоче мати в себе на столі цього, по комп'ютерних мірках, виродка, із у край низькою продуктивністю вінчестера, що не має, а за ціною, що відрізняється від ПК тільки у два рази? Інша справа, що сам Гейтс пропонує не менш половинчате рішення.

Хлопці в Oracle і Sun виявилися з більшим почуттям гумору. Вони прогнозували лавинообразний попит на мережні комп'ютери, які зрештою повинні серйозно потіснити, а то й витиснути персональні комп'ютери не тільки в організаціях, але й у сфері домашнього

використання. За останні роки ХоТерміналів продали кілька сот тисяч штук, але ніхто й ніколи не говорив про їхній помітний вплив на комп'ютерний бізнес.

Більшість експертів упирає на те, що використання СК значне знизить витрати на адміністрування. Правда, кожний з них приводить на підтвердження свої власні оцінки, що часом відрізняються друг від друга на порядок. Але все-таки проблеми адміністрування зв'язані скоріше не із ПК, а з тим, що сучасне програмно-апаратне забезпечення погане пристосоване для адміністрування. Людям заради цього доводиться шукати обхідні шляхи, які може й спростують адміністрування, але в той же час ускладнюють життя кінцевим користувачам. Впливаючи такій логіці, простіше замінити ПК друкарськими машинками й арифмометрами. Як відомо, гільотина - кращий засіб від лупи.

Зрозуміло, що на корпоративному рівні можна змусити користувачів працювати на СК у приказному порядку, але хто захоче поставити їх у себе будинку? До того ж у домашніх умовах мережний комп'ютер буде нормально працювати тільки з високошвидкісним модемом, наприклад з кабельним. А сам кабельний модем у цей час коштує стільки ж, скільки СК. Та й чи навряд користувачам сподобається бути повністю залежними від свого провайдера і його сервера, тим більше що найчастіше все, що їм потрібно, - це попрацювати з текстовим редактором.

Поки пройшов порівняно невеликий строк від моменту появи перших СК. Але й зараз уже можна зробити деякі висновки. Попит на СК виявився зовсім не таким більшим. Так, наприклад, у Росію було поставлено всього кілька сотень СК серії Javastation компанії Sun, причому більша частина була замовлена партнерами Sun, а не кінцевими користувачами.

ТЕХНОЛОГІЯ Netpc

На протипагу ініціативі СК компанії Microsoft і Intel висунули свою власну концепцію під кодовою назвою Netpc, яка спільно зі стратегією Microsoft Zero Admin Windows покликано знизити витрати на програмно-апаратне забезпечення й зменшити витрати на адміністрування клієнтів у середовищі Windows NT. Згідно зі специфікацією тонкий клієнт Netpc повинен мати процесор не менш Pentium 133 МГц (або RISC, сумісний з NT), оперативну пам'ять не менш 16 Мбайт, підтримувати plug'n'play і шину USB.

Завантажуватися й управлятися Netpc буде із сервера NT, а в якості операційної системи буде застосовуватися Windows 95/98 або Windows NT Workstation. Цікаво, що комп'ютер Netpc планува_роби, що Адміністрування Netpc, а також WindowspТерміналів (см. далі) буде здійснюватися через інтерфейс WMI (Windows Management Interface), який стане складовою частиною Windows NT 5.0.

Передбачається, що для роботи в мережі будуть використовуватися додатка Activex і Java, причому, по цілком зрозумілих причинах, Microsoft віддає пріоритет технології Activex.

Очевидно, що Netpc набагато "товстіше" мережного комп'ютера й має могутніші можливості. Крім того, додатка Activex виконуються швидше Java. Activex менш безпечні, чому Java, але в корпоративнім середовищі Intranet це не має такого значення, як в Internet. Єдиним, мабуть, серйозним недоліком є те, що Activex, як і сама платформа Netpc, орієнтована на Windows, що обмежує їхнє використання в гетерогеннім середовищі. Вимоги до пропускну́ї здатності мережі в Netpc такі ж, як у СК, але ціна трохи вище (порядку 1000 доларів). Комп'ютери Netpc подібно СК погано справляються з інтенсивною обробкою графічної інформації: Activex і Java не призначені для цього.

Майбутнє покаже, наскільки очікування Microsoft і Intel у відношенні Netpc виправдані. Але, швидше за все, на ринку домашніх комп'ютерів у нього буде проблем не менше, чим у СК.

Більш імовірний інтерес до Netpc з боку корпоративних замовників, оскільки таке рішення цілком підходить для роботи із простими офісними додатками в локальних мережах.

WINDOWS ТЕРМІНАЛИ Й ТЕХНОЛОГІЯ WINFRAME

Для Microsoft Windows існують кілька продуктів незалежних компаній, що реалізують багатокористувацький графічний інтерфейс. Серед них особливо виділяється технологія Winframe, запропонована Citrix. На це є дві причини: по-перше, Winframe надзвичайно гнучка й потужна технологія, і, по-друге, що, мабуть, найважливіше, технологію Winframe побрала на озброєння Microsoft. На основі Winframe компанія Microsoft реалізує проект Windows-based Terminal Server (кодова назва Hydra), згідно з яким у середовищі Windows NT 5.0 Server будуть підтримуватися WindowseТермінали, що працюють по протоколу T.Share.

Winframe - це багатокористувацьке мережне середовище, реалізована на основі протоколу ICA (незалежна комп'ютерна архітектура, Independent Computing Architecture). На Windows NT Server установлюється серверна частина Winframe, а клієнтами можуть бути найрізноманітніші машини, включаючи несумісні з Windows (див. Малюнок 3). Відповідно до архітектури Winframe додатки можуть виконуватися повністю на сервері Winframe, а на клієнт буде передаватися лише зображення (вірніше, зміни в зображенні), а також здійснюватися введення інформації із клавіатури й миші. Таким чином, на клієнтові можуть відображатися будь-які додатки, що працюють в Windows NT (16- і 32-розрядні програми Windows і 16-розрядні додатки DOS). Протокол ICA у якості транспортного може використовувати фактично будь-який стандартний протокол, зокрема TCP/IP, Netbeui, IPX/SPX і PPP.

(1x1)

Малюнок

3.

Winframe підтримує різні платформи.

Клієнт Winframe самий тонкий із усіх тонких клієнтів. За повідомленням Citrix, для роботи клієнта Winframe досить бездискової машини на базі 2866го процесора (!) з 640 Кбайт оперативної пам'яті (!) при пропускній здатності мережі 20 Кбіт/з (!!!). Правда, необхідно мати на увазі, що зменшення "товщини" клієнта приводить до збільшення "товщини" сервера, оскільки на ньому виконуються всі додатки клієнтів. Позитивною рисою Winframe є те, що він підтримує кластери Windows NT Server.

Технологія Winframe підтримує криптографічний захист, власний сервіс дистанційного доступу, стиск даних, локальну й вилучену печатку, відображення мережних дисків сервера Winframe на локальні диски й має багато інших цікавих особливостей. Надтонкі клієнти Winframe знайдуть своє продовження в WindowseТерміналах проекту Microsoft Windows-based Terminal Server (для Windows NT Server 5.0), хоча Microsoft розв'язала використовувати тут власний протокол T.Share. Надтонкі клієнти як інтерфейси в зовнішній мир можуть задіяти браузері Web, які, як і інші додатки, будуть виконуватися на сервері.

На основі технології Winframe можуть реалізовуватися й інші типи клієнтів на зразок СК, Netpc і навіть більш товсті (див. Малюнок 4), у яких є локальні диски й принтери. У такому варіанті клієнти Winframe для доступу до корпоративної інформації можуть використовувати додатки Java і Activex, виконувані локально, а для роботи з офісними програмами задіяти сервер Winframe у якості сервера додатків.

(1x1)

Малюнок

4.

Додатка, виконувані на сервері Winframe, можуть звертатися до ресурсів як сервера, так і клієнта.

Технологія Winframe знайшла чимало прихильників (крім Microsoft) у комп'ютерному світі. Наприклад, по ліцензії Citrix компанії Textronic, Insignia і NCD продають продукти (Windd, Ntrigue і Win-Center Pro відповідно) можливості, що надають, доступу до сервера NT з боку Unix-машин і X-терміналів.

Winframe, безумовно, дуже цікавий продукт, але в нього є ряд недоліків. Насамперед це викликане орієнтацією сервера Winframe винятково на Microsoft Windows NT Server 3.51. Крім того, ліцензія на один клієнта Winframe досить дорога - від 200 до 400 доларів (залежно від числа клієнтів).

Оскільки у випадку надтонкого клієнта всі додатки виконуються на сервері, то серйозні графічні пакети працюють погано: сервер випробовує більше перевантаження навіть при невеликій кількості користувачів. Виконання графічних програм на клієтові Winframe не витримує ніякого порівняння з роботою на X-терміналах. Разом з тим на відміну від програм для X Window System у середовищі Winframe можуть виконуватися самі звичайні програми Windows і DOS, тобто не потрібно спеціального програмування розраховуючи на серверне розподілене середовище. Клієнти Winframe відмінно підходять для адміністрування вилучених серверів NT, у тому числі й по повільних лініях, і задовільно - для роботи з офісними додатками.

У порівнянні з іншими технологіями Winframe (необов'язково в продуктах Citrix) більш перспективна хоча б через свою велику гнучкість і невисоких вимог до ресурсів.

Висновок

При організації вилученого керування серверами необхідно брати до уваги безліч факторів, у першу чергу, характеристики мережний ОС, продуктивність ліній зв'язку, питання безпечної аутентифікації. Найбільш повний набір засобів керування надає UNIX, однак, при грамотному підході, адміністраторам Windows NT і Netware також немає підстави для занепокоєння.

Тонкі клієнти знову вертаються в життя. Однак якщо відкинути зайву рекламну лушпайку, то далеко не завжди дешеве рішення - краще. Вивід напрашується сам собою: чому "тонше" клієнт, тем "товстіше" сервер і тим гірше замовник працює із графікою.

Загальний доступ до мережних ресурсів.

1. Служба общего доступа (sharing)
2. Протоколы общего доступа
 - SMB/CIFS
 - NFS
 - DFS
3. Низкоуровневые средства общего доступа. Протокол DAFS

Служба загального доступу (sharing)

Комп'ютерна мережа по визначенню являє собою розподілену систему. Її призначення — спільна робота користувачів. Така робота має на увазі *доступ до мережних ресурсів*, як те файли й каталоги, принтери і т.д. Для користувача звертання до мережних ресурсів повинне бути прозорим, тобто:

1. Вилучені ресурси повинні виглядати так, немов вони є локальними й звертання до них з додатків відбувається одноманітно.
2. Клієнтові повинне бути байдуже, яка платформа використовується як сервер загального доступу.

При цьому потрібно враховувати й те, що не всякий користувач повинен мати доступ до конкретного ресурсу, і не всякий ресурс повинен бути доступний усім. Т.е. необхідно забезпечити можливості керування загальним доступом як на рівні користувачів, так і на рівні окремих ресурсів.

Зазначені можливості надають спеціальні протоколи загального доступу. Найпоширенішими з них є SMB/CIFS для ОС Windows і NFS, використовуваний в Unix-Подібних системах.

Протоколи загального доступу

SMB/CIFS

SMB (Server Message Block) — це протокол, запропонований IBM для організації загального доступу до файлів, принтерів, послідовних портів, поштових гнізд (mail slots), іменованим каналам (named pipes) і API мережних комп'ютерів. Протокол SMB може бути використаний поверх мережних протоколів стека TCP/IP, а також поверх ряду інших мережних протоколів.

SMB — це типовий клієнтрсерверний протокол, який дозволяє клієнтському додатку виконувати операції доступу до загального ресурсу (читання, запис і т.п.) через запити до сервера. SMB вимагає встановлення й підтримки з'єднання, але може працювати й у датаграмному режимі.

В 1992 році з'явилася Samba — вільна реалізація протоколу SMB для Unix-Подібних операційних систем. Оскільки Microsoft не публікувала специфікації SMB і доповнень до нього, творця Samba Ендрю Триджеллу (Andrew Tridgell) довелося провести зворотню розробку протоколу на основі аналізу пакетів.

Просування протоколу SMB забезпечила корпорація Microsoft, включивши його підтримку у свої продукти. У мережній середовищі Microsoft Windows SMB був основним протоколом прикладного рівня для роботи з ресурсами ЛВС. Він призначений для виконання функцій загального доступу до файлів і принтерам, авторизації користувачів і обміну повідомленнями.

Протокол SMB представляє чотири види сервісів:

- *Керування сесіями.* Створення, підтримка й розрив логічного каналу між робочою станцією й мережними ресурсами файлового сервера.
- *Файловий доступ.* Робоча станція може звернутися до файлсерверу із запитом на виконання типових файлових операцій (відкриття файлу, читання даних і т.п.).
- *Сервіс печатки.* Робоча станція може ставити файли в чергу для печатки на сервері й одержувати інформацію про чергу печатки.
- *Сервіс повідомлень.* SMB підтримує просту передачу адресних і широкомовних повідомлень по локальній мережі.

Якщо режим керування сесіями в SMB прозорий для користувача, то іншими сервісами користувач може управляти безпосередньо за допомогою команди `net` (див. `net /?` у консолі ОС Windows).



Рис. 1. Netbios/SMB

Для керування сесіями протокол SMB споконвічно використовував Netbios у реалізації Netbeui (Netbios Extended User Interface) — розширений користувацький інтерфейс дейтаграмної передачі Netbios, розрахований на мережі порядку 20-200 робочих станцій. Мережі, засновані на протоколі Netbeui, легко реалізуються, але їх важко розширювати, тому що протокол Netbeui не маршрутизуємий.

Для використання SMB у мережах з більш складною топологією в Netbios була додана підтримка транспортних протоколів TCP (NBT, Netbios over TCP), IPX, Decnet і Xerox Networking (XNS) (рис. 1)

Сервіси SMB, що працюють у середовищі TCP/IP, використовують різні порти (стандартні назви портів підкреслюють тісний зв'язок SMB з Netbios):

netbios-ns	137/tcp	# NETBIOS Name Service
netbios-ns	137/udp	
netbios-dgm	138/tcp	# NETBIOS Datagram Service
netbios-dgm	138/udp	
netbios-ssn	139/tcp	# NETBIOS session service
netbios-ssn	139/udp	

В SMB перших версій була відсутня аутентифікація — тобто будь-який користувач міг використовувати будь-які ресурси, що, звичайно, обмежувало область застосування малими локальними мережами. Сучасні версії SMB підтримують два рівні доступу:

1. **Доступ на рівні ресурсів.** Обмеження накладаються серверною стороною на каталоги загального доступу. Кожний мережний каталог може бути захищений паролем і клієнт повинен указати цей пароль для одержання доступу до файлів із загального каталогу.
2. **Доступ на рівні користувачів.** Обмеження накладають на кожний файл у кожному загальному каталозі й вони засновані на користувацьких правах. Кожний користувач (клієнт) повинен увійти на сервер під своїм обліковим записом і пройти аутентифікацію. Після завершення перевірки дійсності клієнт одержує відповідний ідентифікатор користувача (user ID), який він повинен пред'являти для одержання доступу до ресурсів сервера.

Прив'язка до Netbios обмежувало використання SMB невеликими локальними мережами. Перша причина — відсутність в Netbios поняття мережі як такого й засобів перенаправку трафіка (маршрутизації). Друга — у прийнятій схемі адресації: ім'я ресурсу, по суті, рядок з 15 символів, плюс байт типу ресурсу: сервер, контролер домена і т.д. Природно, така однорангова система іменування не годиться для інтернету.

Для усунення цих обмежень в останніх версіях SMB використовувався протокол NBT (Netbios over TCP), що працював поверх стека TCP/IP.

CIFS створювався спільно розроблювачами Samba Team, незалежним співтовариством і Microsoft. Після того як протокол CIFS був представлений як відкритий стандарт, Microsoft припинила фінансування проекту й співробітництво з Samba Team, а підтримка CIFS у переробці Microsoft для сумісності з колишніми версіями SMB була включено в ОС Windows 2000.

CIFS (Common Internet File System) — це відкритий стандартний протокол (див. [CIFS](#)) на основі SMB, який забезпечує доступ до файлів і сервісам на вилучених комп'ютерах у мережах TCP/IP. На відміну від SMB, основним транспортом для CIFS є протокол TCP. Для серверів CIFS зареєстровані порти 445/TCP і 445/UDP (microsoft-ds # Microsoft Naked CIFS, див. /etc/services)

CIFS забезпечує функціональність схожу на FTP (File Transfer Protocol), але надає клієнтам поліпшений (схожий на прямий) контроль над файлами. Основні можливості CIFS наведено в [таблиці 1](#).

Табл. 1. Можливості протоколу CIFS

Можливість	Опис
Доступ до системи	Підтримка файлових операцій, таких як відкриття, читання, запис, пошук і закриття файлу або каталогу
Блокування файлів і записів	Неблокувальні додатки не мають доступу до заблокованого файлу або запису.
Безпечне кешування (випереджальне читання файлу)	Підтримується одночасний доступ на читання/запис безліччю клієнтів

(read-ahead) і відкладений запис (write-behind))

Повідомлення про зміну файлу	Додатки можуть реєструватися на сервері для одержання повідомлень про зміни у файлах або директоріях
"Переговори" про версію протоколу	Коли клієнт і сервер устанавлюють перший мережний контакт, вони обмінюються інформацією про версію протоколу (діалекті), який буде використовуватися. Різні діалекти можуть включати нові типи повідомлень, а також відмінності у форматах від інших діалектів.
Розширені атрибути	Підтримуються атрибути, що не ставляться до атрибутів файлової системи. Такі, наприклад, атрибути як ім'я автора, можуть бути додані до вбудованих системних атрибутів файлу (дата створення, дата модифікації та ін.)
Розподілені віртуальні томи	Протокол підтримує багатотомну віртуальну файлову систему, у якій усі "піддерева" файлової ієрархії для клієнта виглядають як одне ціле. CIFS прозора для користувача обробляє доступ до фізично переміщених або реплікованих елементів такої файлової системи.
Незалежність від серверів розпізнавання імен	Клієнти можуть використовувати будь-який механізм розпізнавання імен. Приміром, сервери DNS можуть використовуватися для одержання доступу до файлових ресурсів сервера через Internet.
Пакетні запити	Множинні файлові запити можуть бути "упаковані" в одне повідомлення, що скорочує час очікування відповіді сервера. Пакетна обробка можлива навіть тоді, коли наступні запити залежать від результатів виконання попередніх.
Підтримка Unicode	Рядка у форматі Unicode можуть використовуватися в іменах файлів, ресурсів і облікових записах користувачів.

Мережна файлова система NFS

NFS (Network File System, мережна файлова система) — це стандарт, який включає опис розподіленої файлової системи й мережного протоколу для роботи з нею. Перша специфікація NFS була розроблена компанією Sun в 1989 році й призначала для використання тільки в UNIX. Пізніше реалізації клієнтської й серверної частин стали розповсюдженими й в інших системах. Поточна версія (на момент написання цього матеріалу) має назва NFS v4 і є відкритим стандартом ([RFC 3010](#)).

Ця система дозволяє користувачеві працювати з вилученими даними так само, як з локальними — тобто абсолютно прозоро. Не вважаючи, природно, тимчасових затримок.

Протокол NFS, як і SMB/CIFS, використовує клієнтсерверну модель взаємодії. У ранніх версіях NFS для транспортування даних використовувався UdpоПротокол, у сучасних — використовується TCP. Сервіс NFS працює на наступних зареєстрованих портах:

```
nfs      2049/tcp      # Network File System - Sun Microsystems
nfs      2049/udp      # Network File System - Sun Microsystems
```

Застосування протоколу TCP у якості транспорту дозволило розв'язати питання спільного доступу прямолінійно, без необхідних маневрів, однак ціною цьому є деяке зниження продуктивності в порівнянні з UDP.

NFS є «рідною» файловою системою для UNIX і відповідає логіці файлових операцій цієї операційної системи. Це ставиться як до простору імен файлів, так і до підтримуваних файлових атрибутів. Підтримка NFS є у всіх популярних систем на основі UNIX.

Уровни OSI	Протоколи NFS
Прикладной	NFS
Презентационный	XDR
Сеансовый	RPC
Транспортный	TCP/UDP
Сетевой	IP
Канальный	Ethernet, TokenRing и др.
Физический	

Рис. 2. Організація NFS відповідно до мережної моделі OSI.

Структура NFS включає три компоненти різного рівня:

- Прикладний рівень (властиво NFS) — це виклики вилучених процедур (rpc), які й виконують необхідні операції з файлами й каталогами на стороні сервера.
- Функції презентаційного рівня виконує протокол **XDR** (eXternal Data Representation), який є міжплатформним стандартом абстракції даних. Протокол XDR описує уніфіковану, *канонічну*, форму вистави дан, що не залежить від архітектури обчислювальної системи. При передачі пакетів РрсКлієнт переводить локальні дані в канонічну форму, а сервер проробляє зворотну операцію.
- Сервіс **RPC** (Remote Procedure Call), що забезпечує запит вилучених процедур клієнтом і їх виконання на сервері, представляє функції сеансового рівня.

Процедура підключення мережного ресурсу засобами NFS називається «експортуванням». Клієнт може запросити в сервера список експортованих ресурсів, що представляються їм. Сам сервер NFS, на відміну від, наприклад, сервера SMB, не займається широкомовним розсиланням списку своїх експортованих ресурсів.

Залежно від заданих опцій, експортований ресурс може бути змонтований «тільки для читання», можна визначити список хостів, яким дозволено монтування, указати

використання захищеного RPC (securerpc) та ін. Одна з опцій визначає спосіб монтування: «тверде» (hard) або «м'яке» (soft).

- При «**твердім**» монтуванні клієнт буде намагатися змонтувати файлову систему будь-що-будь. Якщо сервер не працює, це приведе до того, що весь сервіс NFS як би зависне: процеси, що звертаються до файлової системи, перейдуть у стан очікування закінчення виконання запитів RPC. З погляду користувацьких процесів файлова система буде виглядати як дуже повільний локальний диск. При поверненні сервера в робочий стан сервіс NFS продовжить функціонування.
- При «**м'якім**» монтуванні клієнт NFS зробить кілька спроб підключитися до сервера. Якщо сервер не відгукується, то система видає повідомлення про помилку й припиняє спроби зробити монтування. З погляду логіки файлових операцій при відмові сервера «м'яке» монтування емулює збій локального диска.

На запитання, який з режимів, «м'який» або «твердий», краще, однозначно відповісти неможливо. Якщо дані на клієнтові й сервері повинні бути синхронізовані при тимчасовій відмові сервісу, то «тверде» монтування виявляється переважніше. Цей режим незамінний також у випадках, коли монтуємі файлові системи містять у своєму складі програми й файли, життєво важливі для роботи клієнта, зокрема для бездисккових машин. В інших випадках, особливо коли мова йде про системи «тільки для читання», режим «м'якого» монтування представляється більш зручним.

З погляду безпеки перші реалізації NFS були вкрай слабкі: аутентифікація виконувалася, по суті, тільки по іреадресі клієнта. Використання NIS не набагато збільшувало захищеність системи. У версіях NFS 3 і 4 ці питання були перероблені шляхом додавання підтримки списків доступу (ACL), захищеного грс і ряду інших рішень, які дозволили сертифікувати NFS у міноборони США.

Загальний доступ у змішаній мережі

Сервіс NFS ідеально підходить для мереж на основі UNIX, тому що поставляється практично з усіма версіями цієї операційної системи. Більше того, підтримка NFS реалізована на рівні ядра UNIX. На жаль, використання NFS на клієнтських комп'ютерах з Windows створює певні проблеми, пов'язані з необхідністю установки спеціалізованого й досить дорогого клієнтського ПО. У таких мережах використання засобів поділу ресурсів на основі протоколу SMB/CIFS, зокрема ПО Samba, виглядає більш кращим.

Низькорівневі засоби загального доступу. Протокол DAFS

Архітектура віртуального інтерфейсу

Архітектура віртуального інтерфейсу (VIA) - це спільна розробка Microsoft, Intel і Compaq, яка визначає абстрактну модель низькорівневої технології введення/виводу. VIA використовується для організації високошвидкісного обміну даними між двома процесами, які працюють на різних серверах або системах зберігання в центрах обробки даних (ЦОД).

Virtual Interface (VI) — це протокол зв'язку в архітектурі Virtual Interface Architecture

DAFS (Direct Access File System) — це стандартний протокол файлового доступу, який базується на NFS v4. Він дозволяє прикладним завданням передавати дані в обхід

операційної системи і її буферного простору прямо до транспортних ресурсів, зберігаючи семантику, властиву файловим системам. DAFS використовує переваги новітніх технологій передачі даних за схемою пам'ять-пам'ять. Його використання забезпечує високі швидкості файлового введення/виводу, мінімальне завантаження CPU і всієї системи, завдяки значному зменшенню кількості операцій і переривань, які звичайно необхідні при обробці мережних протоколів. Особливо ефективним є використання апаратних засобів підтримки VI (Virtual Interface).

Коротко описати алгоритм роботи DAFS можна в такий спосіб: нехай є два додатки, що здійснюють доступ до мережі за допомогою мережного інтерфейсу користувацького рівня (VI, рис. 3), тоді:

1. Драйвер пристроїв операційної системи управляє апаратними засобами інтерфейсу традиційним способом, здійснюючи контроль над доступом додатків до встаткування.
2. Додатка виділяють буфери повідомлень у власному адресному просторі й звертаються до драйвера пристроїв для одержання доступу до мережного інтерфейсу. Після відповідного налаштування вони автоматично ініціюють процес передачі й приймання, а інтерфейс пересилає інформацію в буфери додатків і у зворотному напрямку, використовуючи звичайний механізм прямого доступу до пам'яті (direct memory access, DMA)

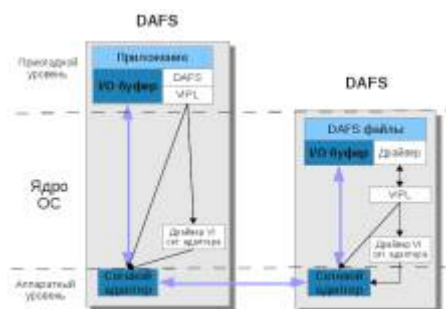


Рис. 3. Загальний принцип роботи DAFS

Додатка мають доступ до мережі з використанням віртуального інтерфейсу. Драйвер NIC ОС управляє мережним контролером і забезпечує прямий доступ додатків до інтерфейсу. Додатка виділяють буфери у власному адресному просторі, куди ухвалюють і звідки відправляють повідомлення, використовуючи DMA.

Архітектура мережного інтерфейсу користувацького рівня варіюється залежно від особливостей конкретних додатків і мереж — від способу визначення додатками місця розташування повідомлень, що пересилаються, від місцезнаходження виділюваних для приймання інформації буферів, від порядку повідомлення додатків про повідомлення, що зробили. У деяких мережних інтерфейсах (наприклад, в інтерфейсах Active Message або Fast Message) операції передачі й приймання реалізовані у вигляді функцій, поміщених у користувацьку бібліотеку, яка завантажується при ініціалізації кожного процесу. В інших (наприклад, в U-NET і VIA) для кожного процесу створюються черги, якими маніпулюють самі додатки. Ці черги обслуговуються апаратними засобами інтерфейсу.

DAFS спроектований для мереж зберігання даних (NAS) і використовується в кластерному і серверному оточенні баз даних і різноманітних Інтернет-Додатків, орієнтованих на безперервну роботу. Він забезпечує найменші затримки доступу до загальних файлових

ресурсів і даним, а також підтримує інтелектуальні механізми відновлення працездатності системи й даних.

Контрольні питання

1. Який протокол використовується для організації загального доступу до ресурсів у мережах Windows?
2. Які ресурси можна виділити в загальний доступ за допомогою протоколу SMB?
3. Навіщо в NFS виділений спеціальний протокол презентаційного рівня?
4. Чим відрізняється «тверде» монтування томів NFS від «м'якого»?

Режим дистанційного доступу до мережі. Сервіс RRAS

1. Удаленный доступ к сети
2. Дистанционное управление
3. Протоколы удаленного доступа
 - SLIP (?)
 - PPP
 - PPPE
4. Програмное обеспечение RRAS

Дистанційний доступ - мережна технологія, що забезпечує взаємодію абонентських систем (клієнтів) з локальними мережами через інфраструктуру телекомунікаційних мережі різного рівня. Служба дистанційного доступу дозволяє абонентам мережі, наприклад співробітникам, що перебувають у відрядженні, підключатися до ЛВС головного офісу, використовуючи звичайні телефонні лінії. А при використанні технології VPN, дистанційний доступ можна реалізувати поверх мережі Інтернет. Ці можливості забезпечуються за допомогою **сервера дистанційного доступу** - комп'ютера, що має мережне підключення й здатного ухвалювати вхідні з'єднання по протоколу PPP.

Служба дистанційного доступу представляє два режими роботи: *дистанційний доступ до мережі* й *дистанційне керування*. Для реалізації обох режимів сервер повинен підтримувати функції маршрутизації, щоб перенаправляти запити із зовнішнього інтерфейсу, до якого підключений абонент, на внутрішній інтерфейс із локальною мережею й назад. Саме тому в ОС Windows служби дистанційного доступу й маршрутизації об'єднані в загальний сервіс - Routing and Remote Access Service (RRAS).

Дистанційний доступ до мережі

Режим " дистанційний доступ до мережі" дозволяє вилученим клієнтам підключатися до серверів і робочим станціям локальної мережі, так, ніби вони були підключені безпосередньо. Т.е. у цьому режимі вилучена абонентська система розглядається як рівноправна робоча станція ЛВС.

Дистанційне керування

Режим "дистанційного керування" дозволяє вилученим клієнтам управляти роботою сервера або робочої станції, включеної в локальну мережу. Робота в режимі "дистанційного керування" відповідає клієнтр серверної моделі «сервер терміналів»

***Вилучений комп'ютер** (вилучений хост) - сервер, фізично розміщений на деякій відстані від клієнта й доступ до якого (для клієнтів) можливий тільки по мережі.*

Основні засоби доступу до серверів баз даних

Підключення до сервера БД

КлієнтрСерверні СУБД, як правило, представляють клієнтам кілька різних способів підключення, серед яких іменовані канали, сокети, мережні порти. Останні є основним способом взаємодії додатка й сервера баз даних. Номера зареєстрованих портів для розповсюджених СУБД можна подивитися у файлі /etc/services (cat /etc/services | grep sql, лістинг 1).

Лістинг 1. Зареєстровані порти клієнтрсерверних СУБД

sql*net	66/tcp	# Oracle SQL*NET
sql*net	66/udp	# Oracle SQL*NET
sqlserv	118/tcp	# SQL Services
sqlserv	118/udp	# SQL Services
sql-net	150/tcp	# SQL-NET
sql-net	150/udp	# SQL-NET
sqlsrv	156/tcp	# SQL Service
sqlsrv	156/udp	# SQL Service
mini-sql	1114/tcp	# Mini SQL
mini-sql	1114/udp	# Mini SQL
mysql-cluster	1186/tcp	# Mysql Cluster Manager
mysql-cluster	1186/udp	# Mysql Cluster Manager
ms-sql-s	1433/tcp	# Microsoft-sql-server
ms-sql-s	1433/udp	# Microsoft-sql-server
ms-sql-m	1434/tcp	# Microsoft-sql-monitor
ms-sql-m	1434/udp	# Microsoft-sql-monitor
sybase-sqlany	1498/tcp	# Sybase SQL Any
sybase-sqlany	1498/udp	# Sybase SQL Any
unisql	1978/tcp	# Unisql
unisql	1978/udp	# Unisql
unisql-java	1979/tcp	# Unisql Java
unisql-java	1979/udp	# Unisql Java
mysql-im	2273/tcp	# Mysql Instance Manager
mysql-im	2273/udp	# Mysql Instance Manager
mysql	3306/tcp	# Mysql

mysql	3306/udp	# Mysql
ssql	3352/tcp	# Scalable SQL
ssql	3352/udp	# Scalable SQL
postgresql	5432/tcp	# Postgresql Database
postgresql	5432/udp	# Postgresql Database
sqlexec	9088/tcp	# IBM Informix SQL Interface
sqlexec	9088/udp	# IBM Informix SQL Interface
sqlexec-ssl	9089/tcp	# IBM Informix SQL Interface - Encrypted
sqlexec-ssl	9089/udp	# IBM Informix SQL Interface - Encrypted

Клієнт СУБД

Усю роботу клієнтського додатка із сервером БД можна звести до виконання трьох завдань: підключення до сервера, відправлення запитів і одержання результатів, відключення. Ці та інші функції доступні клієнтському додатку через API сервера баз даних. Функції API, доступні клієнтові, описані в документації відповідної СУБД. У лістингу 2 наведений приклад використання API Mysql у програмі мовою C.

Лістинг 2. Приклад MysqlКлієнта мовою C

```
#include <stdio.h>
#include <stdlib.h>
#include "mysql.h" // підключення API

MYSQL mysql;
MYSQL_RES *res;
MYSQL_ROW row;

void exiterr(int exitcode)
{
    fprintf(stderr, "%s\n", mysql_error(&mysql));
    exit(exitcode);
}

int main()
{
    uint i = 0;

    if (!mysql_connect(&mysql, "host", "username", "password")) // з'єднання із БД
        exiterr(1);
    if (mysql_select_db(&mysql, "payroll")) exiterr(2);
    if (mysql_query(&mysql, "SELECT name, rate FROM emp_master")) // запит
        exiterr(3);
    // обробка відповіді
    if (!(res = mysql_store_result(&mysql))) exiterr(4);
    while((row = mysql_fetch_row(res))) {
        for (i=0 ; i < mysql_num_fields(res); i++)
            printf("%s\n", row[i]);
    }
    if (!mysql_eof(res)) exiterr(5);
    mysql_free_result(res);
    mysql_close(&mysql); // відключення
}
```

Постійні з'єднання із сервером БД

Операції відкриття й закриття з'єднання із БД привносять додаткові накладні витрати в процес взаємодії клієнтського додатка й сервера БД. Знизити їх можна шляхом установа-

постійного з'єднання (permanent connection) з базою даних. У цьому випадку з'єднання, відкрите спеціальною функцією API, утримується в цьому стані для майбутнього використання. При повторнім звертанні до цього ж сервера тим же користувачем (з логіном і паролем, зазначеними при першій підключенні) СУБД поверне дескриптор раніше відкритого з'єднання.

Приклад установлення постійного підключення до СУБД Mysql наведено в лістингу 3.

Лістинг 3. Постійне підключення до Mysql з PhpСкрипта

```
<?php
$link = mysql_pconnect("localhost", "username", "password")
        or die("Could not connect: " . mysql_error());
print ("Connected successfully");
?>
```

Обмеження

Використання API СУБД у клієнтських додатках простий, але не завжди підходящий спосіб. При переході з однієї СУБД на іншу це зажадає переписування коду клієнтів. Більш прозорим робота з базами даних робить застосування проміжного програмного забезпечення або використання уніфікованих інтерфейсів прикладного програмування (ODBC, JDBC и др.).

Уніфіковані засоби доступу до серверів баз даних

Внутрішні механізми доступу до БД

API СУБД

Наприклад, функції PHP для роботи dbase:

```
dbase_add_record -- Adds a record to a database
dbase_close -- Closes a database
dbase_create -- Creates a database
        int dbase_create ( string filename, array fields ) - створює нову базу
        даних

dbase_delete_record -- Deletes a record from a database
dbase_get_header_info -- Gets the header info of a database
dbase_get_record_with_names -- Gets a record from a database as an associative
        array
dbase_get_record -- Gets a record from a database as an indexed array
dbase_numfields -- Gets the number of fields of a database
dbase_numrecords -- Gets the number of records in a database
dbase_open -- Opens a database
dbase_pack -- Packs a database
dbase_replace_record -- Replaces a record in a database
```

Підтримка SQL

власні підмножини мови

Зовнішні засоби інтеграції додатків БД

Бібліотека Borland Database Engine (BDE)

Ключовий механізм BDE (Borland Database Engine), що забезпечує роботу візуальних компонентів баз даних, діє як інтерфейс між вашим додатком і самою базою даних. BDE реалізований у вигляді набору системних DLL файлів. Взаємодія компонентних об'єктів з BDE ніяк не специфікує конкретну базу даних і не залежить від реалізації обміну інформацією на нижньому рівні ієрархії. Саме BDE звертається у свою чергу до драйверів, специфічних для бази даних зазначеного типу, повертаючи вашому додатку запитані фактичні дані. BDE відіграє роль, аналогічну контролеру драйверів ODBC (Open Database Connectivity) виробництва фірми Microsoft, ізолюючи додатка від нижнього рівня взаємодії з базою даних і збільшуючи загальну продуктивність зв'язку за рахунок використання кеш-пам'яті. Використовуючи BDE, ви одержуєте доступ до всіх локальних стандартних баз даних вашого комп'ютера, до джерел даних ODBC і до SQL серверам баз даних в архітектурі мережного зв'язку клієнт/сервер.

Уніфікована технологія BDE застосовується у всіх продуктах виробництва корпорації Borland: C++Builder, Borland C++, Delphi, Intrabuilder і Jbuilder. Щоб одержати доступ до вмісту бази даних, додатку необхідно знати тільки ідентифікатор її псевдоніма (alias).

Якщо створене за допомогою 3++ Builder додаток у процесі роботи звертається до баз даних, воно, як правило, використовує для цієї мети бібліотеку BDE (Borland Database Engine), засновану на технології IDAPI (Integrated Database Application Program Interface). Ця бібліотека встановлюється автоматично при установці 3++ Builder. За замовчуванням вона встановлюється в каталог C:\Program Files\Borland\Common Files\BDE.

Слід зазначити, що файли, що входять до складу бібліотеки BDE, призначені для використання не тільки додатками, створеними за допомогою 3++ Builder, але й багатьма іншими продуктами Borland (Visual dbase, Paradox, Delphi, Borland C++, Intrabuilder), створеними на їхній основі додатками, а також офісними додатками Corel (наприклад, електронною таблицею Quattro Pro), генератором звітів Crystal Reports (Seagate Software) . Тому при наявності декількох, що використовують BDE додатків усі 32-розрядні додатки (у тому числі C++ Builder) використовують установлену останньої 32-розрядну версію BDE. У цьому випадку за замовчуванням програма установки C++ Builder запропонує поставити BDE у каталог, де встановлена вже використовувана версія BDE.



Рис.1. Зв'язок додатків із джерелами даних за допомогою BDE

BDE забезпечує для створених додатків:

- безпосередній доступ до локальних баз даних (dbase, Paradox, текстові файли)

- доступ до SqlСерверам (Oracle, Sybase, MS SQL Server, Interbase, Informix, DB2) за допомогою драйверів Borland SQL Links
- доступ до будь-яких джерел даних, що мають драйвер ODBC (Open Database Connectivity), наприклад, до файлів електронних таблиць (Excel, Lotus 1-2-3), серверам баз даних, що не мають драйверів SQL Links (наприклад, Gupta/Centura)
- створення додатків клієнт-сервер різномірні дані, що використовують
- високу продуктивність при роботі із плоскими таблицями
- використання SQL (Structured Query Language - мова запитів до серверних СУБД), у тому числі при роботі з локальними даними
- ізоляцію додатка від засобів мовної підтримки
- ізоляцію додатка від конфігурації системи й мережі

Microsoft Jet Database Engine

Ця стаття про JET Red використовується в Microsoft Access. Для здійснення JET Blue ISAM, див. Extensible Storage Engine.

Microsoft Jet Database Engine є Database Engine, на якому кілька продуктів Microsoft були побудовані. Бази даних є основним компонентом бази даних, збір інформації, що зберігається на комп'ютері на систематичній основі. Перша версія Jet була розроблена в 1992 році, що полягає із трьох модулів, які можуть бути використані для керування базою даних.

JET коштує спільних Engine Technology, іноді іменуються Microsoft Jet Engine або просто Jet. Microsoft Access і Visual Basic використовують або використовували Jet як лежачі в їхній основі бази даних. З тих пор він був замінений для загального використання, однак, спочатку Microsoft Desktop Engine (MSDE), а потім по SQL Server Express. Jet зараз є частиною Microsoft Windows і більше не компонента Microsoft Data Access Components (MDAC). Для більших потреб бази даних, бази даних Jet може бути підвищений (або, у термінології Microsoft, "до розміру"), щоб флагман Microsoft базу даних продуктів, SQL Server.

Протягом багатьох років, Jet стало майже синонімом Microsoft Access, у тому ступені, де багато з людей ставляться до бази даних Jet, як "доступ до бази даних". Навіть Microsoft співробітники роблять це іноді. Є партії, які бачать це як неправильне й заявляють, що Jet являє собою базу даних, і доступ до бази даних розробки додатків інструмент (база даних будівельник).

Архітектура Jet

Jet дозволяє маніпуляції реляційними базами даних і є частиною реляційної системи керування базами даних (СУБД). Він пропонує єдиний інтерфейс, що інше програмне забезпечення можна використовувати для доступу до баз даних Microsoft і забезпечує підтримку для забезпечення безпеки, цілісності, обробки транзакцій, індексування, записи й сторінки замок, і реплікації даних. У більш пізніх версіях, двигун був розширений, щоб мати можливість запускати SQL запити, зберігати символічні дані у форматі Unicode, створити бази даних думок і дозволяють двонаправленої реплікації з Microsoft SQL Server.

Бібліотеки Jet

Є три модулі для Jet: Одним з них є рідний Jet ISAM драйвер, динамічні бібліотеки (DLL), які можуть безпосередньо управляти Microsoft Access файли бази даних (MDB) з

використанням індексованих послідовного методу доступу (ISAM). Ще один з модулів містить драйвери ISAM, DLL, які дозволяють доступ до різних баз даних ISAM, серед них Xbase, Paradox, Foxpro і Btrieve, залежно від версії Jet. Остаточний модуль Data Access Objects (DAO) DLL. DAO надає API, що дозволяє програмістам доступ JET баз даних з використанням будь-якої мови програмування.

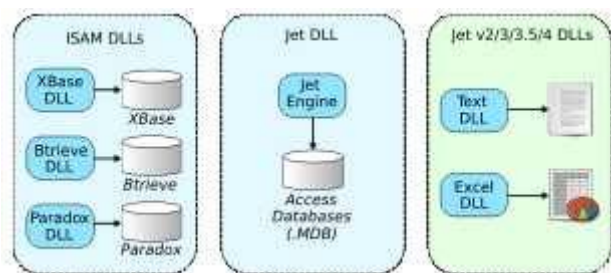


Рис. Компоненти MS Jet Database Engine

Open Database Connectivity (ODBC)

Тема нашої сьогоднішньої розмови - ODBC. Якщо ви хоч раз писали СУБД (системи керування базами даних), то вам, напевно, знайома ця абревіатура. Розшифровується ODBC як Open Database Connectivity (вільний переклад - відкрита система зв'язку з базами даних). Open (відкрита) у цьому випадку означає "розширювана", "легко нарощувана", "відкрита для модифікацій". Ви самі це зрозумієте, якщо подивитесь на архітектуру побудови ODBC:

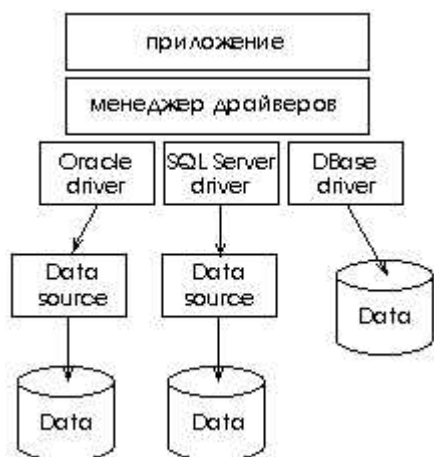


Рис. Архітектура ODBC

Як легко помітити, вона дійсно легко нарощувана. Для додавання нового типу БД потрібно лише написати драйвер і зареєструвати його. Ще одна перевага, що впливає з такої побудови ODBC - користувацький додаток спілкується з фізичної БД через менеджер драйверів, фактично нічого не знаючи про тип використовуваної БД (спілкування менеджера драйверів і самих драйверів більш менш стандартно, так що з погляду користувача всі джерела даних мають практично однакові властивості). Таким чином, ви легко можете поміняти фізичний тип бази даних, а додаток навіть не довідається про це (звичайно, існують виключення через особливості підтримки мови SQL різними типами БД, але вони несуттєві).

Фізично ODBC являє собою набір динамічних бібліотек DLL, які обслуговують підключення й роботу з конкретним типом бази даних. При запиті на підключення до певної, заздалегідь описаній бази "активізується" певна DLL - драйвер цього типу БД. Звертання до певної бази даних відбувається по імені так званого джерела даних ODBC (або DSN - data source name).

Можна представити DSN як свого роду оголошення БД на даному комп'ютері, причому DSN може бути користувацьким, тобто перебуває в області видимості тільки одного користувача, у сеансі якого був створений цей DSN, і системним, тобто видимим усіма користувачами даного комп'ютера.

Керування джерелами даних ODBC (та й взагалі настроюванням усієї системи ODBC) здійснюється за допомогою спеціальної програми - OdbcАдміністратора. Давайте розберемо, де його можна знайти в різних версіях Windows:

Windows 9x - сам файл, що виконується, odbcad32.exe лежить у каталозі Windows\System. Запускати його можна прямо або через Панель керування (значок "Джерела даних ODBC (32біт)"). Якщо на комп'ютері встановлений BDE, запустите адміністратор BDE і з меню Object виберіть пункт ODBC Administrator.

Windows 2000 - файл, що виконується, odbcad32.exe лежить у каталозі Winnt\System32, а запускати його можна через Панель керування -> Адміністрування -> Джерела даних ODBC. На жаль, у мене немає можливості перевірити, як з ODBC в NT3.51 / 4 - не думаю, що там хлопці з Microsoft сильно його сховали.

Чому ODBC?

Адже є ж BDE, скажете ви. Так, можливо ви праві. Однак тут є "але". В обох движків є свої переваги й недоліки. ODBC набагато краще працює з "рідними, від Microsoft" джерелами даних продуктів Microsoft Office - типу Access, Excel, MS SQL Server (взагаліте через Access я й затіяв усю цю писанину). Крім того, у стандартний дистрибутив BDE, який можна на кожному кроці знайти в Інтернеті, я щось жодного разу не зустрічав SQL Links, втім, як я жодного разу не зустрічав і окремого дистрибутива SQL Links. А саме після установки SQL Links в BDE з'являється можливість створювати аліаси до Access, MS SQL Server, Oracle і ін. типам баз даних. Установлювати ж повну версію C++Builder або Delphi на комп'ютер користувача може не кожний. А ODBC на "голій" машині вже містить драйвери Access, Excel, Foxpro і SQL Server. Загалом, для загального розвитку не перешкодить освоїти й ODBC, тим більше що access'овские бази, з якими ми будемо працювати через ODBC, дуже надійні (не в приклад стандартним Vde'шным paradox'овым, у яких раз у раз сиплються індекси, а про таке поле як лічильник я взагалі мовчу), практично на будь-якому комп'ютері користувача встановлений Office, що дозволяє вносити корективи "на місці", та й зручність усієї бази в одному файлі теж іноді позначається.

JDBC

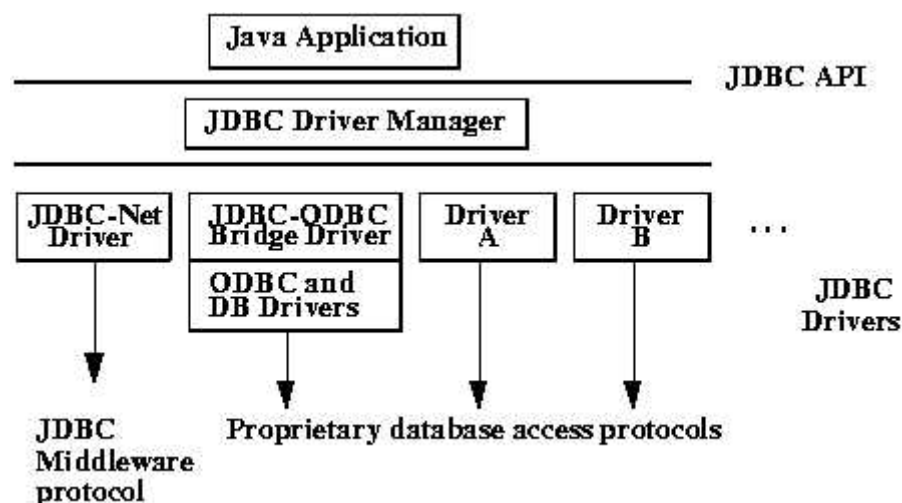


Рис. Архітектура JDBC

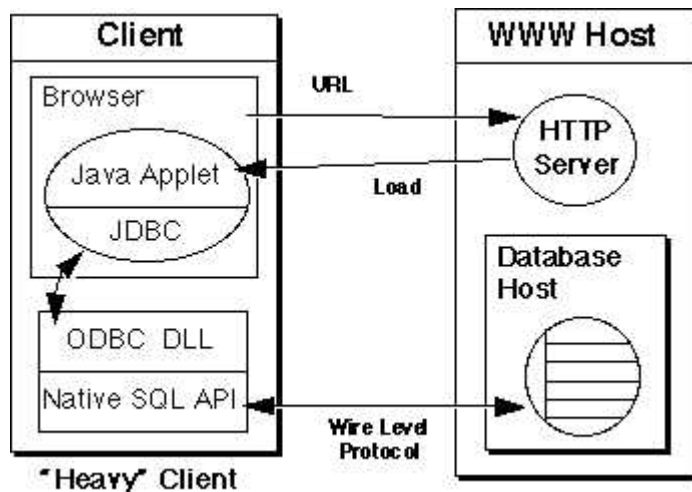


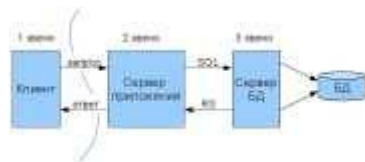
Рис. Спільне використання JDBC і ODBC

Контрольні питання

Сервери додатків

Винесення прикладної логіки в окремий рівень представляє розроблювачам додаткову гнучкість у створенні розподілених інформаційних систем. Розміщення й виконання програм на стороні сервера знижує вимоги до апаратного забезпечення клієнтів і зменшує проблеми забезпечення сумісності в гетерогенній мережній середовищі.

Сервер додатків — це сервісна програма, яка забезпечує доступ клієнтів до прикладних програм, що виконуються на сервері. Сервер додатків звичайно виділяється як середня ланка (мал. 1) у трехуровневой клиент-серверной архитектуре (3-tier):



Модель "сервер додатків"

1. Перший рівень, інтерфейсний, як правило, графічний (GUI).
2. Середній рівень, здійснений програмний код, розміщений звичайно на виділеному сервері.
3. Третій рівень, фоновий — бази даних. Сюди ж ставляться, успадковані засоби доступу до даних і керування транзакціями.

У мережній середовищі сервер додатків є посередником між фронт-ендами клієнтів і серверами баз даних.

Бізнес-Логіка може бути реалізована на стороні сервера як цілком (вилучений код), так і частково (розподілений код). У першому випадку до сервера можуть звертатися термінали й «тонкі» клієнти й така взаємодія відповідає моделі «сервер терміналів». «Толсті» і

гішклієнти можуть одержувати компоненти серверного додатка й виконувати їх на своїй стороні (наприклад javascript, апплеты, flash).

Мобільний софт

Завдання, розв'язувані серверами додатків добре ілюструються на прикладі мобільних сервісів. Можливості мобільних пристроїв споконвічно обмежені фізичними розмірами й часом автономної роботи (інші обмеження, в основному, впливають із цих двох). Мобільний додаток розробляється з урахуванням цих обмежень, але тому що софт для телефону повинен бути адаптований для використання на конкретній моделі, то процес розробки ускладнюється. Розділивши мобільний додаток на клієнтську (вистава даних) і серверну (прикладна логіка) частини, розроблювач одержує наступні можливості:

- урізана по функціоналу клієнтська частина виходить менш вимогливою до ресурсів;
- для підтримки нових пристроїв потрібно адаптувати тільки фронт-енд, не зачіпаючи прикладну логіку;
- зміни в програмі (розширення функціональності, виправлення помилок і т.п.) виконуються на сервері додатків і поширюються на всіх клієнтів.

Клієнти можуть взаємодіяти з додатками через API сервера (JavaКлієнт <—> контейнер сервлетів <—> сервлет). Більшу гнучкість і універсальність представляє взаємодію через сторонні сервіси, у першу чергу — через веб-сервер.

Поняття сервера додатків традиційно зв'язують із платформою Java, указуючи на те, що сервер JavaДодатків представляє реалізацію специфікації сервлетів, можливо у вигляді JSP, і ще деякі сервісні послуги, у першу чергу з'єднання з базою даних.

Але це щось більше й менше одночасно: *сервер додатків надає середовище, у якому прикладні програми можуть працювати, незалежно від того, що і як саме вони роблять.*

Тому, щоб відповісти на запитання, чи є (і в якому ступені) якесь сервісне ПО сервером додатків, варто зрівняти його заявлені функції зі списком атрибутів, властивих цієї категорії:

- Надає модель контейнера для додатків.
- Надає сервісні послуги для програм.
- Забезпечує керування додатками й/або представляє засоби їх розробки.
- Відповідає індустріальним специфікаціям і стандартам.
- Додамо сюди ж обслуговування веб-сторінок, через реальну затребуваність технологій на основі WWW.

Реалізації

По наведених ознаках у розглянуту категорію попадають, наприклад, традиційні терміналссерверні системи, технологія CGI, контейнери Java Сервлетів і др.

Успадковані рішення

Сервери терміналів представляють середовище для вилученого виконання програм, у якості якої виступає сама операційна система. Доступ до них здійснюється по протоколах вилученого керування (telnet, ssh, RDP, VNC і т.п.) із клієнтського ПО (емулятор терміналу,

засобу керування вилученим робочим столом і т.п.). Керування запущеною програмою виконується через емульований на клієнтові користувацький інтерфейс (текстовий або графічний) операційної системи. На серверній стороні взаємодія програм з ОС реалізується через системні виклики. Керування також здійснюється засобами операційної системи. Розробка може вестися на будь-якій мові, доступному в конкретній ОС.

Загальний шлюзовий інтерфейс (CGI) — технологія доступу до додатків через веб-сервер. Відмінності від сервера терміналів тут у тому, що користувацький інтерфейс надається у вигляді веб-сторінок. Запити веб-клієнтів, звернені до програм, розміщених у виділеному каталозі (як правило cgi або cgi-bin) перенаправляються на їхній вхід через стандартний потік уведення (stdin). Результати виконання у вигляді гіпертексту додаток повертає веб-серверу через stdout.

Сервери JavaДодатків

Платформа Java є індустріальним стандартом, що дозволяють створювати з уніфікованих компонентів інтероперабельні програмні рішення для самих різних систем, у яких може бути запущена віртуальна машина Java (JVM).

Контейнер сервлетів — один з архітектурних компонентів J2EE, що представляє оточення для виконання сервлетів. Сервлет — це JavaДодаток, що виконується на стороні сервера (на відміну від аплета). Контейнер сервлетів може працювати як повноцінний самостійний сервер, але частіше використовується (інтегрується) разом з іншим серверним ПО. Забезпечує обмін даними між сервлетом і клієнтами, бере на себе виконання таких функцій, як створення програмного середовища для функціонуючого сервлета, ідентифікацію й авторизацію клієнтів, організацію сесії для кожного з них.

Концепція сервлетконтейнера дозволяє створювати як універсальні, так і спеціалізовані сервери додатків (наприклад, для мобільних сервісів).

Прикладом реалізації контейнера сервлетів є Apache Tomcat, який використовується в таких серверах додатків як Apache Geronimo, Jboss, Glassfish, IBM Websphere Application Server (WAS).

Інші рішення

Компанія Microsoft представляє власні рішення для підтримки бізнес-логіки й сервісної інфраструктури на основі ОС Windows Server і технології .NET Framework. Основним засобом розробки є мова C#.

Мова python, що одержав популярність багато в чому завдяки Google, є основним засобом розробки для сервера веб-додатків Zope.

Для сценаріїв мовою PHP, широко використовуваному для створення веб-сайтів, компанія Zend Technologies (розроблювач самої мови PHP) створила сервер додатків Zend Server.

Сервери додатків: плюси й мінуси

Переваги

Цілісність коду й даних

Розміщення бізнес-логіки на виділеному сервері або обмеженім числі серверних комп'ютерів гарантує доступ до оновленого й модернізованого ПО для всіх клієнтів. Це виключає ризик доступу й керування даними із застарілих і, можливо, несумісних програм.

Централізоване керування

Зміни в конфігурації прикладних програм, такі як, наприклад, зміна сервера баз даних, виконуються централізовано.

Безпека

Централізовані засоби, через які постачальник послуг (сервіс-провайдер) може управляти доступом до даних і компонентів додатка, дозволяють виконувати перевірку дійсності потенційно ненадійних клієнтів у середньому шарі й не торкатися рівня бази даних.

Продуктивність

Сервер додатків може вирішувати завдання балансування мережного трафіка й розподілу навантаження між іншими фізичними серверами системи.

Загальна вартість володіння

Сукупність перерахованих вище переваг, а на додаток до них перерозподіл витрат на встаткування із клієнтської на серверну сторону, може привести до економії засобів для організації. Так само на зниженні загальної вартості володіння може відбитися практика оренди програмного забезпечення. Справедливості заради потрібно відзначити, що вартість самого серверного ПО, а також витрати на його впровадження й супровід можуть бути досить високими.

Недоліки

Централізація

Системи, побудовані на основі сервера додатків, мають один основний недолік, властивий усім централізованим рішенням — «падіння» сервера приведе до неприступності програм для всіх клієнтів. До того ж ефекту приведуть і неполадки в мережнім підключенні.

Захист інформації

Ця проблема, у принципі, актуальна для будь-яких мережних рішень, що використовують для передачі даних інфраструктуру публічних мереж.

Веб-сервіси як засіб інтеграції додатків в WWW

Що є веб-сервіс?

Всесвітня павутина є готовою платформою для створення й використання розподілених машинно-орієнтованих систем на основі веб-сервісів. Веб-сервер виступає як сервер додатків, до яких звертаються не кінцеві користувачі, а сторонні додатки. Це дозволяє

багаторазово використовувати функціональні елементи, усунути дублювання коду, спростити рішення завдань інтеграції додатків.

Веб-служба, **веб-сервіс** (англ. web-service) — це мережна технологія, що забезпечує міжпрограмна взаємодія на основі веб-стандартів. Консорціум W3C визначає веб-сервіс, як «програмну систему, розроблену для підтримки інтероперабельної міжкомп'ютерної (machine-to-machine) взаємодії через мережу»



Рис. 1. Концепція веб-сервісу



Рис. 2. Протоколи веб-сервісів

Веб-служби: концепції й протоколи

Веб-сервіс ідентифікується рядком URI. Веб-сервіс має програмний інтерфейс, представлений у машинно-оброблюваному форматі WSDL. Інші системи взаємодіють із цим веб-сервісом шляхом обміну повідомленнями протоколу SOAP. У якості транспорту для повідомлень використовується протокол HTTP. Опис веб-сервісів і їх API можуть бути знайдені засобами UDDI. Концептуальна схема технології наведено на рисунку 1.

- **SOAP** (Simple Object Access Protocol) — протокол обміну повідомленнями між споживачем і постачальником веб-сервісу;
- **WSDL** (Web Services Description Language) — мова опису зовнішніх інтерфейсів веб-служби;
- **UDDI** (Universal Discovery, Description and Integration) — універсальний інтерфейс розпізнавання, опису й інтеграції, використовуваний для формування каталогу веб-сервісів і доступу до нього.

Усі ці специфікації засновані на XML і, відповідно, успадковують його переваги (структурованість, гнучкість і т.д.) і недоліки (громіздкість, повільність).

SOAP

SOAP (споконвічно *Simple Object Access Protocol*, а у версії 1.2 офіційне розшифрування абревіатури відсутнє) — простий протокол доступу до об'єктів (компонентам розподіленої обчислювальної системи), заснований на обміні структурованими повідомленнями. Як будь-

який текстовий протокол, SOAP може використовуватися з будь-яким протоколом прикладного рівня: SMTP, FTP, HTTPS і ін., але найчастіше SOAP використовується поверх HTTP.

Усі повідомлення SOAP оформляються у вигляді структури, називаної *конвертом* (envelop), що включає наступні елементи:

- Ідентифікатор повідомлення (локальне ім'я).
- Опціональний елемент Header (заголовок):
 - Нуль або більш посилань на використовувані простори імен;
 - Нуль або більш властивостей, доступних у цьому просторі імен.
- Обов'язковий елемент Body (тіло повідомлення)
 - Нуль або більш посилань на використовувані простори імен;
 - Дочірні елементи тіла повідомлення

Розгорнутий список елементів повідомлення SOAP наведений у схемі даних (для SOAP версії 1.2).

Приклад повідомлення SOAP:

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
      <n:priority>1</n:priority>
      <n:expires>2001-06-22T14:00:00-05:00</n:expires>
    </n:alertcontrol>
  </env:Header>
  <env:Body>
    <m:alert xmlns:m="http://example.org/alert">
      <m:msg>Get up at 6:30 AM</m:msg>
    </m:alert>
  </env:Body>
</env:Envelope>
```

XML-RPC: Не конкурент, а альтернатива SOAP



Концепція XML-RPC

XML-RPC — дуже простий і ефективний протокол взаємодії веб-сервісів. Він не призначений для рішення глобальних завдань, як SOAP, але широко використовується в багатьох веб-розробках.

XML-RPC — це "... специфікація і набір реалізацій, які дозволяють програмному забезпеченню, що працює на різних операційних системах і в різних умовах, викликати процедури через Інтернет. Це вилучений виклик процедури з використанням HTTP як транспорту й XML як способу кодування. XML-RPC розроблений настільки простим, наскільки це можливо для складних структур дан, що підлягають передачі, обробці й прийманню". — [xmlrpc.com]

"Ми прагли, чистий, розширюваний і дуже простий формат. Він повинен представляти HtmlКодеру можливість заглянути у файл, що містить опис XML-RPC виклику, зрозуміти, що той робить і могли змінити його, щоб він заробив з першої або другої спроби... Ми також

прагли, щоб це був легко реалізований протокол, який може бути швидко адаптований для роботи в іншому середовищі або на інших операційних системах."- [xmlrpc.com]

WSDL

Мова опису веб-сервісів (*Web services Description Language*, WSDL) призначений для уніфікованого вистави зовнішніх інтерфейсів веб-служби. Поточна версія протоколу (на момент написання цієї лекції) WSDL 2.0 і вона має деякі відмінності від попередніх версій (див. табл. 1 і рис. 3).

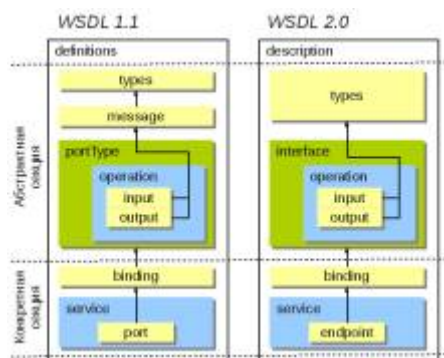


Рис. 3. Структура протоколу WSDL

Таблиця 1. Основні елементи протоколу WSDL.

Елемент WSDL 1.1	Елемент WSDL 2.0	Короткий опис
Porttype	Interface	Представляє опис інтерфейсу веб-сервісу (список операцій і їх параметрів).
Service	Service	Список системних функцій
Binding	Binding	Специфікує інтерфейси й задає параметри зв'язування із протоколом SOAP: стиль зв'язування (RPC/Document) і транспорт (SOAP). Ця секція доступна й для кожної з операцій
Operation	Operation	Визначає операцію, що представляється веб-сервером. WsdlpОперація — це аналог традиційним функціям і процедурам.
Message	не викор.	Повідомлення, пов'язане з певною операцією. Містить інформацію, необхідну для виконання даної операції. Кожне повідомлення може складатися з декількох логічних частин, що описують типи даних і імена атрибутів. У версії 2.0 було виключено, тому що була впроваджена підтримка XML Schema для всіх елементів.
Types	Types	Опис даних відповідно до XML Schema.

У специфікації WSDL 1.1 було визначено 4 шаблону обміну повідомленнями (типи операцій):

- Односпрямовані операції (One-way): операція може ухвалювати повідомлення, але не буде повертати відповідь.
- ЗапитВідповідь (Request-response): операція може прийняти запит і повинна повернути відповідь.
- ПитанняВідповідь (Solicit-response): операція може надіслати запит і буде чекати відповідь на нього.
- Повідомлення (Notification): операція може послати повідомлення, але не буде очікувати відповідь.

У версії WSDL 2.0 ці шаблони змінені й розширені у бік підтримки повідомлень про помилки (наприклад, шаблон Robust-in-only), але для зворотної сумісності підтримуються типи WSDL 1.1

Пример описания веб-сервиса на языке WSDL (версия 2.0).

UDDI



Рис. 4. Місце UDDI у стеці протоколів веб-служб

Universal Description, Discovery and Integration (UDDI, універсальний інтерфейс розпізнавання, опису й інтеграції) — відкритий стандарт, затверджений OASIS, що визначає методи публікації й виявлення мережних програмних компонентів сервіс-орієнтованої архітектури (SOA). У практичній реалізації UDDI являє собою мережний реєстр (службу каталогов), що представляє дані й метадані про веб-сервіси й доступний за адресою <http://uddi.xml.org/services>.

UDDI опирається на галузеві стандарти HTTP, XML, XML Schema (XSD), SOAP і WSDL. Концептуальний зв'язок між UDDI і іншими протоколами стека веб-сервісів показано на рисунку 4.

Функціональне призначення реєстру UDDI — вистава даних і метаданих про веб-служби. Він може використовуватися як у мережі загального користування, так і в межах внутрішньої інфраструктури організації. Реєстр UDDI пропонує заснований на стандартах механізм класифікації, каталогізації й керування веб-службами, що дозволяє застосовувати їх (веб-сервіси) іншими додатками. Цей механізм включає засоби для пошуку сервісу, виклику цієї служби, а також для керування метаданими про цю службу.

Ключовими функціями UDDI є публікація інформації про службу в реєстрі й пошук цієї інформації сторонніми додатками. Поряд із цими, реалізовані й типові для служби каталогів функції: вистава моделі збережених даних і структури інформаційної бази, відносини між

об'єктами реєстру, реплікація, забезпечення безпеки і т.д. усі основні функції реєстру представлені у вигляді веб-сервісів і доступні через API UDDI.

Веб-сервіси: Pro et Contra

Гідності

- Забезпечують взаємодія програмних систем незалежно від платформи.
- Засновані на базі відкритих стандартів і протоколів.
- Використання HTTP дозволяє додаткам взаємодіяти через міжмережний екран.

Недоліки

- Менша продуктивність і більший обсяг мережного трафіка в порівнянні з такими технологіями як CORBA або DCOM.

Мережні технології: тенденції й перспективи розвитку

У розвитку мережних технологій явно виділяються три основні тенденції: ріст числа підключених мобільних клієнтів, удосконалювання наявних і поява нових веб-сервісів і збільшення частки онлайнового відеотрафіка.

Голос з минулого

«Американці потребують телефону, а ми — ні. У нас є багато посильних.» Сер У. Прис, головний інженер британського поштового відомства, 1878 рік.

«Хто, чорт побори, прагне чути розмова акторів?» Г.М. Уорнер, Warner Bros, 1927 рік.

«Я думаю, що світовий ринок може досягти п'яти комп'ютерів.» Томас Ватсон, керівник IBM, 1943 рік.

«Телебачення не зможе провести на будь-якому захопленому їм ринку й перших шести місяців. Людям скоро набридне дивитися на фанерний ящик щоночі.» Дэррил Занук, 20th Century Fox, 1946 рік.

У першім десятилітті XXI століття Інтернет «перемінив статус» із глобальної комп'ютерної мережі на «глобальний інформаційний простір», виявивши себе як у соціальній, так і в економічній сферах і продовжуючи розвиватися. Можливість доступу до Мережі не тільки з комп'ютера, але й з інших пристроїв популярність, що росте, онлайнових версій традиційно оффлайнових телекомунікаційних послуг (телефонія, радіо, телебачення), унікальні онлайнові сервіси — усе це сприяє триваючому росту числа користувачів Інтернет і, як наслідок, збільшенню трафіка. За прогнозами компанії Cisco, представленим в «Індекс розвитку візуальних сетевих технологій», до 2015 року глобальний обсяг трафіка перевищить 50 ексабайт (при 22 ексабайтах в 2010 році). Лівову частину в генерації трафіка займе онлайнове відео, обсяг якого в 2011 році вперше перевищив сукупний трафік інших типів (голос+дані). До 2015 року обсяг відеотрафіка складе більш 30 ексабайт (при 14-15 ексабайт в 2010 р.). Основним засобом доступу до контенту залишиться Інтернет, при тому,

що збільшиться частка трафіка з мобільних пристроїв, прямо підключених до цієї мережі. Обсяг голосового трафіка збільшиться незначно, тому що на зміну «телефонному» голосовому спілкуванню йде відеотелефонний зв'язок.

Доступ до ресурсів

Прогнозоване збільшення мережної активності можливо вплине на прискорений перехід телекомунікаційних компаній від наявної мережної інфраструктури до реалізації концепції *мультисервісної мережі* (рис. 1).

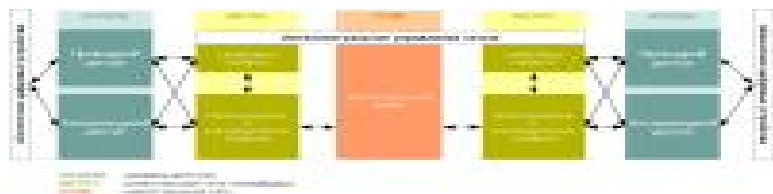


Рис. 1. Концепція мультисервісної мережі

Мультисервісна мережа — це мережне середовище, здатна передавати аудіо-, відеопотоки й дані в уніфікованому (цифровому) форматі по єдиному протоколу (мережний рівень: IP v6). Пакетна комутація, використовувана замість комутації каналів, робить мультисервісну мережу постійно готовою до використання. Протоколи резервування смуги пропускання, керування пріоритетами передачі і якості обслуговування (Qos) дозволяють диференціювати послуги, надавані для різних типів трафіка. Це гарантує прозоре й однакове підключення до мережі й одержання доступу до мережних ресурсів і сервісам як для існуючих клієнтських пристроїв, так і для тих, що з'являться в найближчій майбутньому. Провідний доступ у мультисервісній мережі стане ще швидше, а мобільний — ще подешевшає.

ІнтернетРадіо

Потокове ІнтернетРадіо з'явилося наприкінці 90х років ХХ в. і швидко набрало популярність. Провідні радіостанції представили користувачам можливість слухати ефірні програми через браузер. З ростом кількості мережних радіостанцій сторонні розроблювачі стали пропонувати користувачам спеціалізовані клієнтські додатки — ІнтернетРадіо програвачі.

Прикладом Інтернет радіоплеєра є програма «Radiocent». Крім основної функції, онлайн онлайн-радіо, цей плеєр представляє наступні можливості: доступ до десятків тисяч (!) ІнтернетРадіостанцій; гнучке керування списком відтворення; пошук музики й радіо онлайн по країнах і жанрам; можливість робити запис із ефіру у форматі mp3. WindowssВерсію програми «Radiocent» можна безкоштовно скачати на офіційному сайті.



Інтерфейс програми «Radiocent»

Сервіси

Відеозв'язок стане основним видом абонентського зв'язку, а телебачення переживе трансформацію, у результаті якої фактично відбудеться злиття телевізора й персонального комп'ютера. Телевізори із вбудованим браузером уже є на ринку, а через 3-5 років навіть уже

в Україні провайдери будуть представляти не «оцифроване» ефірне телебачення, а сьогодення цифрове (інтерактивність + HDTV).

Частка онлайн-ових мультимедійних сервісів збільшиться, фільми й музика онлайн стане доступніше і якісніше.

Ринок програмного забезпечення зміститься у бік додатків для мобільних пристроїв, таких як смартфони й планшети. Найбільшу популярність придбають веб-сервіси, що замінюють традиційно оффлайн-ові додатки. Працювати з мережними пакетами прикладних програм можна буде через Інтернет по моделі «програмне забезпечення як сервіс». Для ПК буде розроблятися тільки 20%-25% програмних продуктів.

Розвиток інтернет-комерції приведе до росту числа товарів і послуг, які можна буде замовити в мережних маркетах. Звичайний досвід здійснення покупок може повністю змінитися: не потрібно буде йти в магазин за продуктами. Досить буде зі смартфона зайти на сайт супермаркету й зробити замовлення потрібних продуктів, відразу ж зі смартфона його оплатити й дочекатися доставки.

Розвиток інтернет-банкінга приведе до появи додатків «клієнтбанк» для смартфонів. Візування фінансових операцій у такому додатку буде виконуватися біометрично або сенсорними «жестами» на тачскрині.

Сервіси «віртуальної реальності» дозволять «побачити» себе в автомобілі моделі, що сподобався, або «примірити» одяг певного типу в заданих умовах.

Дослідники компанії Gartner вважають, що до кінця 2016 року більш половини великих компаній будуть зберігати інформацію про своїх клієнтів і користувачів не на власних серверах, а в дата-центрах сервіс-провайдерів, щоб заощадити на ІТ витратах. Ріст популярності всіляких «хмар» робить цей прогноз обґрунтованим.

Згідно із прогнозами Gartner, аналітичні відділи організацій не зможуть упоратися зі збільшенням обсягу даних і ефективно використовувати цей потік інформації в комп'ютерних системах підтримки прийняття рішень до прояву нових інтелектуальних алгоритмів аналізу даних.

Візуалізація й користувацькі інтерфейси

Уже давно мова йде про кардинальну зміну користувацьких інтерфейсів цифрових пристроїв. Актуальна поки ще концепція WIMP повільно, але вірно поступається позиціями інтелектуальному середовищу SILK. Але... безліч слів із приводу майбутніх технологій не замінюють декількох відеороликів. Пропоновані фрагменти ілюструють перспективи розвитку користувацьких інтерфейсів і технологій візуалізації мережних додатків (і не тільки).

Прогнози — справа невдячна?

«Потенційний світовий ринок копіювальних машин — 5000 штук, у крайньому випадку.» З листа керівників IBM до майбутніх засновників Xerox, 1959 рік.

«Немає ніяких підстав припускати, що хто-небудь захоче мати комп'ютер у своєму будинку.» Кен Олсен, засновник Digital Equipment Corp, 1977 рік.

«Нікому не знадобиться більше, чим 637 кб пам'яті для персональних комп'ютерів — 640 кб буде досить для будь-якого», Білл Гейтс, Microsoft, 1981 рік.

« До наступного різдва Irod буде мертвий, закінчиться, піде, йому капут.» Сер Алан Шугар, британський бізнесмен, 2005 рік.

Ложка дьогтю

Як би райдужні перспективи в розвитку мережних технологій нас не очікували, але платити доводиться за все. І не завжди грошми. Найвища ціна, яку виставить користувачам Мережа майбутнього — *втрата приватності*.

Прикриваючись боротьбою з тероризмом, захистом авторських прав, політичним і релігійним екстремізмом і іншими бутафорськими небезпеками, уряду доведуть до досконалості автоматизовані системи тотального контролю над користувачами. Аж до ведення довічної перепони всіх дій в інтернеті. Вмонтовані чипи постійно підключених до Мережі інтернет-гаджетів дозволять Великому Братові стежити за кожним кроком громадянина й маніпулювати його поведінкою. Реалізувати це буде легко — шляхом відключення повсякденних у майбутньому сервісів або, навпаки, таємним підключенням спеціалізованих мережних служб. Приватність, якщо збережеться, стане привілеєм невеликого числа «товстих гаманців».