# 3. ЗВ'ЯЗОК

**Зв'язок між процесами** – це спосіб взаємодії й обміну даними між процесами. Сучасні розподілені системи часто містять тисячі процесів, розкиданих ненадійною мережею, наприклад Internet. Щоб спростити розробку масштабних прикладних систем, необхідно замінити найпростіші засоби взаємодії в телекомунікаційних мережах.

## 3.1. Рівні протоколів

Еталонна модель взаємодії відкритих систем (Open Systems Interconnection Reference Model) OSI/ISO спрощує роботу з безліччю рівнів і понять, використовуваних у передачі даних, та чітко визначає сім рівнів, дає їм стандартні імена і вказує, який рівень за що відповідає. Модель OSI розроблено для того, щоб надати відкритим системам можливість взаємодіяти.

**Відкритою** називають систему, здатну взаємодіяти з будь-якою іншою відкритою системою за стандартними правилами, що визначають формат, вміст і зміст повідомлень, які відправляються і приймаються.

Ці правила зафіксовані в чинних стандартах і їх називають **протоколами** (protocols). Усі протоколи поділяють на два основні типи. Перший тип – протоколи *зі встановленням з'єднання (connection-oriented),* у яких перед початком обміну даними відправник і одержувач мають встановити з'єднання і, можливо, домовитися про те, який протокол вони використовуватимуть. Після завершення обміну вони мають розірвати з'єднання. Системою зі встановленням з'єднання є, наприклад, телефон. Другий тип – використання протоколів *без установлення з'єднання (connectionless),* у такому разі підготовки не потрібно. Відправник надсилає перше повідомлення, як тільки він готовий це зробити.

У моделі OSI взаємодію поділяють на сім рівнів, як показано на рис. 3.1. Кожен рівень відповідає за один специфічний аспект взаємодії, надає інтерфейс для роботи з вищим рівнем. Інтерфейс містить набір операцій, які ра-

зом визначають інтерфейс, що надається тим рівнем, яким користується поточний рівень.

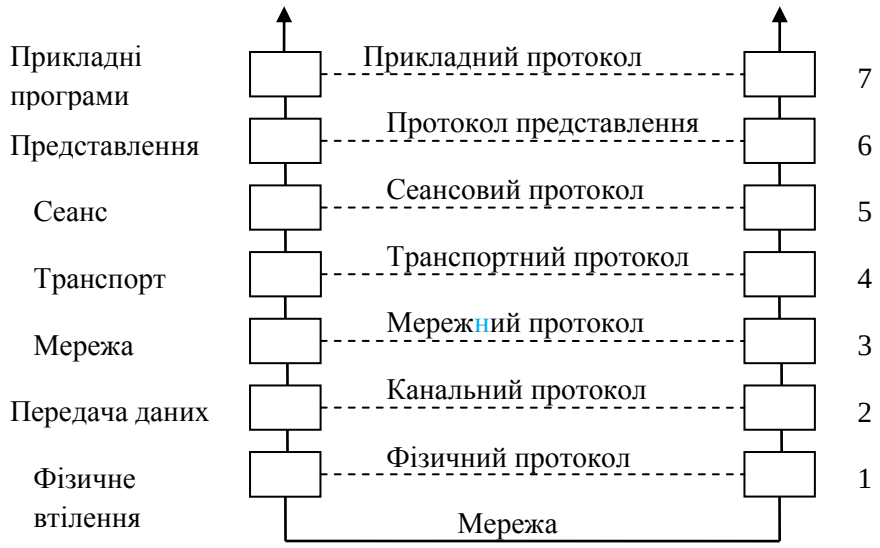


*Рис. 3.1.* Рівні, інтерфейси і протоколи OSI

**Приклад.** Коли процес *A* на машині *1* хоче поспілкуватися з процесом *B* на машині *2*, він створює повідомлення й надсилає його прикладному рівню своєї машини. Програмне забезпечення прикладного рівня додає в початок повідомлення свій *заголовок (header)* і передає отримане повідомлення через інтерфейс із рівня *7* на рівень *6*. Рівень представлення (6), у свою чергу, додає в початок повідомлення свій заголовок і передає результат вниз, на сеансовий рівень (5) і т. д. Деякі рівні додають не лише заголовок у початок, але і завершення в кінець. Коли повідомлення дійде до фізичного рівня, останній виконає його реальну передачу, як це показано на рис. 3.2.

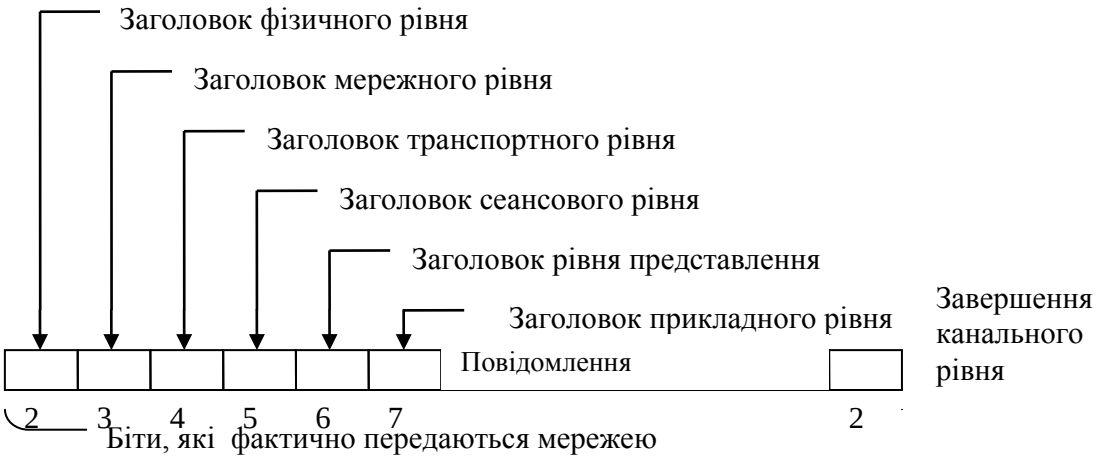


*Рис. 3.2*. Формат повідомлення, переданого мережею

Коли повідомлення надходить на машину *2*, воно передається вгору, при цьому на кожному рівні зчитується й перевіряється відповідний заголовок. Зрештою повідомлення досягає одержувача, процеса *B*, що може відповісти на нього, у процесі надання відповіді одержувачем усе повторюється у зворотному напрямку. Інформацію із заголовка рівня *n* використає протокол рівня *n*.

### *3.1.1. Низькорівневі протоколи*

**Фізичний рівень.** Фізичний рівень відповідає за передачу нулів й одиниць, а його протоколи - за стандартизацію електричних, механічних і сигнальних інтерфейсів, щоб, якщо одна машина надсилає нуль, то друга прийняла його як нуль, а не як один: ADSL (Asymmetric Subscriber Line), ISDN (Integrated Services Digital Network), SONET (Synchronous Optical Networking), стандартні модемні протоколи (протоколи серії ITU V), що використовуються в з'єднаннях між аналоговими модемами телефонною лінією тощо.

**Канальний рівень.** Фізичний рівень тільки пересилає біти. Поки немає помилок, все добре, однак у реальних мережах з'являються помилки, їх потрібно знаходити й виправляти. Це і є основним завданням канального рівня, оскільки він групує біти в модулі, які зазвичай називають **кадрами** (frames), і стежить за тим, щоб кожен кадр було передано правильно за рахунок розміщення спеціальної бітової маски в початок і кінець кожного кадру для їх маркування, а також за рахунок обчислення **контрольної суми** *(checksum)*, тобто підсумовування всіх байтів кадру. Канальний рівень додає контрольну суму до кадру. Коли кадр приймається, приймач повторно обчислює контрольну суму даних і порівнює результат з контрольною сумою, що надійшла разом з кадром. Якщо вони збігаються, то кадр уважається правильним і приймається; якщо різні, то одержувач просить відправника повторно надіслати цей кадр. Кадри послідовно нумеруються із зазначенням номерів у заголовку, таким чином, що всі розуміють, де який кадр.

**Приклад.** Протоколи: ARCNET (Attached Resource Computer NETwork), CDP (Cisco Discovery Protocol), DCAP (Data Link Switching Client Access Protocol), DTP (Dynamic Trunking Protocol), L2FP (Layer 2 Forwarding Protocol), L2TP (Layer 2 Tunnelling Protocol) тощо.

**Мережний рівень.** Повідомлення, яке надсилається від відправника до одержувача, має пройти безліч мережних сегментів, на кожному з яких обирається вихідний напрям. Вибір найкращого шляху називають *маршрутизацією (routing)* і це є основним завданням мережного рівня.

Нині найбільш поширеним мережним протоколом, що не потребує встановлення з'єднання, є *протокол Internet (Internet protocol, IP)*, який входить до стеку протоколів Internet. Для іменування повідомлення на мережному рівні використовують термін «**пакет»** (packet). IP-пакет може бути надісланий без попередньої підготовки, маршрут кожного з IP-пакетів до місця призначення обирається незалежно від інших пакетів. Жодний внутрішній шлях не обирається заздалегідь і не запам'ятовується.

**Приклад.** Протоколом із з'єднанням є *віртуальний канал (virtual channel)* на основі мереж ATM (Asynchronous Transfer Mode), мережний протокол без установлення з'єднання CLNP (Connection Less Network Protocol).

**Віртуальний канал в ATM** – це непряме з'єднання, встановлене від джерела до приймача, яке може проходити через кілька проміжних ATM-комутаторів.

Протоколи *без установлення з'єднання:* протокол IPv4(Internet Protocol version 4), IPv6 (Internet Protocol version 6), IGRP (Interior Gateway Routing Protocol), IPSec (Internet Protocol Security) тощо.

### *3.1.2. Транспортні протоколи*

**Транспортний рівень** – це верхня частина того, що називають базовим стеком мережних протоколів, оскільки в ньому реалізовано всі служби, які необхідні для побудови мережних прикладних програм і які не ввійшли в інтерфейс мережного рівня.

Після одержання повідомлення з прикладного рівня транспортний рівень розбиває його для успішної передачі на достатньо дрібні частини, призначає їм послідовні номери й пересилає їх. Взаємодія на рівні заголовка транспортного рівня зводиться до обговорення того, який пакет був посланий, який прийнятий, скільки місця є в адресата для прийому подальших повідомлень, що варто послати повторно тощо.

**Приклад.** Транспортний протокол для Internet називають *протоколом керування передачею (Transmission Control Protocol, TCP)*. Комбінація TCP/IP зараз є стандартом де-

факто у разі мережної взаємодії. Комплект протоколів Internet також містить транспортний протокол *UDP (Universal Datagram Protocol – універсальний протокол датаграм)*, який не потребує з'єднання та фактично є протоколом IP з деякими незначними доповненнями.

Офіційний транспортний протокол ISO має п'ять різновидів – від TP0 до TP4. Відмінності полягають у процесах обробки помилок і можливості працювати з декількома транспортними з'єднаннями на основі одного з'єднання низького рівня (особливо X.25). Вибір того, який з них використати, залежить від властивостей протоколу, який лежить нижче мережного рівня. Жоден з них не має перевантажуватися.

### *3.1.3. Протоколи верхнього рівня*

**Сеансові протоколи і протоколи представлення. Сеансовий рівень** фактично є розширеною версією транспортного рівня, оскільки забезпечує керування діалогом, відслідковуючи й запам'ятовуючи, яка сторона говорить у цей момент, і надає інформацію синхронізації. Ця інформація потрібна, щоб користувачі створювали контрольні точки у разі довгих сеансів передачі даних та повідомлення про збій у процесі такого сеансу. Контрольні точки необхідні для того, щоб зробити відкат лише до останньої контрольної точки, а не проходити весь шлях спочатку. На практиці сеансовий рівень використовується деякими прикладними програмами, підтримується рідко і навіть не входить до комплекту протоколів Internet.

Рівень представлення відповідає за зміст переданих бітів. Більшість повідомлень містять не випадкові послідовності бітів, а структуровану інформацію (прізвища, адреси, грошові суми тощо). На рівні представлення можна визначити записи, які містять подібні поля, і вимагати у відправника повідомляти одержувача, що повідомлення містить окремі записи відповідного формату.

**Прикладні протоколи. Прикладний рівень** моделі OSI містить набір стандартного мережного прикладного програмного забезпечення, наприклад для роботи з електронною поштою, для передачі файлів й емуляції термінала. На сьогодні він став місцем розташування всього прикладного програмного забезпечення і протоколів, які не вдалося розмістити на жодному з нижчих рівнів. З погляду еталонної моделі OSI всі розподілені системи є просто розподіленим між обчислювальними вузлами прикладним програмним забезпеченням.

**Приклад.** Популярний в Internet *протокол передачі файлів (File Transfer Protocol, FTP)* визначає передачу файлів між клієнтською машиною й сервером. Цей протокол не слід плутати з програмою *ftp*, що являє собою прикладну програму для передачі файлів і збігається (але не повністю) з реалізацією протоколу FTP для Internet.

Іншим прикладом спеціального прикладного протоколу може бути *протокол передачі гіпертексту (Hypertext Transfer Protocol, HTTP)*, розроблений для віддаленого керування й завантаження web-сторінок, реалізований у таких прикладних програмах, як web-браузери й web-сервери.

**Протоколи проміжного рівня.** До **проміжного рівня** належать прикладні програми, які розміщують логічно на прикладному рівні, але використовують низку протоколів загального призначення, що дає їм право на власний рівень, який не залежить від інших та охоплює більше спеціалізованих прикладних програм. Можна відокремити *високорівневі протоколи взаємодії* від *протоколів для надання різних служб* проміжного рівня.

**Приклад.** Як приклад розглянемо групу *протоколів розподіленого підтвердження (commit),* які розробляють так, щоб окрема група процесів або всі процеси пройшли через певну операцію, або операцію не було застосовано до жодного з них. Це явище відоме також як атомарність, яку широко використовують у процесі виконання транзакцій. Не тільки транзакції, але й інші прикладні програми, особливо які мають бути стійкими до збоїв, також можуть мати потребу в використанні протоколів розподіленого підтвердження. До таких протоколів належать такі: *протокол SOAP (Simple Object Access Protocol)*, група *протоколів XMPP (Extensible Messaging and Presence Protocol), протокол UMSP (Unifired Memory Space Protocol)* та інші.

Комунікаційні протоколи проміжного рівня підтримують високорівневі комунікаційні служби, з яких розглянемо лише чотири високорівневі комунікаційні служби проміжного рівня: віддалений виклик процедур, віддалене звертання до об’єктів, черги повідомлень і потоки даних.

Такий підхід до поділу на рівні зумовлює дещо змінену еталонну модель взаємодії (рис. 3.3). Порівняно з моделлю OSI сеансовий рівень і рівень представлення замінено одним проміжним рівнем, що містить протоколи, які не залежать від прикладних програм і їх не можна помістити на нижчі рівні. Транспортні служби також можуть бути подані у вигляді служб проміжного рівня, не потребуючи навіть модифікації. Цей підхід аналогічний перенесенню UDP на транспортний рівень.

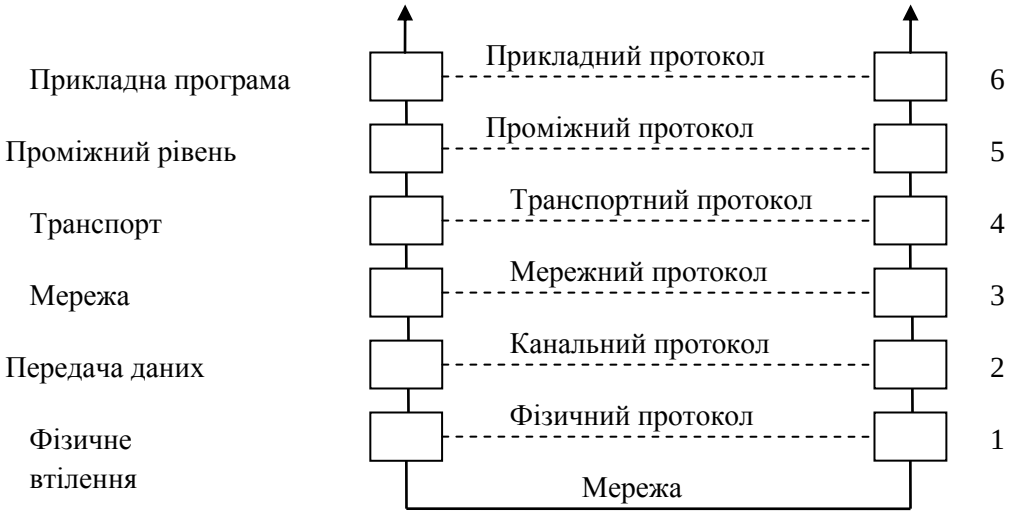


*Рис. 3.3*. Змінена еталонна модель мережної взаємодії

## 3.2. Віддалений виклик процедур

Основою більшості розподілених систем є явний обмін повідомленнями між процесами, однак процедури *send* та *receive* не приховують взаємодії, необхідної для забезпечення прозорості доступу. Було запропоновано дозволити програмам викликати процедури, які перебувають на інших машинах. Коли процес, запущений на машині *A*, викликає процедуру з машини *B*, то процес, який викликається на машині *A* припиняється, а виконання викликаної процедури відбувається на машині *B*. Інформація може бути передана від процесу, який викликає, до процедури, яка викликається, через параметри й повернута процесу у вигляді результату виконання процедури. Цей метод називають **віддаленим викликом процедур** (Remote Procedure Call, RPC*).*

Ідея **виклику віддалених процедур** полягає в розширенні добре відомого і зрозумілого механізму підміни процесу керування обчисленнями й даними, які містяться в середині програми, що виконується на одній машині, на процес, який виконується через мережу. Засоби віддаленого виклику процедур призначені для полегшення організації розподілених обчислень і створення розподілених клієнт-серверних інформаційних систем. Найбільша ефективність використання RPC досягається в тих прикладних програмах, у яких наявний інтерактивний зв’язок між віддаленими компонентами з невеликою