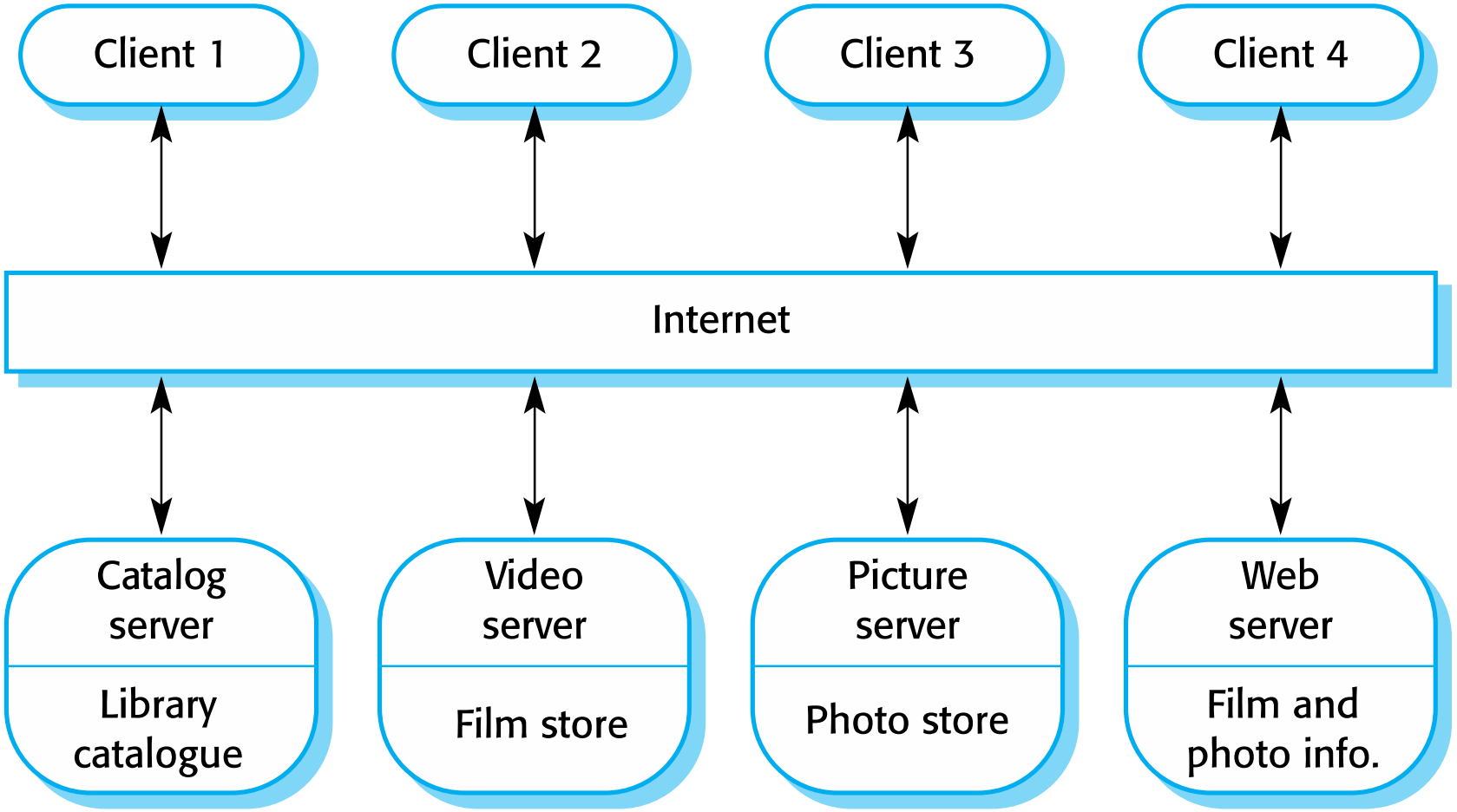
資料庫系統
Class 5: Design of Applications

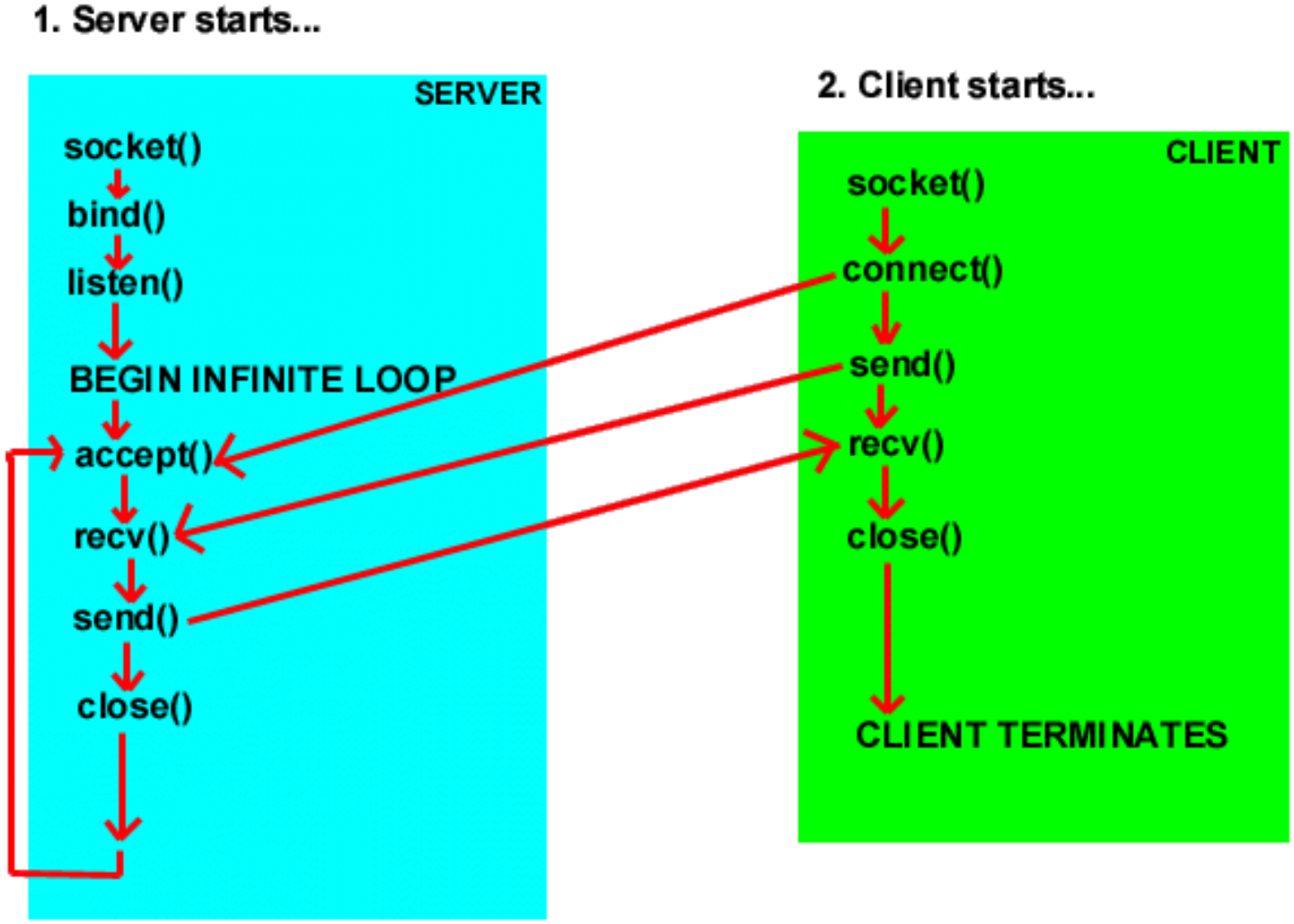
逢甲資工 許懷中

主從式架構 (Client-Server)

- 伺服器: 提供服務
- 用戶端: 使用服務
- 網路: 服務的媒介
- 分散式系統的基礎，描述資訊如何在不同的元件與節點中被處理以及傳遞

主從式架構 (cont.)

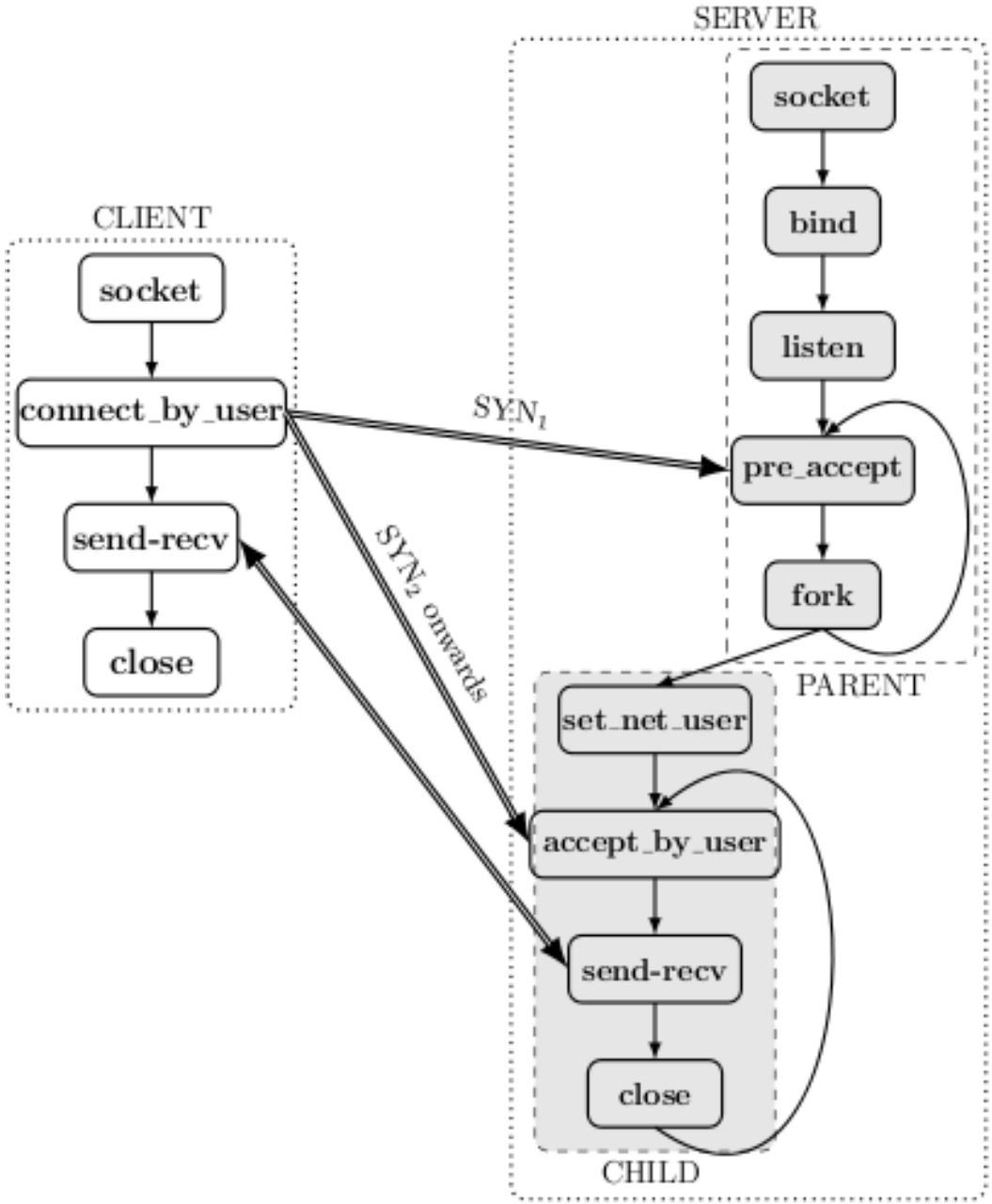

Client 1
Client 2
Client 3
Client 4
Internet
Catalog server
Library catalogue
Video server
Film store
Picture server
Photo store
Web server
Film and photo info.

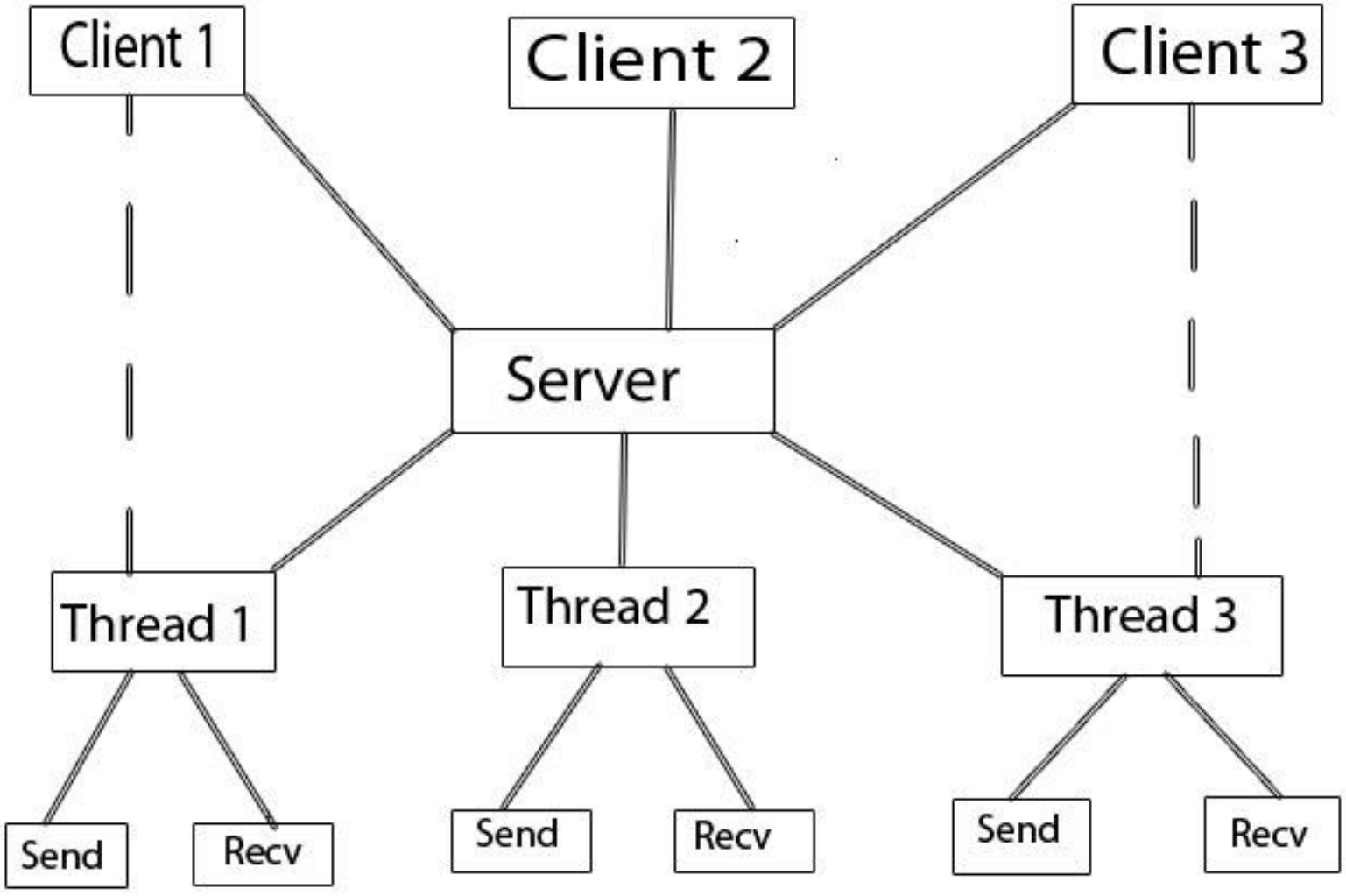


The echo client and server

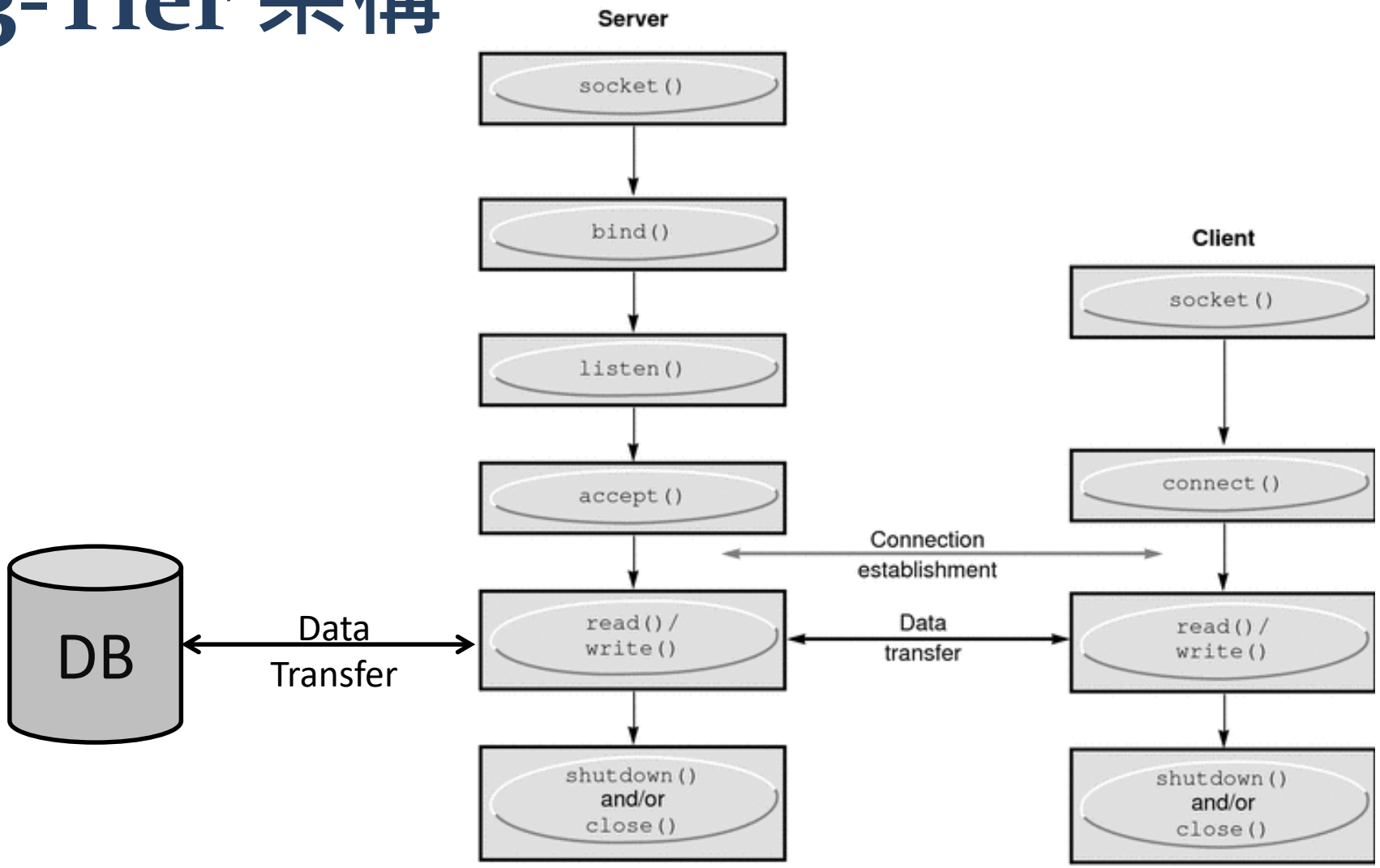
如何與多 Client 連線

- Client 不維持連線，每次取得資料後就斷線
  - Stateless connection
- Child Process
  - fork()
- Multi-threading

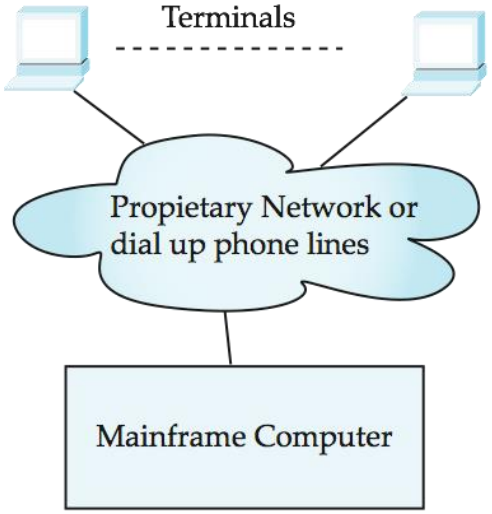

SERVER
socket
bind
listen
pre_accept
fork
PARENT
CLIENT
socket
connect_by_user
send-recv
close
SYN_1
SYN_2 onwards
set_net_user
accept_by_user
send-recv
close
CHILD


Client 1
Client 2
Client 3
Server
Thread 1
Thread 2
Thread 3
Send
Recv
Send
Recv
Send
Recv

3-Tier 架構

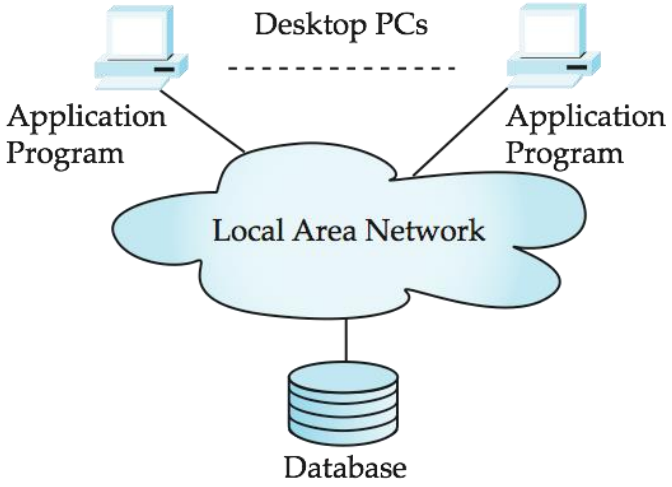

Server
socket()
bind()
listen()
accept()
read()/
write()
shutdown()
and/or
close()
Client
socket()
connect()
read()/
write()
shutdown()
and/or
close()
Connection
establishment
Data
transfer
DB
Data
Transfer

Applications 與 Database



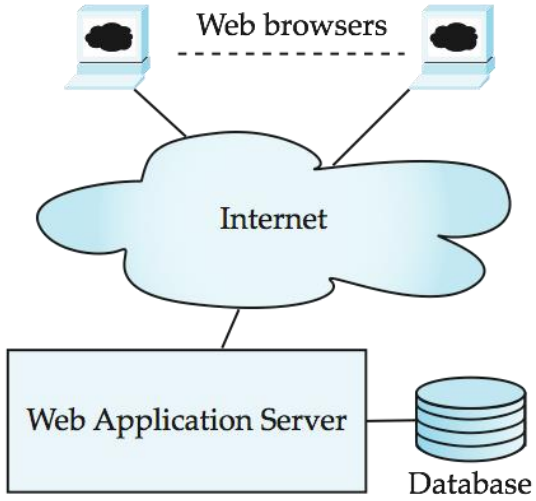
(a) Mainframe Era

1960's-70's



(b) Personal Computer Era

1980's

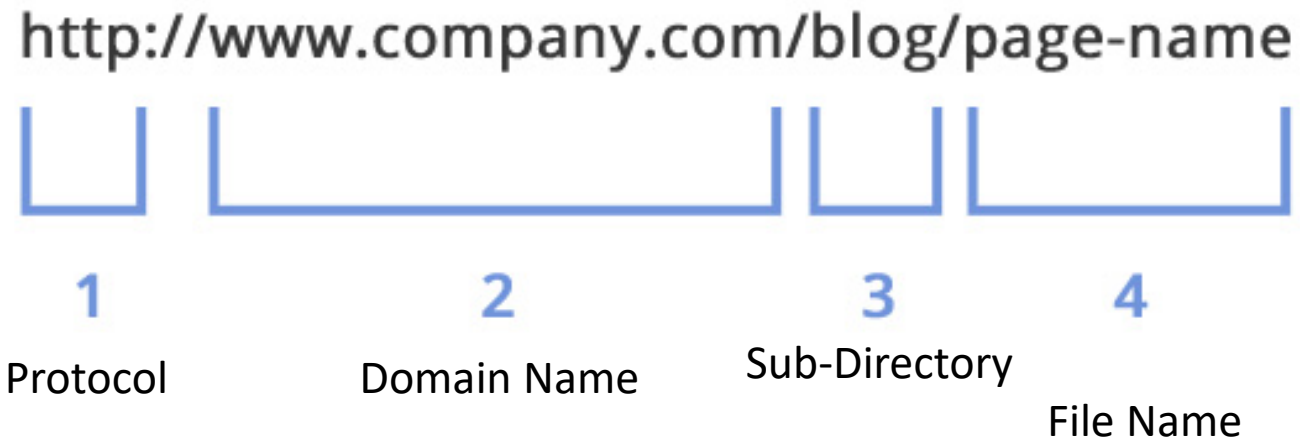


(c) Web era

1990 – present

URL

- Uniform Resource Locator

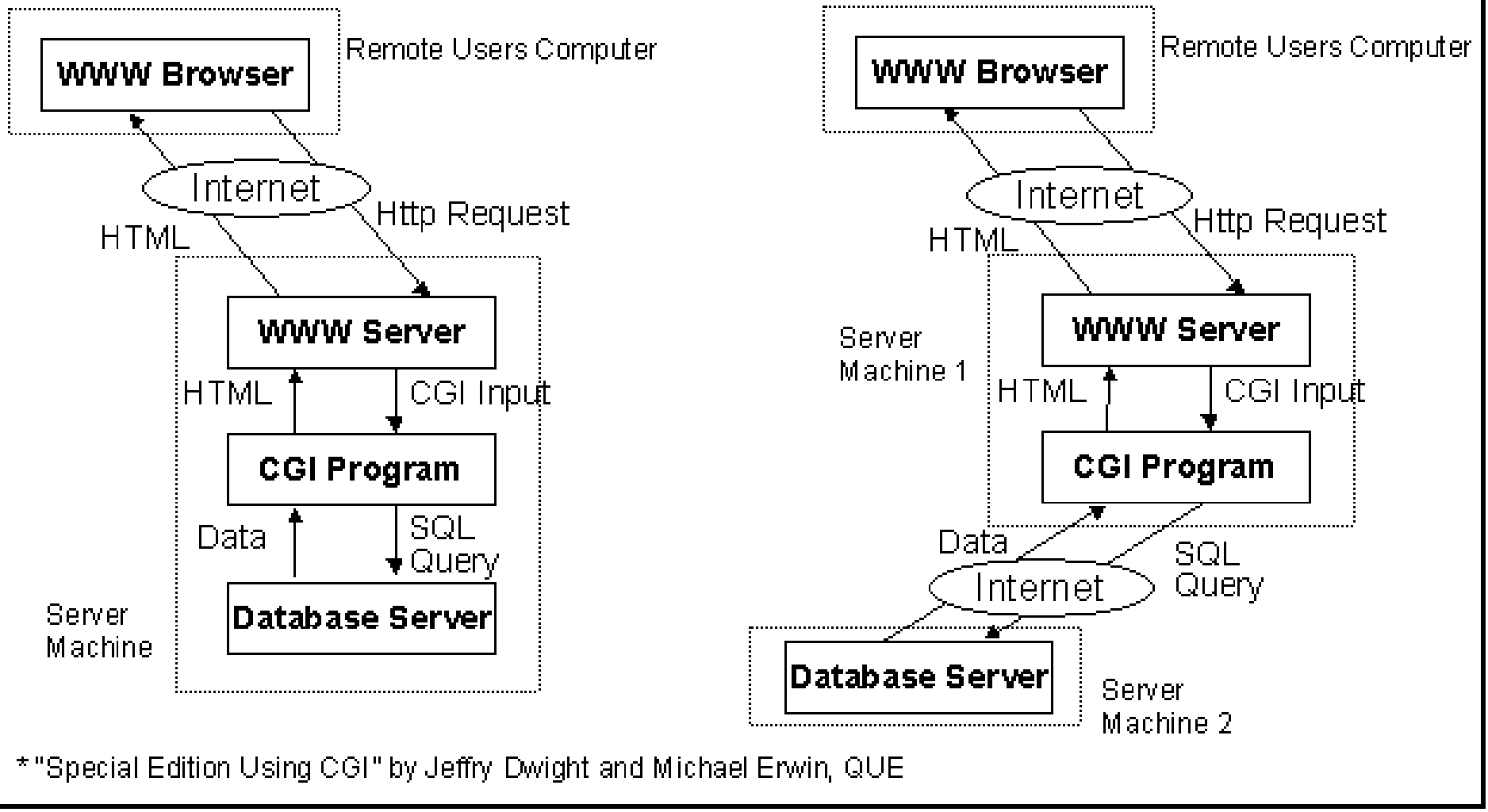


HTML 與 HTTP

- **HyperText Markup Language**
  - 在文件中嵌入圖形與影音
  - 超文件連結
  - Form
- **HyperText Transfer Protocol**
  - Request 與 Response
  - Connectionless

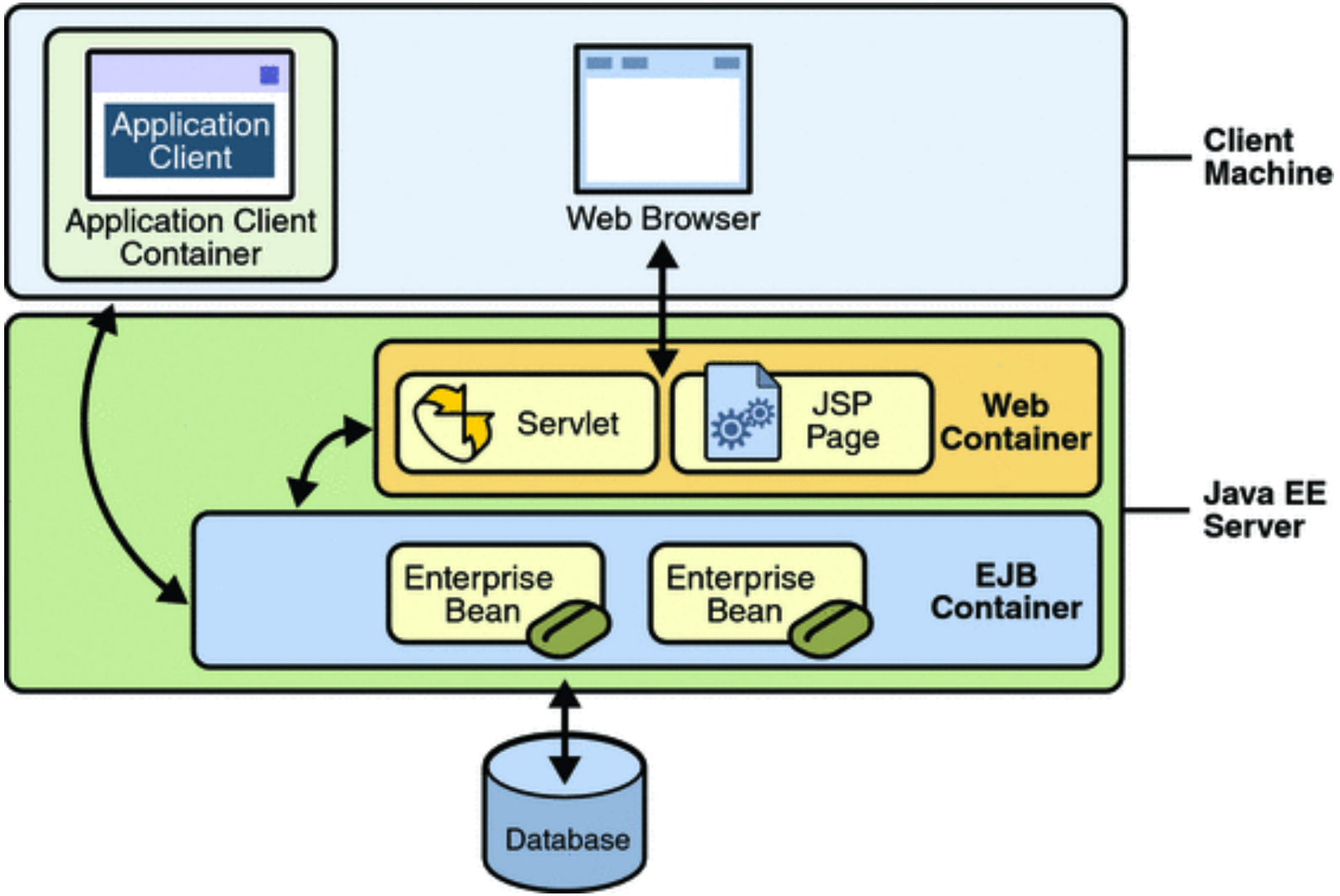
Web Server

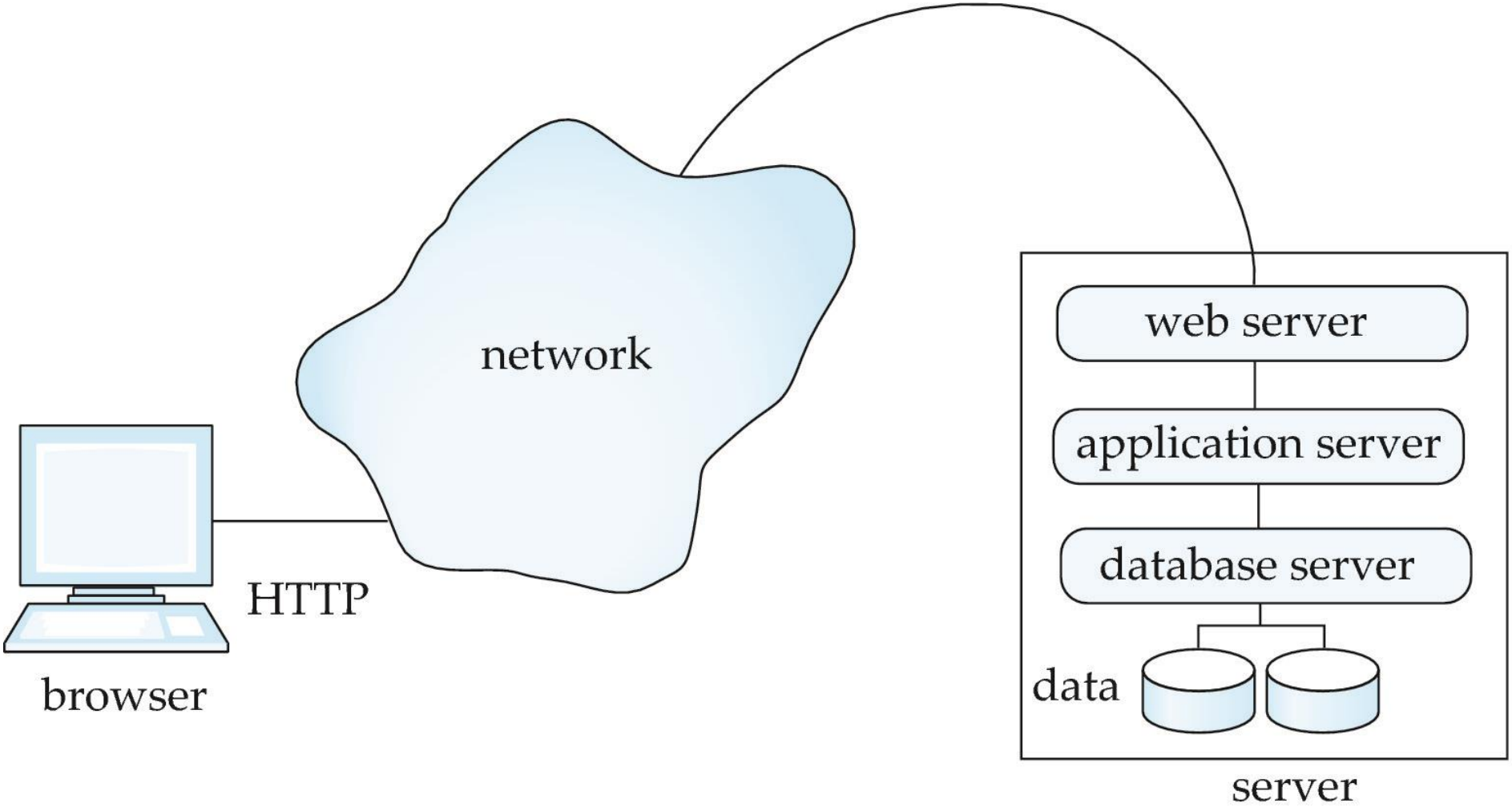
- 接受 Request (URL)，傳回超文件
- 靜態網頁與動態網頁

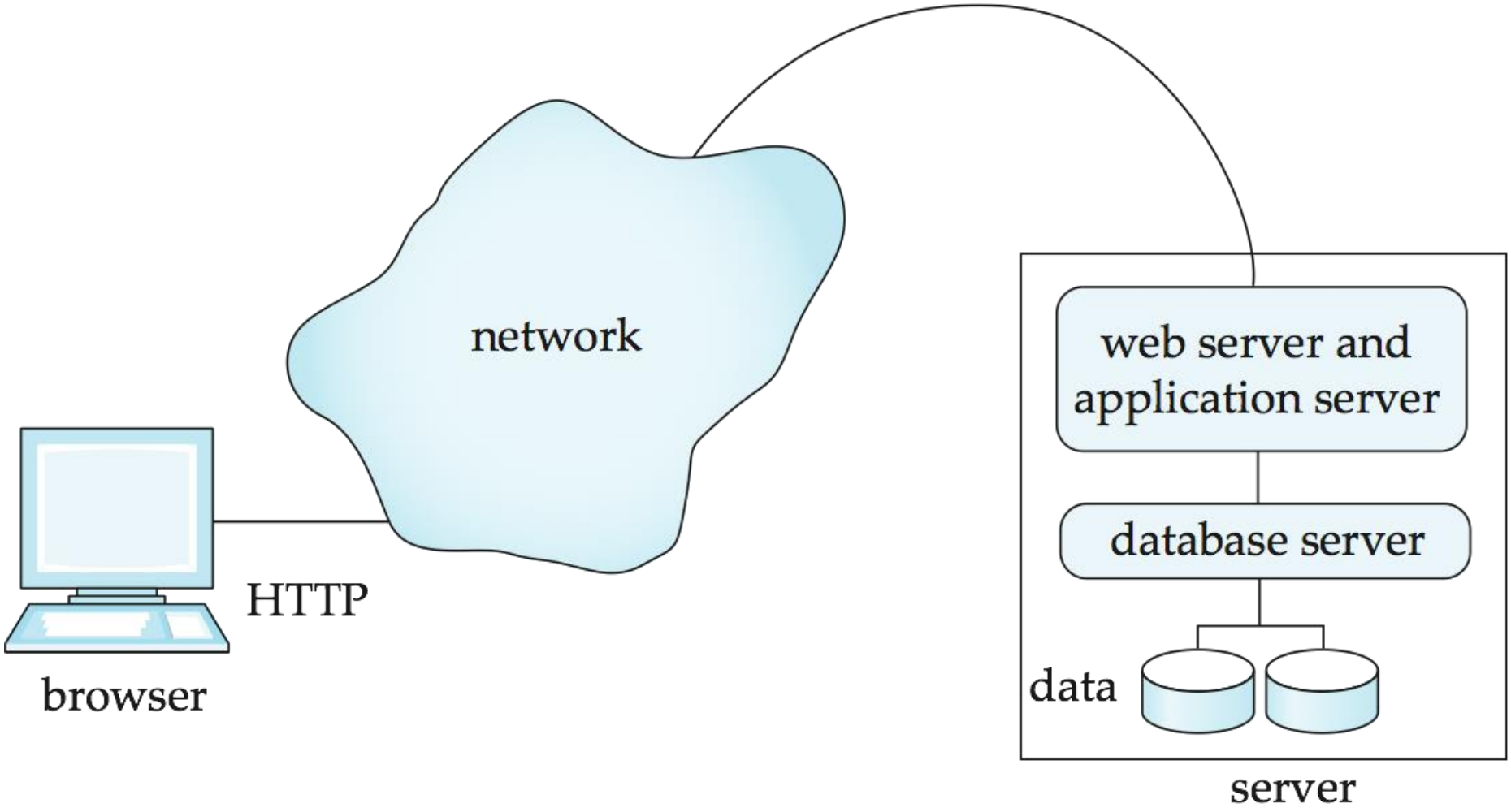


* "Special Edition Using CGI" by Jeffry Dwight and Michael Erwin, QUE

Java 網頁應用架構


Application Client
Application Client Container
Web Browser
Client Machine
Servlet
JSP Page
Web Container
Java EE Server
Enterprise Bean
Enterprise Bean
EJB Container
Database


network
web server
application server
database server
data
HTTP
browser
server


network
web server and
application server
database server
data
HTTP
browser
server

HTTP 與 Sessions

- HTTP 是 connectionless 通訊協定
  - Client 與 Server 之間不會維持連線狀態
  - 每次發出要求都是一個新的連線
- 維持用戶的狀態？
  - 保持使用者登入 => 一個使用者的 session
  - 利用 cookie 達成

動態網頁

- 程式產生網頁
  - CGI, Servlet
- 伺服器端 Script
  - Python, PHP, JSP, ASP

Servlet Sample Code

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class PersonQueryServlet extends HttpServlet {
  public void doGet (HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException
  {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<HEAD><TITLE> Query Result</TITLE></HEAD>");
    out.println("<BODY>");
      ..... BODY OF SERVLET (next slide) ...
    out.println("</BODY>");
    out.close();
  }
}
```

Servlet Sample Code (cont.)

```
String persontype = request.getParameter("persontype");
String number = request.getParameter("name");
if(persontype.equals("student")) {
    ... code to find students with the specified name ...
    ... using JDBC to communicate with the database ..
    out.println("<table BORDER COLS=3>");
    out.println(" <tr> <td>ID</td> <td>Name: </td>" + " <td>Department</td> </tr>");
    for(... each result ...){
        ... retrieve ID, name and dept name
        ... into variables ID, name and deptname
        out.println("<tr> <td>" + ID + "</td>" + "<td>" + name + "</td>" + "<td>" + deptname
                + "</td></tr>");
    };
    out.println("</table>");
}
else {
    ... as above, but for instructors ...
}
```

Servlet 對於 Sessions 的支持

- 確認 Session 是否啟動
  - if (request.getSession(false) == true)
    - .. then existing session
    - else .. redirect to authentication page
  - authentication page
    - check login/password
    - request.getSession(true): creates new session
- 從 Session 中存取變數
  - session.setAttribute(“userid”, userid)
  - session.getAttribute(“userid”)

JSP Sample Code

```
<html>
<head> <title> Hello </title> </head>
<body>
<%
        if (request.getParameter("name") == null)
                { out.println("Hello World"); }
        else { out.println("Hello, " +
                request.getParameter("name")); }
%>
</body>
</html>
```

PHP Sample Code

```
<html>
<head> <title> Hello </title> </head>
<body>
<?php
        if (!isset($_REQUEST[‘name’]))
                { echo “Hello World”; }
        else { echo “Hello, ” + $_REQUEST[‘name’]; }
?>
</body>
</html>
```

Python Sample Code

```
from flask import Flask
from flask import request
app = Flask(__name__)

@app.route('/')
def index():
    name = request.args.get('name')
    if name is not None:
        return "Hello "+name
    else:
        return "Hello World"

if __name__ == '__main__':
    app.debug = True
    app.run()
```

Client 端 Script

- Sandbox
- 用途
  - 動畫
  - 輸入檢查
  - 具備彈性的互動功能
  - User Interface 的一部分

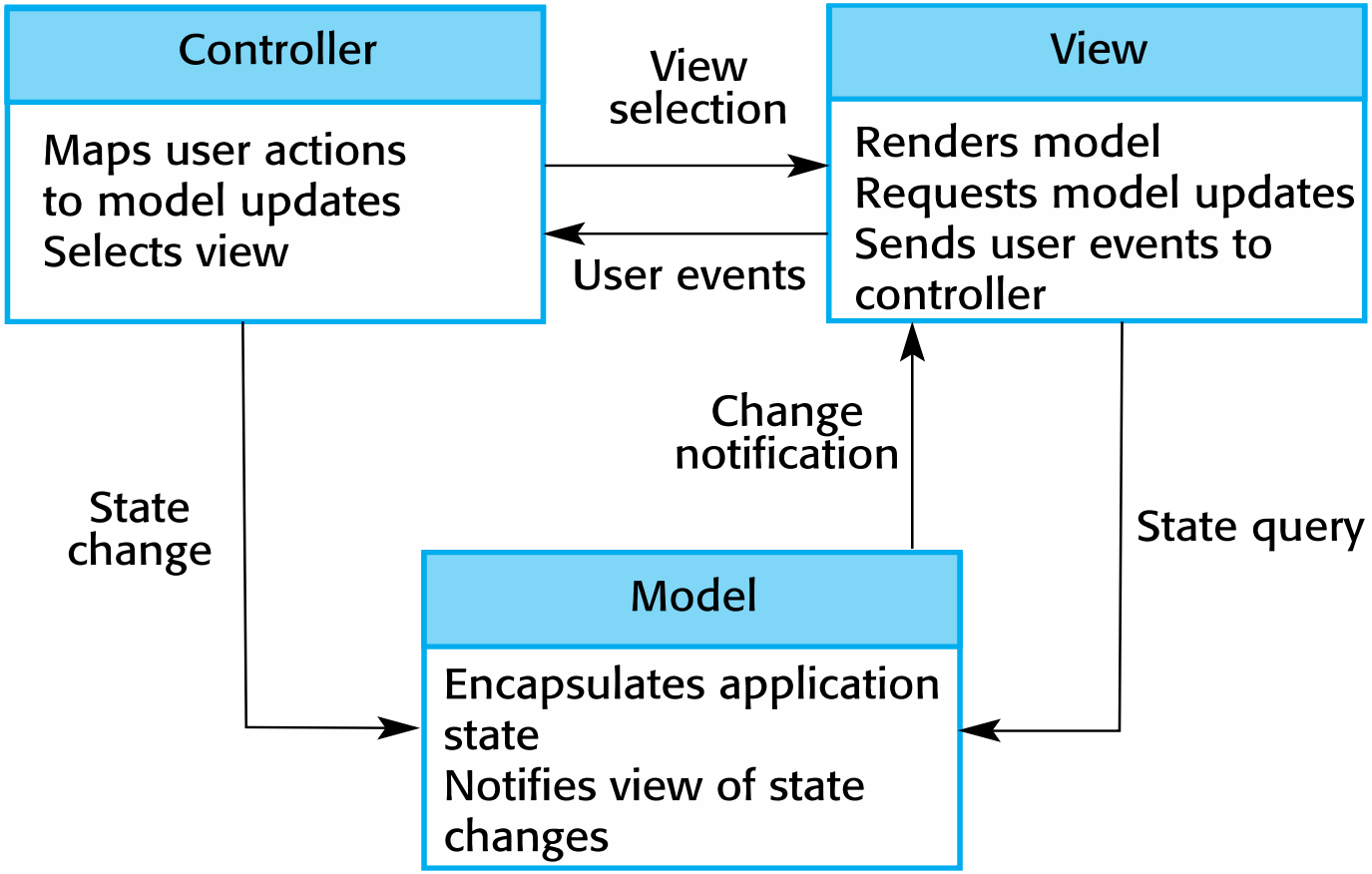
Javascript Sample Code

```
<html> <head>
   <script type="text/javascript">
      function validate() {
         var credits=document.getElementById("credits").value;
         if (isNaN(credits)|| credits<=0 || credits>=16) {
            alert("Credits must be a number greater than 0 and less
   than 16");
            return false
         }
      }
   </script>

</head> <body>
   <form action="createCourse" onsubmit="return validate()">
      Title: <input type="text" id="title" size="20"><br />
      Credits: <input type="text" id="credits" size="2"><br />
      <Input type="submit" value="Submit">
   </form>

</body> </html>
```

Design of Applications

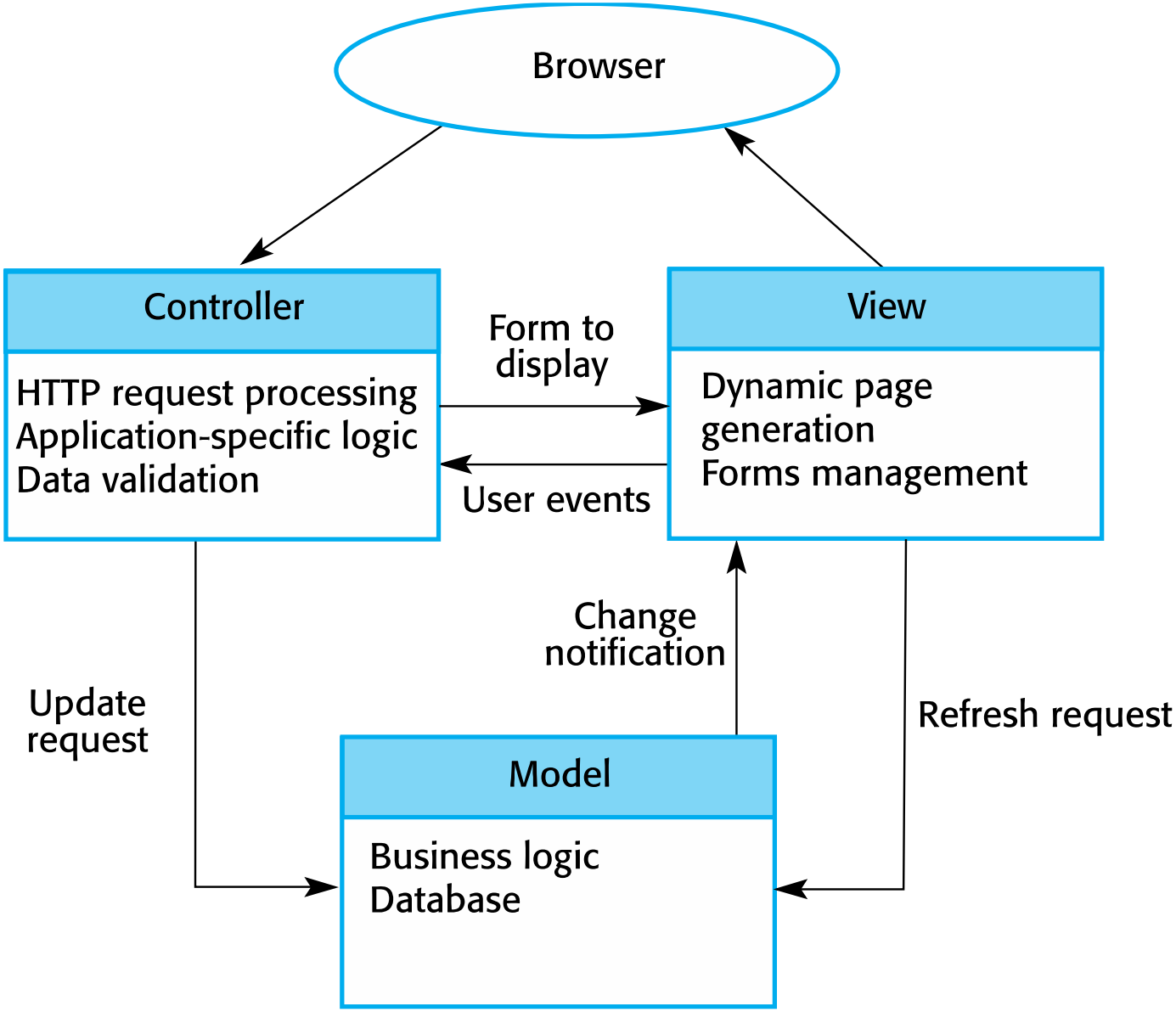
架構設計

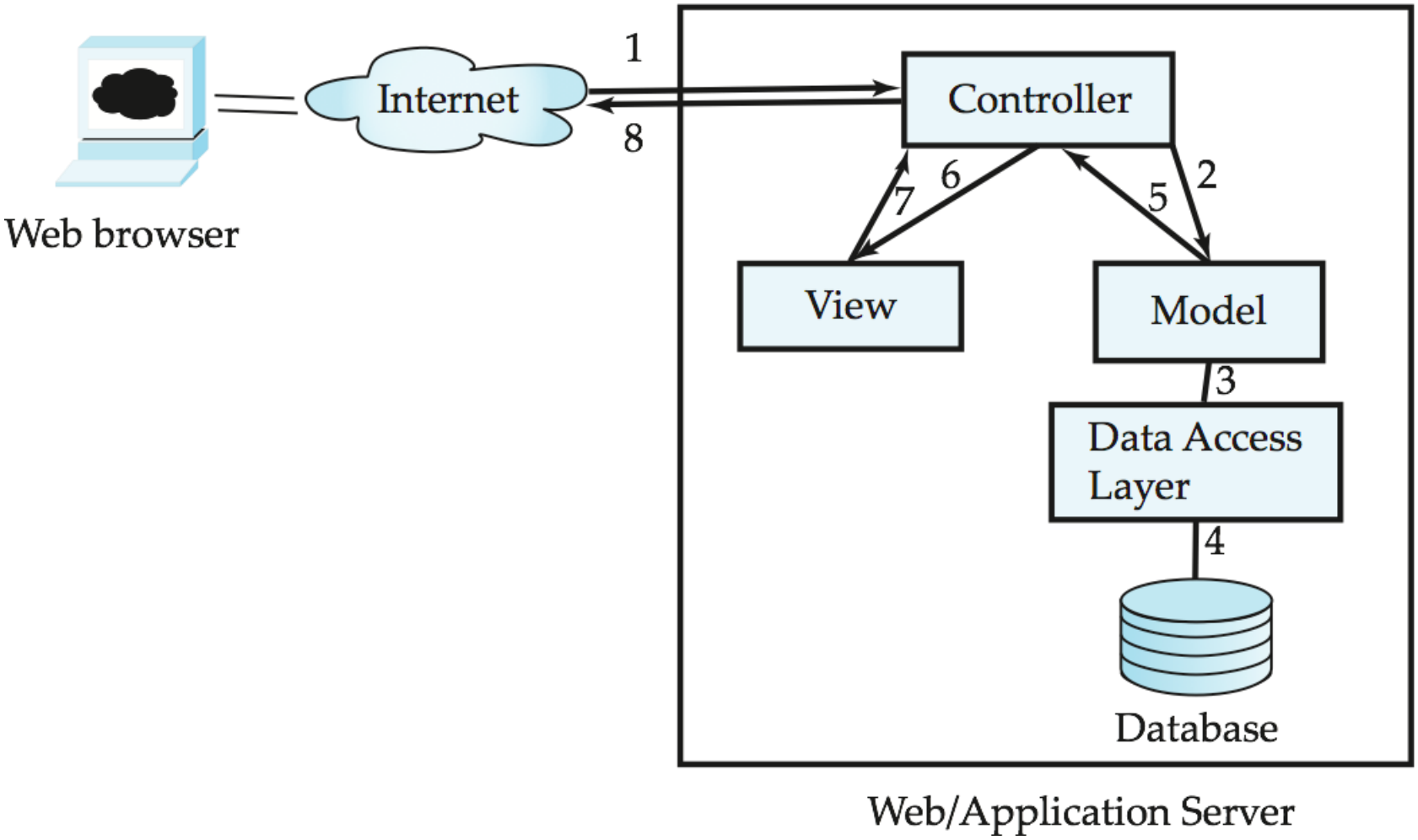
MVC 架構模式


Controller
Maps user actions to model updates
Selects view
View selection
User events
View
Renders model
Requests model updates
Sends user events to controller
State change
Change notification
State query
Model
Encapsulates application state
Notifies view of state changes

MVC 架構模式 (cont.)

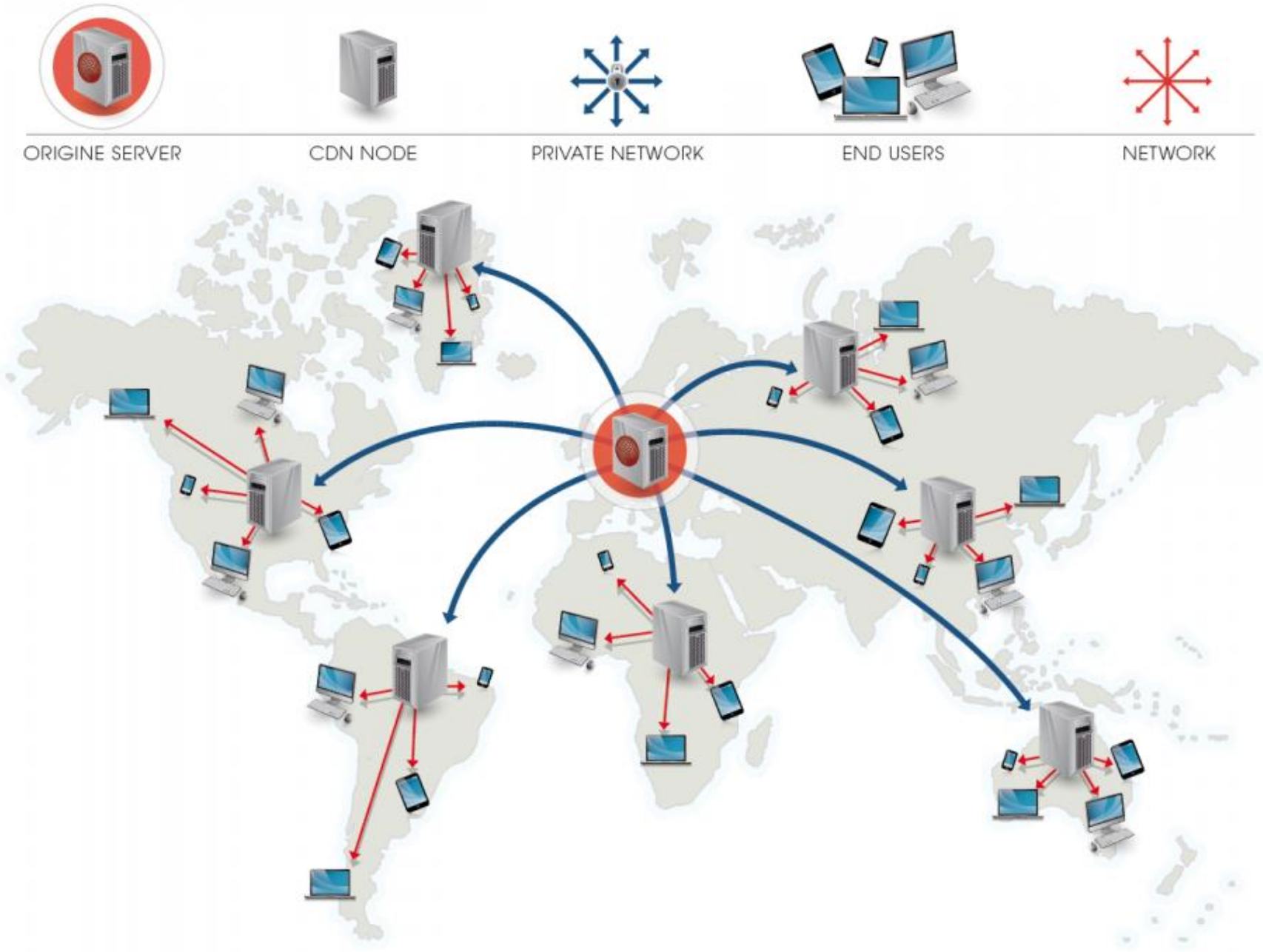
- 將展示 (Presentation)、控制邏輯以及資料三者分離
- 當系統有多種檢視方式時使用
- 展示、控制邏輯與資料模型可以獨立改變
- 當資料模型與展示間的互動很簡單時，採用 MVC 會增加額外的設計負擔


Browser
Controller
HTTP request processing
Application-specific logic
Data validation
Form to display
User events
View
Dynamic page generation
Forms management
Change notification
Update request
Refresh request
Model
Business logic
Database


Web browser
Internet
1
8
Controller
2
5
6
7
View
Model
3
Data Access Layer
4
Database
Web/Application Server

效能問題

- 在 Server 端增進效能
  - DB connection pooling
  - Caching results of DB queries
  - Caching generated HTML
- 在 Client 端增進效能
  - Web Proxy
  - CDN


ORIGINE SERVER
CDN NODE
PRIVATE NETWORK
END USERS
NETWORK

安全問題

- SQL Injection
- SQL 程式碼
  - select * from members where account='$name' and password='$password'
- 使用者輸入 *' or 1=1 /**
- 實際執行
  - select * from members where *account='' or 1=1 /*'* and password=''
- 防範方法: 使用 Prepared Statement

Prepared Statement for Preventing SQL Injection

```
String sqlString = "select * from db_user where username=? And
                    password=?";
PreparedStatement stmt = connection.prepareStatement(sqlString);
stmt.setString(1, username);
stmt.setString(2, pwd);
ResultSet rs = stmt.executeQuery();
```

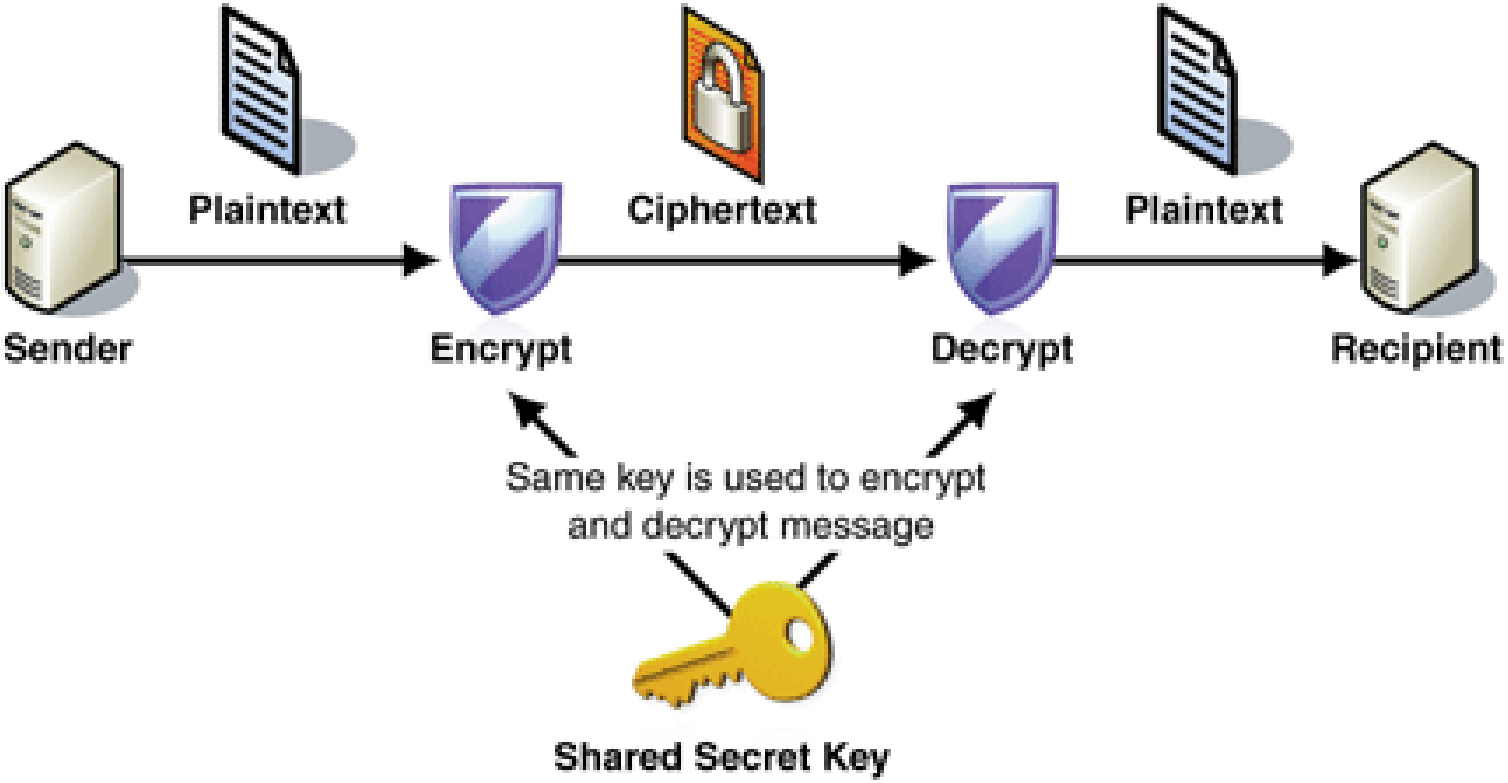
安全問題 (cont.)

- Cross-Site Scripting (XSS) 或 Cross-Site Request Forgery (XSRF)
- <img src = http://mybank.com/transfermoney?amount=1000&toaccount=14523>
- 避免網頁遭到利用
  - 禁止 User 輸入 URL
- 避免遭到這種攻擊
  - 使用 refer 確認來源網頁是 valid page
  - 確認來源 IP 與被授權使用者的 IP 相同
  - 不要用 GET method 進行 update 的動作

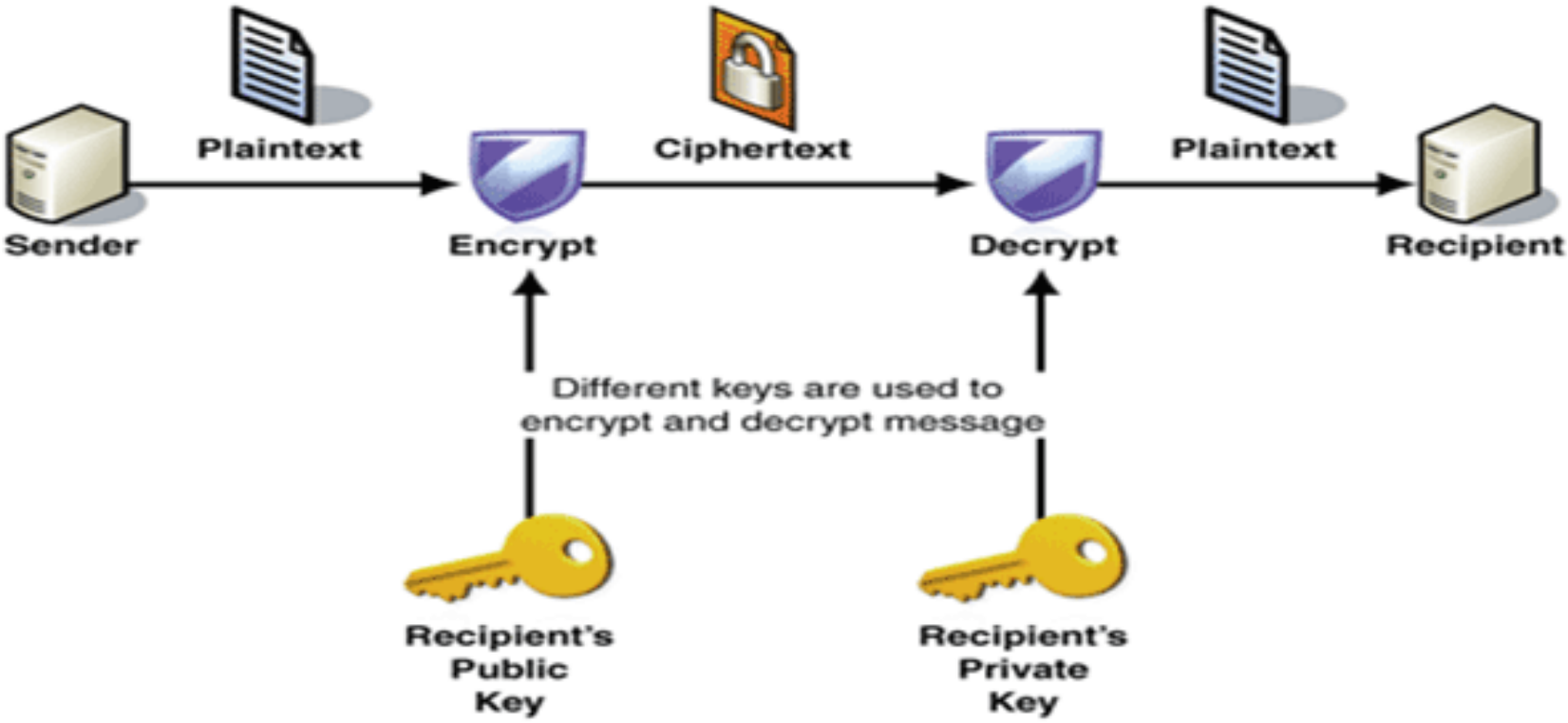
安全問題 (cont.)

- 不要在資料庫中儲存明文密碼
- 授權 (Authentication)
  - Two-Factor 授權
    - 密碼加一次性簡訊驗證
  - Men-in-the-Middle 攻擊
    - 防範方式：數位簽章、HTTPS
  - 集中式授權
  - 第三方授權

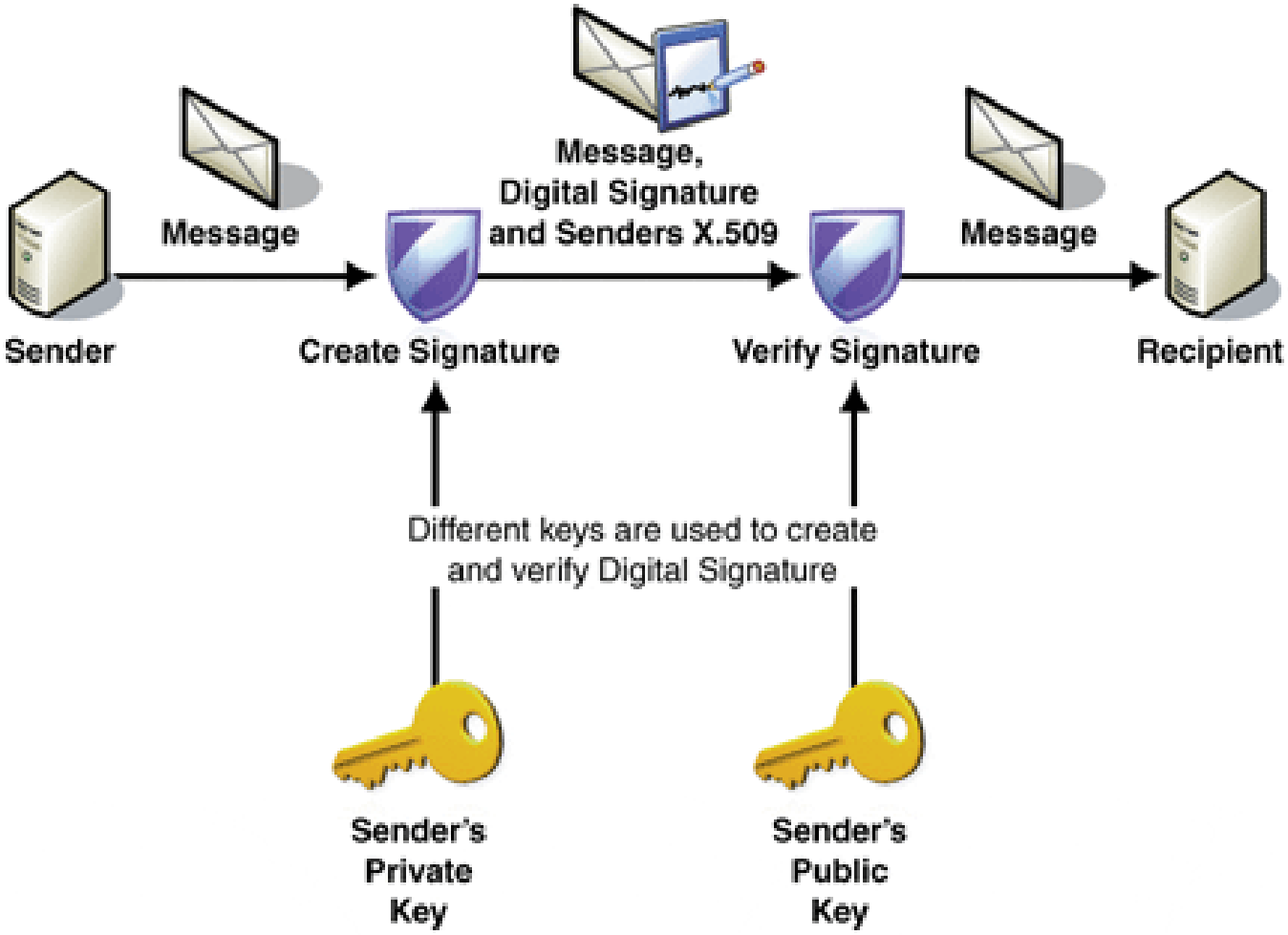
安全問題 (對稱加密)


Plaintext
Ciphertext
Plaintext
Sender
Encrypt
Decrypt
Recipient
Same key is used to encrypt
and decrypt message
Shared Secret Key

安全問題 (非對稱加密)


Plaintext
Ciphertext
Plaintext
Sender
Encrypt
Decrypt
Recipient
Different keys are used to
encrypt and decrypt message
Recipient's
Public
Key
Recipient's
Private
Key

安全問題 (數位簽章)


Message
Message,
Digital Signature
and Senders X.509
Message
Sender
Create Signature
Verify Signature
Recipient
Different keys are used to create
and verify Digital Signature
Sender's
Private
Key
Sender's
Public
Key