# Distributed Systems

## Architectural Styles

Faculty Of Information Technology

Distributed Systems

الانظمة الموزعة

Architectural Styles

الأساليب المعمارية

Faculty Of Information Technology

كلية تكنولوجيا المعلومات

# Architectural styles

## Basic idea

A style is formulated in terms of

- (replaceable) components with well-defined interfaces
- the way that components are connected to each other
- the data exchanged between components
- how these components and connectors are jointly configured into a system.

## Connector

A mechanism that mediates communication, coordination, or cooperation among components. Example: facilities for (remote) procedure call, messaging, or streaming.

Architectures: Architectural styles

الهندسة المعمارية: الأساليب المعمارية

Architectural styles

الأساليب المعمارية

Basic idea

الفكرة الأساسية

A style is formulated in terms of

يتم صياغة النمط من حيث

(replaceable) components with well-defined interfaces the way that components are connected to each other the data exchanged between components how these components and connectors are jointly configured into a system.

مكونات (قابلة للاستبدال) ذات واجهات محددة جيدًا، الطريقة التي ترتبط بها المكونات ببعضها البعض، البيانات المتبادلة بين المكونات، كيف يتم تكوين هذه المكونات والموصلات بشكل مشترك في النظام.

Connector

موصل

A mechanism that mediates communication, coordination, or cooperation among components.

آلية تتوسط الاتصال أو التنسيق أو التعاون بين المكونات.

Example: facilities for (remote) procedure call, messaging, or streaming.

على سبيل المثال: مرافق استدعاء الإجراء (عن بعد)، أو إرسال الرسائل، أو البث.

Using components and connectors, we can come to various configurations, which, in turn, have been classified into architectural styles. Several styles have by now been identified, of which the most important ones for distributed systems are:

- Layered architectures
- Object-based architectures
- Resource-centered architectures
- Event-based architectures

Clements, 1997].

كليمنتس، 1997].

For example, a connector can be formed by the facilities for (remote) procedure calls, message passing, or streaming data.

على سبيل المثال، يمكن تشكيل موصل بواسطة المرافق لاستدعاءات الإجراءات (عن بعد)، أو تمرير الرسائل، أو تدفق البيانات.

In other words, a connector allows for the flow of control and data between components.

بمعنى آخر، يسمح الموصل بتدفق التحكم والبيانات بين المكونات.

Using components and connectors, we can come to various configurations, which, in turn, have been classified into architectural styles.

باستخدام المكونات والموصلات، يمكننا الوصول إلى تكوينات مختلفة، والتي بدورها تم تصنيفها إلى أنماط معمارية.

Several styles have by now been identified, of which the most important ones for distributed

وقد تم حتى الآن تحديد عدة أنماط، أهمها هو الذي سيتم توزيعه

systems are:

الأنظمة هي:

• Layered architectures

• أبنية الطبقات

• Object-based architectures

• البنى القائمة على الكائنات

• Resource-centered architectures

• البنى التي تركز على الموارد

• Event-based architectures In the following, we discuss each of these styles separately.

• البنى المبنية على الأحداث فيما يلي، سنناقش كل نمط من هذه الأنماط على حدة.

We note in

نلاحظ في

advance that in most real-world distributed systems, many different styles are

تقدم أنه في معظم الأنظمة الموزعة في العالم الحقيقي، هناك العديد من الأنماط المختلفة

combined.

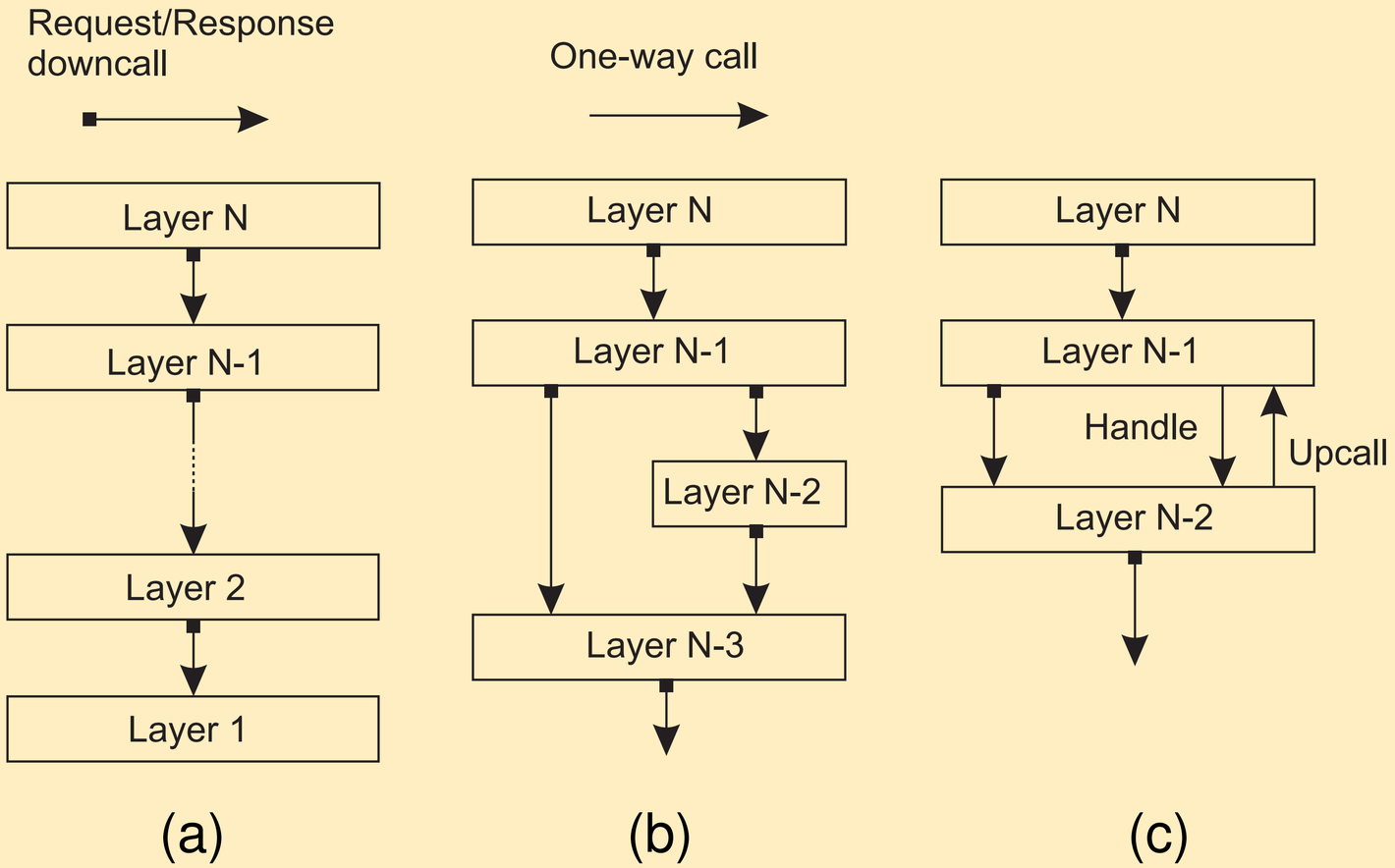
مجموع.

Notably following an approach by which a system is subdivided 3

ولا سيما اتباع النهج الذي يتم من خلاله تقسيم النظام 3

# Layered architecture

## Different layered organizations



Architectures: Architectural styles

الهندسة المعمارية: الأساليب المعمارية

Layered architectures

أبنية الطبقات

Layered architecture

العمارة الطبقات

Different layered organizations

منظمات الطبقات المختلفة

Request/Response

استجابة للطلب

One-way call

مكالمة في اتجاه واحد

Layer N

الطبقة ن

Layer N-1

Layer N-1

الطبقة N-1

Layer N-2

الطبقة N-2

Handle

مقبض

Upcall

اتصل

Layer 2

الطبقة 2

Layer N-3

الطبقة N-3

Layer 1

الطبقة 1

Figure 2.1(a) shows a standard organization in which only downcalls to the next lower layer are made. This organization is commonly deployed in the case of network communication.

In many situations we also encounter the organization shown in Figure 2.1(b). Consider, for example, an application A that makes use of a library $\mathsf{L}_{\mathsf{OS}}$ to interface to an operating system. At the same time, the application uses a specialized mathematical library $\mathsf{L}_{\mathsf{math}}$ that has been implemented by also making use of $\mathsf{L}_{\mathsf{OS}}$. In this case, referring to Figure 2.1(b), A is implemented at layer $N-1$, $\mathsf{L}_{\mathsf{math}}$ at layer $N-2$, and $\mathsf{L}_{\mathsf{OS}}$ which is common to both of them, at layer $N-3$.

Finally, a special situation is shown in Figure 2.1(c). In some cases, it is convenient to have a lower layer do an upcall to its next higher layer. A typical example is when an operating system signals the occurrence of an event, to which end it calls a user-defined operation for which an application had previously passed a reference (typically referred to as a **handle**).

Figure 2.1(a) shows a standard organization in which only downcalls to

يوضح الشكل 2.1 (أ) التنظيم القياسي الذي يتم فيه النداءات السفلية فقط

the next lower layer are made.

يتم تصنيع الطبقة السفلية التالية.

This organization is commonly deployed in the case of network communication.

يتم نشر هذه المنظمة بشكل شائع في حالة اتصالات الشبكة.

In many situations we also encounter the organization shown in Fig-

وفي العديد من المواقف نواجه أيضًا التنظيم الموضح في الشكل-

ure 2.1(b).

2.1(ب).

Consider, for example, an application A that makes use of a library

لنأخذ على سبيل المثال التطبيق (أ) الذي يستخدم المكتبة

LOS to interface to an operating system.

LOS للواجهة لنظام التشغيل.

At the same time, the application uses a specialized mathematical library Lmath that has been implemented by also making use of LOS.

وفي الوقت نفسه، يستخدم التطبيق مكتبة رياضية متخصصة Lmath تم تنفيذها من خلال الاستفادة أيضًا من LOS.

In this case, referring to Figure 2.1(b), A is implemented at

في هذه الحالة، وبالإشارة إلى الشكل 2.1(ب)، يتم تنفيذ A في

layer N − 1, Lmath at layer N − 2, and LOS which is common to both of them, at layer N − 3.

الطبقة N − 1، Lmath عند الطبقة N − 2، و LOS المشترك بينهما، عند الطبقة N − 3.

Finally, a special situation is shown in Figure 2.1(c).

وأخيرًا، يظهر موقف خاص في الشكل 2.1(ج).

In some cases, it

في بعض الحالات، ذلك

is convenient to have a lower layer do an upcall to its next higher layer.

من الملائم أن تقوم الطبقة السفلية بإجراء استدعاء للطبقة الأعلى التالية.

A typical example is when an operating system signals the occurrence of an event, to which end it calls a user-defined operation for which an application

والمثال النموذجي هو عندما يرسل نظام التشغيل إشارة إلى وقوع حدث ما، ومن أجل هذه الغاية يستدعي عملية محددة من قبل المستخدم والتي من أجلها يقوم التطبيق

had previously passed a reference (typically referred to as a handle).

قد مرر مرجعًا مسبقًا (يُشار إليه عادةً باسم المقبض).

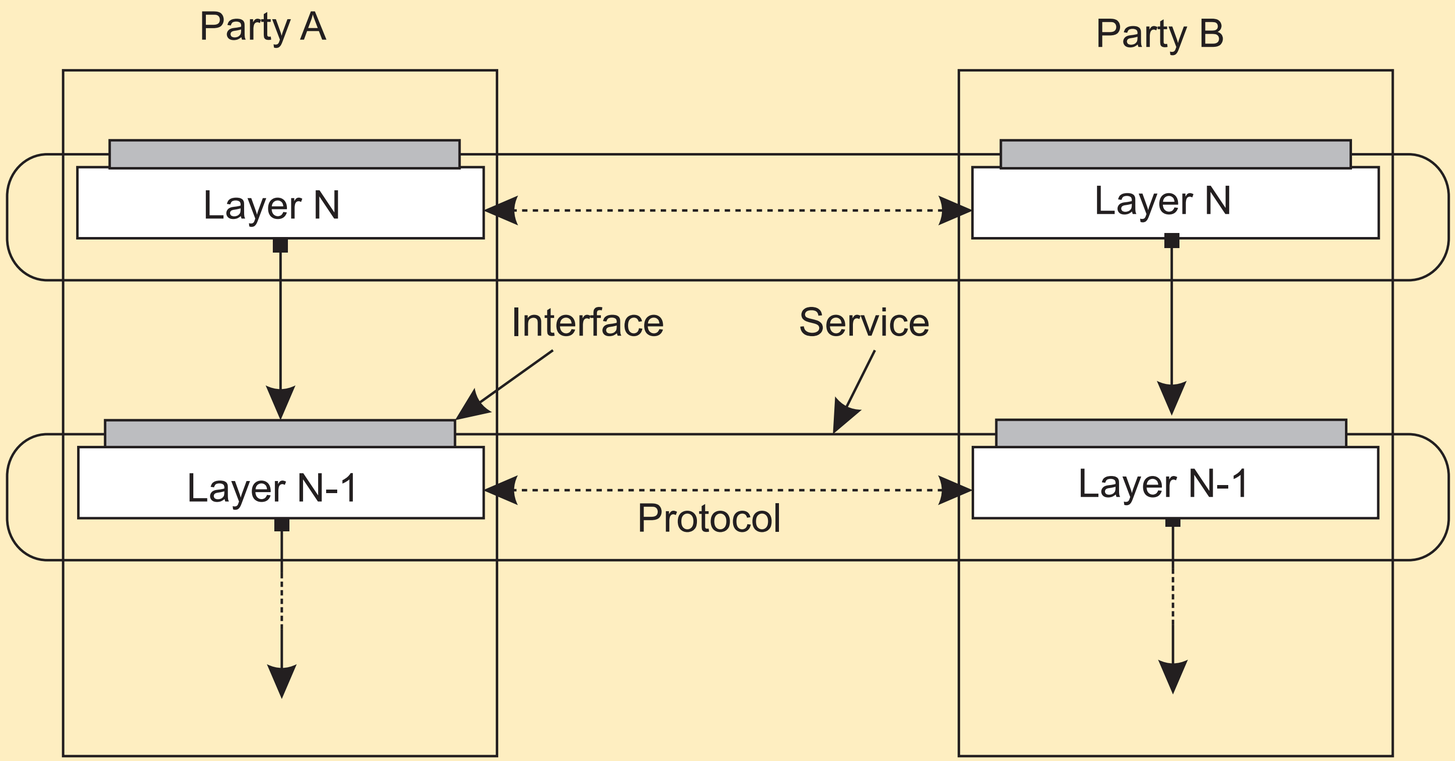


DS 3.01pre

DS 3.01ما قبل

# Example: communication protocols

## Protocol, service, interface



Architectures: Architectural styles

الهندسة المعمارية: الأساليب المعمارية

Layered architectures

أبنية الطبقات

Example: communication protocols

مثال: بروتوكولات الاتصال

Protocol, service, interface

البروتوكول، الخدمة، الواجهة

Party A

الحفله أ

Layer N

الطبقة ن

Party B

الحفلة ب

Interface

واجهه المستخدم

Service

خدمة

Layer N-1

الطبقة N-1

Protocol

بروتوكول

In communication-protocol stacks, each layer implements one or several **communication services** allowing data to be sent from a destination to one or several targets. To this end, each layer offers an **interface** specifying the functions that can be called. In principle, the interface should completely hide the actual implementation of a service. Another important concept in the case of communication is that of a **(communication) protocol**, which describes the rules that parties will follow in order to exchange information. It is important to understand the difference between a service offered by a layer, the interface by which that service is made available, and the protocol that a layer implements to establish communication. This distinction is shown in Figure 2.2.

called communication-protocol stacks.

تسمى مكدسات بروتوكول الاتصال.

We will concentrate here on the global picture only and defer a detailed discussion to Section 4.1.

سنركز هنا على الصورة العالمية فقط ونؤجل المناقشة التفصيلية للقسم 4.1.

In communication-protocol stacks, each layer implements one or several communication services allowing data to be sent from a destination to one or several targets.

في مكدسات بروتوكولات الاتصال، تنفذ كل طبقة خدمة اتصال واحدة أو أكثر مما يسمح بإرسال البيانات من وجهة إلى هدف واحد أو عدة أهداف.

To this end, each layer offers an interface specifying the functions that can be called.

ولتحقيق هذه الغاية، توفر كل طبقة واجهة تحدد الوظائف التي يمكن استدعاؤها.

In principle, the interface should completely hide the actual implementation of a service.

من حيث المبدأ، يجب أن تخفي الواجهة التنفيذ الفعلي للخدمة تمامًا.

Another important concept in the case of communication is that of a (communication) protocol, which describes the rules that parties will follow in order to exchange information.

المفهوم المهم الآخر في حالة الاتصال هو مفهوم بروتوكول (الاتصال)، الذي يصف القواعد التي ستتبعها الأطراف من أجل تبادل المعلومات.

It is important to understand the difference between a service offered by a layer, the interface by which that service is made available, and the protocol that a layer implements to establish communication.

من المهم فهم الفرق بين الخدمة التي تقدمها الطبقة، والواجهة التي يتم من خلالها إتاحة هذه الخدمة، والبروتوكول الذي تنفذه الطبقة لتأسيس الاتصال.

This distinction is shown in Figure 2.2.

يظهر هذا التمييز في الشكل 2.2.

To make this distinction clear, consider a reliable, connection-oriented

لتوضيح هذا التمييز، فكر في نظام موثوق وموجه نحو الاتصال

service, which is provided by many communication systems.

الخدمة التي تقدمها العديد من أنظمة الاتصالات.

In this case, a communicating party first needs to set up a connection to another party 7 before the two can send and receive messages Being reliable means that

في هذه الحالة، يحتاج الطرف المتصل أولاً إلى إعداد اتصال بطرف آخر 7 قبل أن يتمكن الطرفان من إرسال واستقبال الرسائل.

# Two-party communication

## Server

```
1 from socket  import *
2 s = socket(AF_INET, SOCK_STREAM)
3 (conn, addr) = s.accept()  # returns new socket and addr. client
4 while True:                # forever
5   data = conn.recv(1024)   # receive data from client
6   if not data: break       # stop if client stopped
7   conn.send(str(data)+"*") # return sent data plus an "*"
8 conn.close()               # close the connection
```

## Client

```
1 from socket  import *
2 s = socket(AF_INET, SOCK_STREAM)
3 s.connect((HOST, PORT)) # connect to server (block until accepted)
4 s.send('Hello, world')  # send some data
5 data = s.recv(1024)     # receive the response
6 print data               # print the result
7 s.close()               # close the connection
```

Architectures: Architectural styles

الهندسة المعمارية: الأساليب المعمارية

Layered architectures

أبنية الطبقات

Two-party communication

التواصل بين الطرفين

Server

الخادم

1 from socket import *

1 من استيراد المقبس *

2 s = socket(AF_INET, SOCK_STREAM)

2 ثانية = المقبس (AF_INET، SOCK_STREAM)

3 (conn, addr) = s.accept()

3 (كون، أدر) = s.accept()

# returns new socket and addr.

# إرجاع مأخذ توصيل جديد وaddr.

client

عميل

4 while True:

4 بينما صحيح:

# forever

# للأبد

data = conn.recv(1024)

البيانات = conn.recv(1024)

# receive data from client

#استقبال البيانات من العميل

if not data: break

إذا لم تكن البيانات: استراحة

# stop if client stopped

# توقف إذا توقف العميل

7 conn.send(str(data)+"*") # return sent data plus an "*"

7 (conn.send(str(data)+"*") # إرجاع البيانات المرسلة بالإضافة إلى "*"

8 conn.close() # close the connection

8 conn.Close() # أغلق الاتصال

Client

عميل

3 s.connect((HOST, PORT)) # connect to server (block until accepted)

3 s.connect((HOST, PORT)) # الاتصال بالخادم (حظر حتى يتم قبوله)

4 s.send(’Hello, world’)

4 s.send('مرحبا بالعالم')

# send some data

# إرسال بعض البيانات

5 data = s.recv(1024)

5 بيانات = s.recv(1024)

# receive the response

#استلام الرد

6 print data

6 طباعة البيانات

# print the result

#طباعة النتيجة

7 s.close()

7 ثواني.إغلاق()

# close the connection

#أغلق الاتصال

In this example, a server is created that makes use of a **connection-oriented service** as offered by the `socket` library available in Python. This service allows two communicating parties to reliably send and receive data over a connection. The main functions available in its interface are:

- `socket()`: to create an object representing the connection
- `accept()`: a blocking call to wait for incoming connection requests; if successful, the call returns a new socket for a separate connection
- `connect()`: to set up a connection to a specified party
- `close()`: to tear down a connection
- `send()`, `recv()`: to send and receive data over a connection, respectively

(b) A client

(ب) العميل

Figure 2.3: Two communicating parties.

الشكل 2.3: طرفان متصلان.

In this example, a server is created that makes use of a connection-oriented service as offered by the socket library available in Python.

في هذا المثال، يتم إنشاء خادم يستخدم خدمة موجهة للاتصال كما توفرها مكتبة المقبس المتوفرة في Python.

This service allows two communicating parties to reliably send and receive data over a connection.

تسمح هذه الخدمة لطرفين متصلين بإرسال واستقبال البيانات بشكل موثوق عبر الاتصال.

The main functions available in its interface are:

الوظائف الرئيسية المتوفرة في واجهته هي:

• socket(): to create an object representing the connection

• المقبس (): لإنشاء كائن يمثل الاتصال

• accept(): a blocking call to wait for incoming connection requests; if successful, the call returns a new socket for a separate connection

• قبول (): مكالمة حظر لانتظار طلبات الاتصال الواردة. إذا نجحت المكالمة بإرجاع مأخذ توصيل جديد لاتصال منفصل

• connect(): to set up a connection to a specified party

• الاتصال (): لإعداد اتصال بطرف محدد

• close(): to tear down a connection

• إغلاق (): لقطع الاتصال

• send(), recv(): to send and receive data over a connection, respectively

• send(), recv(): لإرسال واستقبال البيانات عبر الاتصال، على التوالي



DS 3.01pre

DS 3.01ما قبل

The combination of constants `AF_INET` and `SOCK_STREAM` is used to specify that the TCP protocol should be used in the communication between the two parties. These two constants can be seen as part of the interface, whereas making use of TCP is part of the offered service. *How* TCP is implemented, or for that matter any part of the communication service is hidden completely from the applications.

Finally, also note that these two programs implicitly adhere to an application-level protocol: apparently, if the client sends some data, the server will return it. Indeed, it operates as an echo server where the server adds an asterisk to the data sent by the client.

The combination of constants AF_INET and SOCK_STREAM is used to specify that the TCP protocol should be used in the communication between the two parties.

يتم استخدام مجموعة الثوابت AF_INET وSOCK_STREAM لتحديد ضرورة استخدام بروتوكول TCP في الاتصال بين الطرفين.

These two constants can be seen as part of the interface, whereas making use of TCP is part of the offered service.

يمكن رؤية هذين الثابتين كجزء من الواجهة، في حين أن الاستفادة من TCP جزء من الخدمة المقدمة.

How TCP is implemented, or for that matter any part of the communication service is hidden completely from the applications.

كيفية تنفيذ TCP، أو إخفاء أي جزء من خدمة الاتصال بالكامل من التطبيقات.

Finally, also note that these two programs implicitly adhere to an application-level protocol: apparently, if the client sends some data, the server will return it.

أخيرًا، لاحظ أيضًا أن هذين البرنامجين يلتزمان ضمنيًا ببروتوكول على مستوى التطبيق: على ما يبدو، إذا أرسل العميل بعض البيانات، فسيعيدها الخادم.

Indeed, it operates as an echo server where the server adds an asterisk to the data sent by the client.

في الواقع، فهو يعمل كخادم صدى حيث يضيف الخادم علامة النجمة إلى البيانات المرسلة من قبل العميل.

Application layering

طبقات التطبيق

Let us now turn our attention to the logical layering of applications.

دعونا الآن نحول انتباهنا إلى الطبقات المنطقية للتطبيقات.

Consider-

يعتبر-

ing that a large class of distributed applications is targeted toward supporting

مما يعني أن فئة كبيرة من التطبيقات الموزعة تستهدف الدعم

# Application Layering

## Traditional three-layered view

- Application-interface layer contains units for interfacing to users or external applications
- Processing layer contains the functions of an application, i.e., without specific data
- Data layer contains the data that a client wants to manipulate through the application components

## Observation

This layering is found in many distributed information systems, using traditional database technology and accompanying applications.

Architectures: Architectural styles

الهندسة المعمارية: الأساليب المعمارية

Layered architectures

أبنية الطبقات

Application Layering

طبقات التطبيق

Traditional three-layered view

منظر تقليدي ثلاثي الطبقات

Application-interface layer contains units for interfacing to users or external applications Processing layer contains the functions of an application, i.e., without specific data Data layer contains the data that a client wants to manipulate through the application components

تحتوي طبقة واجهة التطبيق على وحدات للتواصل مع المستخدمين أو التطبيقات الخارجية. تحتوي طبقة المعالجة على وظائف التطبيق، أي بدون بيانات محددة. تحتوي طبقة البيانات على البيانات التي يريد العميل معالجتها من خلال مكونات التطبيق

Observation

ملاحظة

This layering is found in many distributed information systems, using traditional database technology and accompanying applications.

توجد هذه الطبقات في العديد من أنظمة المعلومات الموزعة، باستخدام تقنية قواعد البيانات التقليدية والتطبيقات المصاحبة لها.

As a first example, consider an Internet search engine. Ignoring all the animated banners, images, and other fancy window dressing, the user interface of a search engine can be very simple: a user types in a string of keywords and is subsequently presented with a list of titles of Web pages. The back end is formed by a huge database of Web pages that have been prefetched and indexed. The core of the search engine is a program that transforms the user's string of keywords into one or more database queries. It subsequently ranks the results into a list, and transforms that list into a series of HTML pages. This information retrieval part is typically placed at the processing level. Figure 2.4 shows this organization.

to the processing level.

إلى مستوى المعالجة.

Therefore, we shall give a number of examples to make this level clearer.

ولذلك سنضرب عدداً من الأمثلة لتوضيح هذا المستوى.

As a first example, consider an Internet search engine.

كمثال أول، فكر في محرك بحث على الإنترنت.

Ignoring all the ani-mated banners, images, and other fancy window dressing, the user interface of a search engine can be very simple: a user types in a string of keywords and is subsequently presented with a list of titles of Web pages.

بتجاهل كل اللافتات والصور المتحركة وغيرها من أدوات تزيين النوافذ الفاخرة، يمكن أن تكون واجهة المستخدم الخاصة بمحرك البحث بسيطة للغاية: يقوم المستخدم بكتابة سلسلة من الكلمات الرئيسية ويتم تقديمه لاحقًا بقائمة عناوين صفحات الويب.

The back end is formed by a huge database of Web pages that have been prefetched and indexed.

تتكون الواجهة الخلفية من قاعدة بيانات ضخمة من صفحات الويب التي تم جلبها مسبقًا وفهرستها.

The core of the search engine is a program that transforms the user’s string of keywords into one or more database queries.

جوهر محرك البحث هو برنامج يحول سلسلة الكلمات الرئيسية للمستخدم إلى استعلام قاعدة بيانات واحد أو أكثر.

It subsequently ranks the results into a list, and transforms that list into a series of HTML pages.

ويقوم بعد ذلك بترتيب النتائج في قائمة، ويحول تلك القائمة إلى سلسلة من صفحات HTML.

This information retrieval part is typically placed at the processing level.

عادةً ما يتم وضع جزء استرجاع المعلومات هذا على مستوى المعالجة.

Figure 2.4 shows this organization.

ويبين الشكل 2.4 هذه المنظمة.

As a second example, consider a decision support system for stock bro-kerage.

وكمثال ثان، فكر في نظام دعم القرار لوساطة الأوراق المالية.

Analogous to a search engine, such a system can be divided into the

مشابهًا لمحرك البحث، يمكن تقسيم هذا النظام إلى

following three layers:

الطبقات الثلاث التالية:



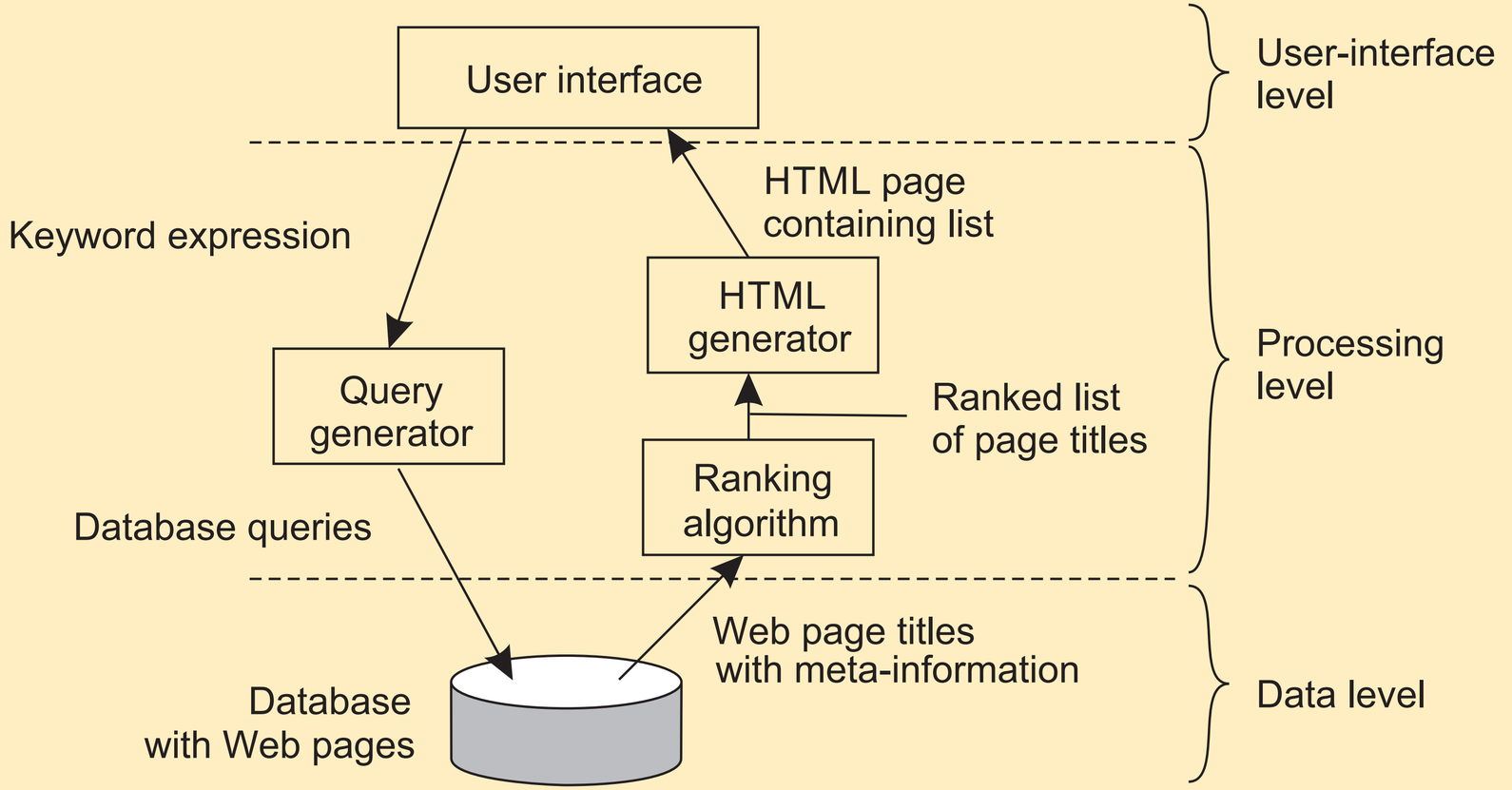


DS 3.01pre

DS 3.01ما قبل

# Application Layering

## Example: a simple search engine



Architectures: Architectural styles

الهندسة المعمارية: الأساليب المعمارية

Layered architectures

أبنية الطبقات

Application Layering

طبقات التطبيق

Example: a simple search engine

مثال: محرك بحث بسيط

User interface

واجهة المستخدم

User-interface level

مستوى واجهة المستخدم

Keyword expression

تعبير الكلمات الرئيسية

HTML page containing list

صفحة HTML تحتوي على قائمة

Query

استفسار

HTML

لغة البرمجة

Ranked list

قائمة مرتبة

Processing

يعالج

generator

مولد كهرباء

level

مستوى

of page titles

من عناوين الصفحات

Database queries

استعلامات قاعدة البيانات

Ranking algorithm

خوارزمية الترتيب

Web page titles

عناوين صفحات الويب

Database

قاعدة البيانات

with meta-information

مع المعلومات الفوقية

Data level

## مستوى البيانات

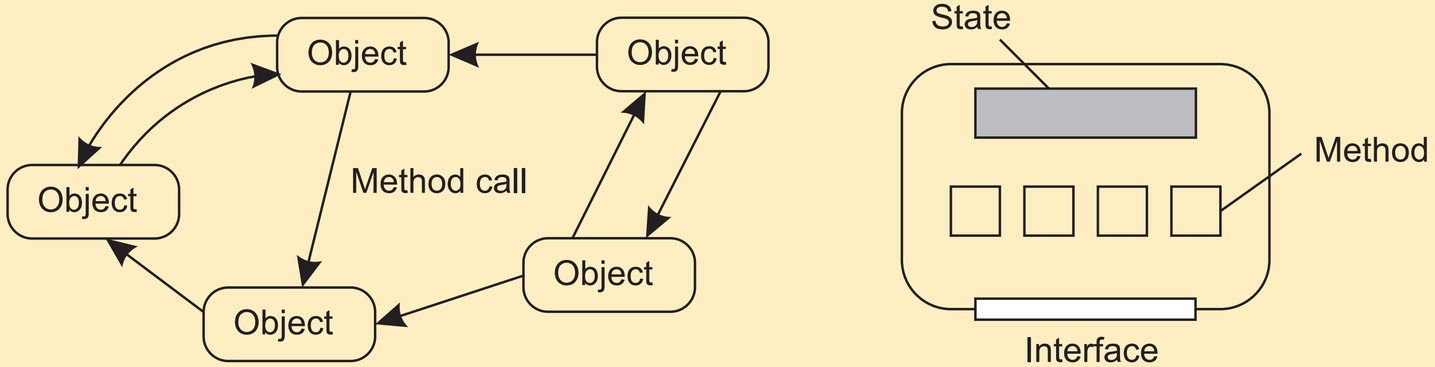
## with Web pages

## مع صفحات الويب

# Object-based style

## Essence

Components are objects, connected to each other through procedure calls. Objects may be placed on different machines; calls can thus execute across a network.







## Encapsulation

Objects are said to encapsulate data and offer methods on that data without revealing the internal implementation.

Architectures: Architectural styles

الهندسة المعمارية: الأساليب المعمارية

Object-based and service-oriented architectures

البنى القائمة على الكائنات والموجهة نحو الخدمة

Object-based style

النمط القائم على الكائنات

Essence

جوهر

Components are objects, connected to each other through procedure calls.

المكونات هي كائنات، متصلة ببعضها البعض من خلال استدعاءات الإجراءات.

Objects may be placed on different machines; calls can thus execute across a network.

يمكن وضع الأشياء على أجهزة مختلفة؛ وبالتالي يمكن تنفيذ المكالمات عبر الشبكة.

State Object Object

كائن كائن الحالة

Object

هدف

Method call

استدعاء الطريقة

Method

طريقة

Interface

واجهه المستخدم

Encapsulation

التغليف

Objects are said to encapsulate data and offer methods on that data without revealing the internal implementation.

يقال إن الكائنات تقوم بتغليف البيانات وتقديم طرق على تلك البيانات دون الكشف عن التنفيذ الداخلي.

# Resource Based Architecture

## Representational State Transfer (REST)

Resource Based Architecture

العمارة القائمة على الموارد

Representational State Transfer (REST)

نقل الحالة التمثيلية (REST)

# RESTful architectures

## Essence

View a distributed system as a collection of resources, individually managed by components. Resources may be added, removed, retrieved, and modified by (remote) applications.

1. Resources are identified through a single naming scheme
2. All services offer the same interface
3. Messages sent to or from a service are fully self-described
4. After executing an operation at a service, that component forgets everything about the caller

## Basic operations

| Operation | Description |
|---|---|
| `PUT` | Create a new resource |
| `GET` | Retrieve the state of a resource in some representation |
| `DELETE` | Delete a resource |
| `POST` | Modify a resource by transferring a new state |

Architectures: Architectural styles

الهندسة المعمارية: الأساليب المعمارية

Resource-based architectures

البنى القائمة على الموارد

RESTful architectures

أبنية مريحة

Essence

جوهر

View a distributed system as a collection of resources, individually managed by components.

عرض النظام الموزع كمجموعة من الموارد، تتم إدارتها بشكل فردي بواسطة المكونات.

Resources may be added, removed, retrieved, and modified by (remote) applications.

يمكن إضافة الموارد وإزالتها واسترجاعها وتعديلها بواسطة التطبيقات (البعيدة).

Resources are identified through a single naming scheme

يتم تحديد الموارد من خلال نظام تسمية واحد

All services offer the same interface

جميع الخدمات تقدم نفس الواجهة

Messages sent to or from a service are fully self-described

يتم وصف الرسائل المرسلة إلى الخدمة أو منها بشكل كامل

After executing an operation at a service, that component forgets

بعد تنفيذ عملية في الخدمة، يتم نسيان هذا المكون

everything about the caller

كل شيء عن المتصل

Basic operations

العمليات الأساسية

Operation

عملية

Description

وصف

PUT

يضع

Create a new resource

إنشاء مورد جديد

GET

يحصل

Retrieve the state of a resource in some representation

استرداد حالة المورد في بعض التمثيل

DELETE

يمسح

Delete a resource

حذف مورد

POST

بريد

Modify a resource by transferring a new state

تعديل مورد عن طريق نقل حالة جديدة

# Example: Amazon's Simple Storage Service

## Essence

Objects (i.e., files) are placed into buckets (i.e., directories). Buckets cannot be placed into buckets. Operations on `ObjectName` in bucket `BucketName` require the following identifier:

```
http://BucketName.s3.amazonaws.com/ObjectName
```

## Typical operations

All operations are carried out by sending HTTP requests:

- Create a bucket/object: `PUT`, along with the URI
- Listing objects: `GET` on a bucket name
- Reading an object: `GET` on a full URI

Architectures: Architectural styles

الهندسة المعمارية: الأساليب المعمارية

Resource-based architectures

البنى القائمة على الموارد

Example: Amazon’s Simple Storage Service

مثال: خدمة التخزين البسيطة من أمازون

Essence

جوهر

Objects (i.e., files) are placed into buckets (i.e., directories).

يتم وضع الكائنات (أي الملفات) في مجموعات (أي الدلائل).

Buckets cannot be placed into buckets.

لا يمكن وضع الدلاء في الدلاء.

Operations on ObjectName in bucket BucketName require the following identifier:

تتطلب العمليات على ObjectName في الحاوية BucketName المعرف التالي:

Typical operations

## العمليات النموذجية

All operations are carried out by sending HTTP requests:

يتم تنفيذ جميع العمليات عن طريق إرسال طلبات HTTP:

Create a bucket/object: PUT, along with the URI Listing objects: GET on a bucket name
Reading an object: GET on a full URI

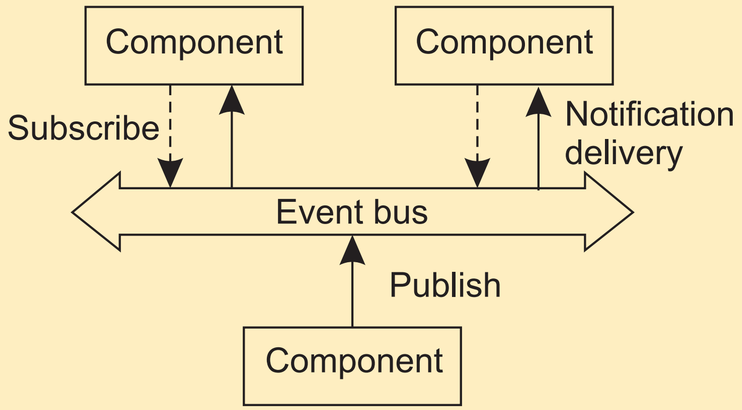
إنشاء مجموعة/كائن: PUT، جنبًا إلى جنب مع URI إدراج الكائنات: الحصول على اسم المجموعة قراءة كائن: الحصول على URI كامل

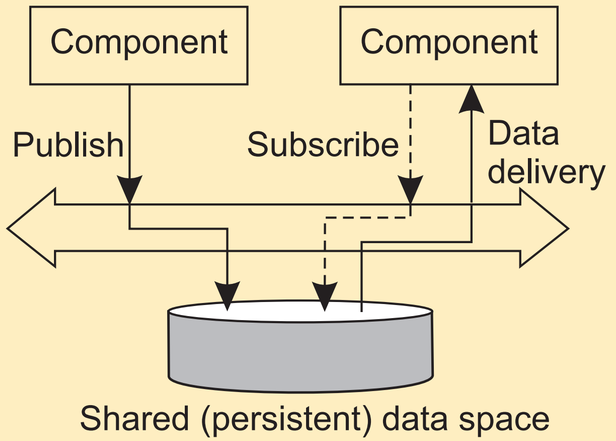
# Coordination

## Temporal and referential coupling

| | **Temporally coupled** | **Temporally decoupled** |
|---|---|---|
| **Referentially coupled** | Direct | Mailbox |
| **Referentially decoupled** | Event-based | Shared data space |

## Event-based and Shared data space



Architectures: Architectural styles

الهندسة المعمارية: الأساليب المعمارية

Publish-subscribe architectures

بنيات النشر والاشتراك

Coordination

تنسيق

Temporal and referential coupling

اقتران زمني ومرجعي

Temporally coupled

مقترنة مؤقتا

Temporally decoupled

مفصولة مؤقتا

Referentially coupled

مقترنة مرجعيا

Direct

مباشر

Mailbox

صندوق بريد

Referentially decoupled

مفصولة بشكل مرجعي

Event-

حدث-

based

قائم على

Shared data space

مساحة البيانات المشتركة

Event-based and Shared data space

مساحة البيانات المستندة إلى الأحداث والمشتركة

Component

عنصر

Subscribe

يشترك

Notification

إشعار

Publish

ينشر

Data

بيانات

delivery

توصيل

Event bus

حافلة الحدث

Shared (persistent) data space

مساحة البيانات المشتركة (الدائمة).

- Shared data spaces are often combined with event-based coordination: a process subscribes to certain tuples by providing a search pattern; when a process inserts a tuple into the data space, matching subscribers are notified. In both cases, we are dealing with a publish-subscribe architecture, and indeed, the key characteristic feature is that processes have no explicit reference to each other. The difference between a pure event-based architectural style , and that of a shared data space, is shown in the figure. An abstraction of the mechanism by which publishers and subscribers are matched, known as an event bus have also been shown.

• Shared data spaces are often combined with event-based coordination: a process subscribes to certain tuples by providing a search pattern; when a process inserts a tuple into the data space, matching subscribers are notified.

• غالبًا ما يتم دمج مساحات البيانات المشتركة مع التنسيق القائم على الحدث: حيث تشترك العملية في صفوف معينة من خلال توفير نمط بحث؛ عندما تقوم إحدى العمليات بإدراج صف في مساحة البيانات، يتم إخطار المشتركين المطابقين.

In both cases, we are dealing with a publish-subscribe architecture, and indeed, the key characteristic feature is that processes have no explicit reference to each other.

في كلتا الحالتين، نحن نتعامل مع بنية النشر والاشتراك، وفي الواقع، السمة المميزة الرئيسية هي أن العمليات ليس لها إشارة واضحة لبعضها البعض.

The difference between a pure event-based architectural style , and that of a shared data space, is shown in the figure.

يظهر في الشكل الفرق بين النمط المعماري القائم على الحدث ونمط مساحة البيانات المشتركة.

An abstraction of the mechanism by which publishers and subscribers are matched, known as an event bus have also been shown.

تم أيضًا عرض ملخص للآلية التي يتم من خلالها مطابقة الناشرين والمشتركين، والمعروفة باسم ناقل الأحداث.

# Developing adaptable middleware

## Problem

Middleware contains solutions that are good for most applications $\Rightarrow$ you may want to adapt its behavior for specific applications.

Architectures: Middleware organization

البنى: تنظيم الوسيطة

Interceptors

اعتراضية

Developing adaptable middleware

تطوير البرمجيات الوسيطة القابلة للتكيف

Problem

مشكلة

want to adapt its behavior for specific applications.

تريد تكييف سلوكها لتطبيقات محددة.

Middleware contains solutions that are good for most applications ) you may

تحتوي البرامج الوسيطة على حلول مناسبة لمعظم التطبيقات).

- AN interceptor

  - Conceptually, an interceptor is nothing but a software construct that will break the usual flow of control and allow other (application specific) code to be executed. Interceptors are a primary means for adapting middleware to the specific needs of an application.

a remote-object invocation is carried out in three steps:

1. Object A is offered a local interface that is the same as the interface offered by object B. A calls the method available in that interface.

2. The call by A is transformed into a generic object invocation, made possible through a general object-invocation interface offered by the middleware at the machine where A resides.

3. Finally, the generic object invocation is transformed into a message that is sent through the transport-level network interface as offered by A's local operating system.

• AN interceptor

• اعتراضية

• Conceptually, an interceptor is nothing but a software construct that will break the usual flow of control and allow other (application specific) code to be executed.

• من الناحية النظرية، فإن المعترض ليس سوى بناء برمجي من شأنه أن يكسر التدفق المعتاد للتحكم ويسمح بتنفيذ تعليمات برمجية أخرى (خاصة بالتطبيق).

Interceptors are a primary means for adapting middleware to the specific needs of an application.

تعد أدوات الاعتراض وسيلة أساسية لتكييف البرامج الوسيطة مع الاحتياجات المحددة للتطبيق.

a remote-object invocation is carried out in three steps:

يتم تنفيذ استدعاء الكائن البعيد في ثلاث خطوات:

Object A is offered a local interface that is the same as the interface offered by object B.

يتم تقديم واجهة محلية للكائن A مماثلة للواجهة التي يقدمها الكائن B.

A calls the method available in that interface.

يستدعي الطريقة المتاحة في تلك الواجهة.

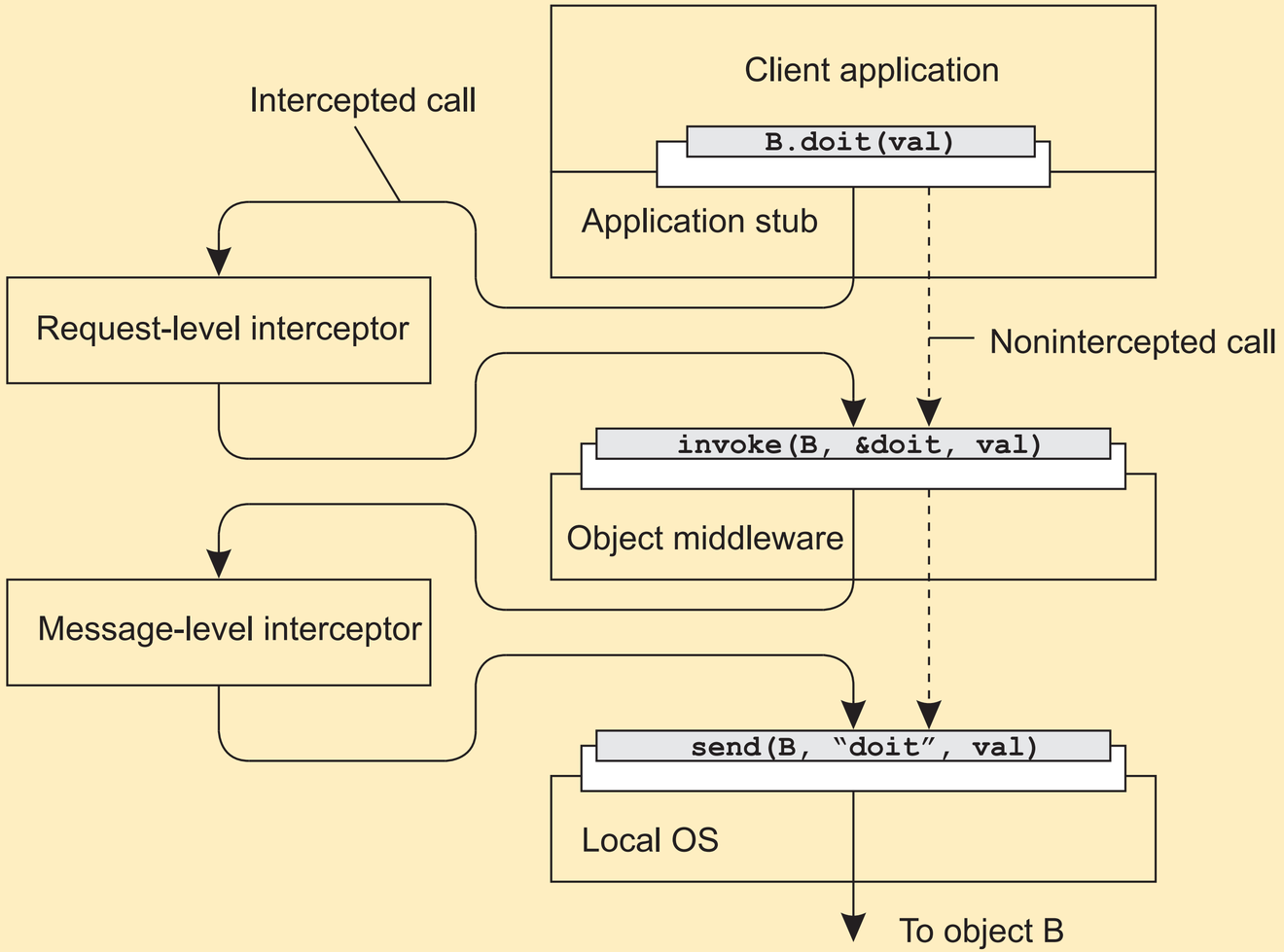
The call by A is transformed into a generic object invocation, made possible through a general object-invocation interface offered by the middleware at the machine where A resides.

يتم تحويل استدعاء الكائن A إلى استدعاء كائن عام، وهو ما أصبح ممكنًا من خلال واجهة استدعاء الكائن العامة التي توفرها البرامج الوسيطة على الجهاز الذي يوجد به A.

Finally, the generic object invocation is transformed into a message that is sent through the transport-level network interface as offered by A's local operating system.

وأخيرًا، يتم تحويل استدعاء الكائن العام إلى رسالة يتم إرسالها عبر واجهة الشبكة على مستوى النقل كما يقدمها نظام التشغيل المحلي الخاص بـ A.

# Intercept the usual flow of control



Architectures: Middleware organization

البنى: تنظيم الوسيطة

Interceptors

اعتراضية

Intercept the usual flow of control

اعتراض التدفق المعتاد للسيطرة

Intercepted call

تم اعتراض المكالمة

Client application

تطبيق العميل

B.doit(val)

ب.دويت (فال)

Request-level interceptor

اعتراض على مستوى الطلب

Application stub

كعب التطبيق

Nonintercepted call

مكالمة غير اعتراضية

invoke(B, &doit, val)

استدعاء (B، &doit، val)

Object middleware

الوسيطة للكائنات

Message-level interceptor

اعتراضية على مستوى الرسالة

send(B, “doit”, val)

إرسال (ب، "doit"، فال)

Local OS

نظام التشغيل المحلي

To object B

للاعتراض ب

- This scheme is shown in the Figure. After the first step, the call B.doit(val) is transformed into a generic call, such as invoke(B,&doit,val) with a reference to B's method and the parameters that go along with the call. Now imagine that object B is replicated. In that case, each replica should actually be invoked. This is a clear point where interception can help. What the request-level interceptor will do, is simply call invoke(B,&doit,val) for each of the replicas. The beauty of this all is that the object A need not be aware of the replication of B, but also the object middleware need not have special components that deal with this replicated call. Only the request-level interceptor, which may be added to the middleware, needs to know about B's replication. In the end, a call to a remote object will have to be sent over the network.

• This scheme is shown in the Figure.

• يظهر هذا المخطط في الشكل.

After the first step, the call B.doit(val) is transformed into a generic call, such as invoke(B,&doit,val) with a reference to B’s method and the parameters that go along with the call.

بعد الخطوة الأولى، يتم تحويل الاستدعاء B.doit(val) إلى استدعاء عام، مثل invoc(B,&doit,val) مع الإشارة إلى أسلوب B والمعلمات المصاحبة للاستدعاء.

Now imagine that object B is replicated.

الآن تخيل أن الكائن B قد تم نسخه.

In that case, each replica should actually be invoked.

في هذه الحالة، يجب بالفعل استدعاء كل نسخة متماثلة.

This is a clear point where interception can help.

وهذه نقطة واضحة يمكن أن يساعد فيها الاعتراض.

What the request-level interceptor will do, is simply call invoke(B,&doit,val) for each of the replicas.

ما سيفعله المعترض على مستوى الطلب هو ببساطة استدعاء استدعاء (B,&doit,val) لكل نسخة من النسخ المتماثلة.

The beauty of this all is that the object A need not be aware of the replication of B, but also the object middleware need not have special components that deal with this replicated call.

الجميل في هذا كله هو أن الكائن A لا يحتاج إلى أن يكون على دراية بتكرار الكائن B، ولكن أيضًا لا تحتاج البرامج الوسيطة للكائن إلى مكونات خاصة تتعامل مع هذا الاستدعاء المكرر.

Only the request-level interceptor, which may be added to the middleware, needs to know about B’s replication.

فقط المعترض على مستوى الطلب، والذي يمكن إضافته إلى البرنامج الوسيط، هو الذي يحتاج إلى معرفة النسخ المتماثل لـ B.

In the end, a call to a remote object will have to be sent over the network.

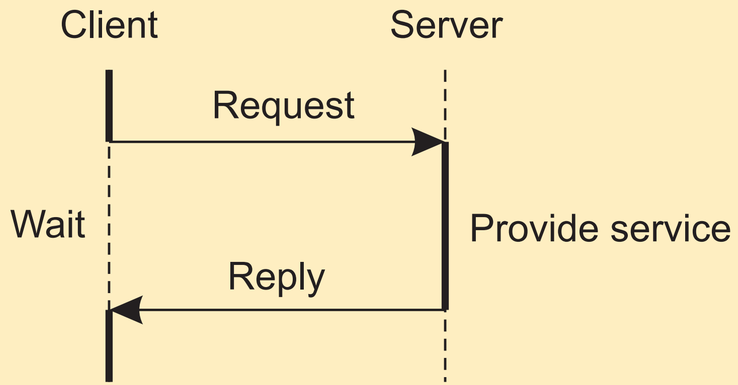
في النهاية، يجب إرسال مكالمة إلى كائن بعيد عبر الشبكة.

# Centralized system architectures

## Basic Client–Server Model

Characteristics:

- There are processes offering services (servers)
- There are processes that use services (clients)
- Clients and servers can be on different machines
- Clients follow request/reply model with respect to using services



Architectures: System architecture

البنى: بنية النظام

Centralized organizations

المنظمات المركزية

Centralized system architectures

بنيات النظام المركزي

Basic Client–Server Model

نموذج العميل-الخادم الأساسي

Characteristics:

صفات:

There are processes offering services (servers) There are processes that use services (clients) Clients and servers can be on different machines Clients follow request/reply model with respect to using services

هناك عمليات تقدم خدمات (خوادم) هناك عمليات تستخدم خدمات (عملاء) يمكن أن يكون العملاء والخوادم على أجهزة مختلفة يتبع العملاء نموذج الطلب/الرد فيما يتعلق باستخدام الخدمات

Client

عميل

Server

الخادم

Wait

انتظر

Request

طلب

Provide service

تقديم الخدمة

Reply

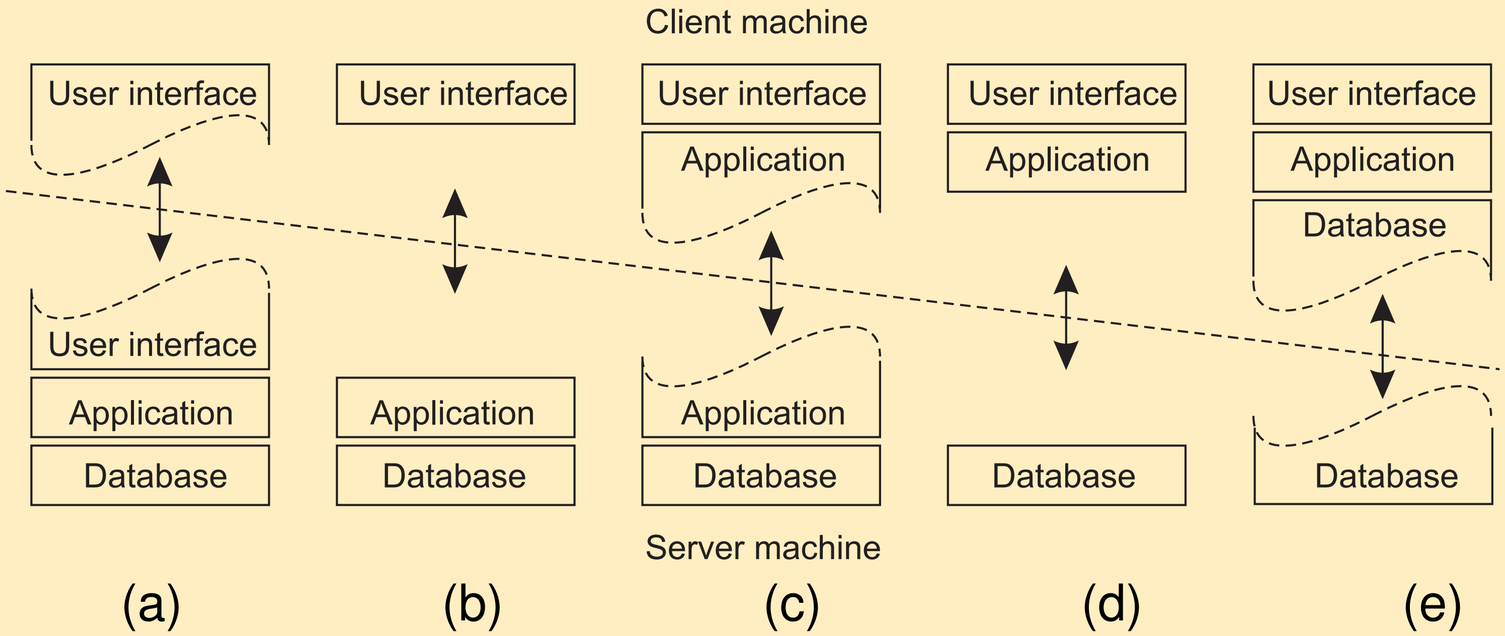
رد

# Multi-tiered centralized system architectures

## Some traditional organizations

- Single-tiered: dumb terminal/mainframe configuration
- Two-tiered: client/single server configuration
- Three-tiered: each layer on separate machine

## Traditional two-tiered configurations



Architectures: System architecture

البنى: بنية النظام

Centralized organizations

المنظمات المركزية

Multi-tiered centralized system architectures

بنيات النظام المركزي متعدد المستويات

Some traditional organizations

بعض المنظمات التقليدية

Single-tiered: dumb terminal/mainframe configuration Two-tiered: client/single server configuration Three-tiered: each layer on separate machine

طبقة واحدة: تكوين المحطة الطرفية/الحاسب المركزي الغبي مستويين: تكوين العميل/الخادم الفردي ثلاث طبقات: كل طبقة على جهاز منفصل

Traditional two-tiered configurations

التكوينات التقليدية ذات المستويين

User interface

واجهة المستخدم

Client machine

آلة العميل

Application

طلب

Database

قاعدة البيانات

Server machine

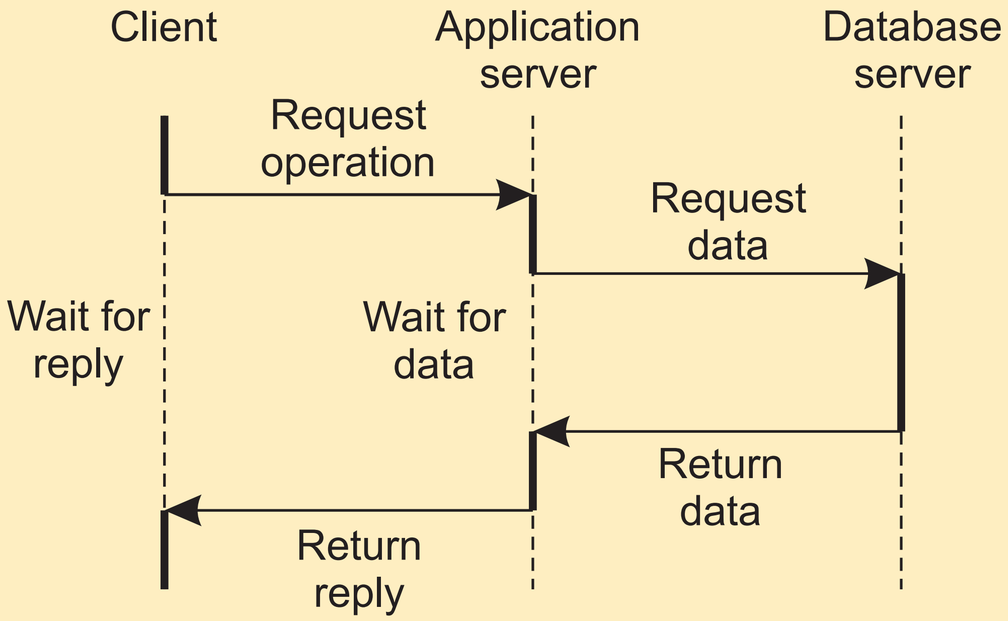
آلة الخادم

Multitiered Architectures

معماريات متعددة المستويات

# Being client and server at the same time

## Three-tiered architecture



Architectures: System architecture

البنى: بنية النظام

Centralized organizations

المنظمات المركزية

Being client and server at the same time

كونه العميل و الخادم في نفس الوقت

Three-tiered architecture

العمارة ثلاثية الطبقات

Client

عميل

Application

طلب

Request

طلب

Database

قاعدة البيانات

server

الخادم

operation

عملية

Wait for

أنتظر لأجل

data

بيانات

reply

رد

Return

يعود

Return data

إرجاع البيانات

# Alternative organizations

## Vertical distribution

Comes from dividing distributed applications into three logical layers, and running the components from each layer on a different server (machine).

## Horizontal distribution

A client or server may be physically split up into logically equivalent parts, but each part is operating on its own share of the complete data set.

## Peer-to-peer architectures

Processes are all equal: the functions that need to be carried out are represented by every process $\Rightarrow$ each process will act as a client and a server at the same time (i.e., acting as a servant).

Architectures: System architecture

البنى: بنية النظام

Decentralized organizations: peer-to-peer systems

المنظمات اللامركزية: أنظمة الند للند

Alternative organizations

المنظمات البديلة

Vertical distribution

التوزيع العمودي

Comes from dividing distributed applications into three logical layers, and running the components from each layer on a different server (machine).

يأتي من تقسيم التطبيقات الموزعة إلى ثلاث طبقات منطقية، وتشغيل المكونات من كل طبقة على خادم (جهاز) مختلف.

Horizontal distribution

التوزيع الأفقي

A client or server may be physically split up into logically equivalent parts, but each part is operating on its own share of the complete data set.

قد يتم تقسيم العميل أو الخادم فعليًا إلى أجزاء متكافئة منطقيًا، ولكن كل جزء يعمل على حصته الخاصة من مجموعة البيانات الكاملة.

Peer-to-peer architectures

معماريات نظير إلى نظير

Processes are all equal: the functions that need to be carried out are represented by every process ) each process will act as a client and a server at the same time (i.e., acting as a servant).

جميع العمليات متساوية: الوظائف التي يجب تنفيذها يتم تمثيلها في كل عملية. ستعمل كل عملية كعميل وخادم في نفس الوقت (أي تعمل كخادم).

# Decentralized Architectures

**Observation**

In the last couple of years we have been seeing a tremendous growth in peer-to-peer systems.

- Structured P2P: nodes are organized following a specific distributed data structure
- Unstructured P2P: nodes have randomly selected neighbors
- Hybrid P2P: some nodes are appointed special functions in a well-organized fashion

**Note**

In virtually all cases, we are dealing with overlay networks: data is routed over connections setup between the nodes (cf. application-level multicasting)

Architectures

أبنية

2.2 System Architectures

2.2 بنيات النظام

Decentralized Architectures

البنى اللامركزية

Observation

ملاحظة

In the last couple of years we have been seeing a tremendous growth

في العامين الماضيين شهدنا نموا هائلا

in peer-to-peer systems.

في أنظمة نظير إلى نظير.

Structured P2P: nodes are organized following a specific

P2P منظم: يتم تنظيم العقد وفقًا لخطة محددة

distributed data structure

بنية البيانات الموزعة

Unstructured P2P: nodes have randomly selected neighbors

P2P غير منظم: العقد لديها جيران تم اختيارهم بشكل عشوائي

Hybrid P2P: some nodes are appointed special functions in a

Hybrid P2P: يتم تعيين بعض العقد بوظائف خاصة في أ

well-organized fashion

أزياء منظمة تنظيما جيدا

Note

ملحوظة

In virtually all cases, we are dealing with overlay networks: data is

في جميع الحالات تقريبًا، نحن نتعامل مع شبكات التراكب: البيانات هي

routed over connections setup between the nodes (cf.

توجيهها عبر إعداد الاتصالات بين العقد (راجع

application-level

مستوى التطبيق

multicasting)

البث المتعدد)

# Structured P2P

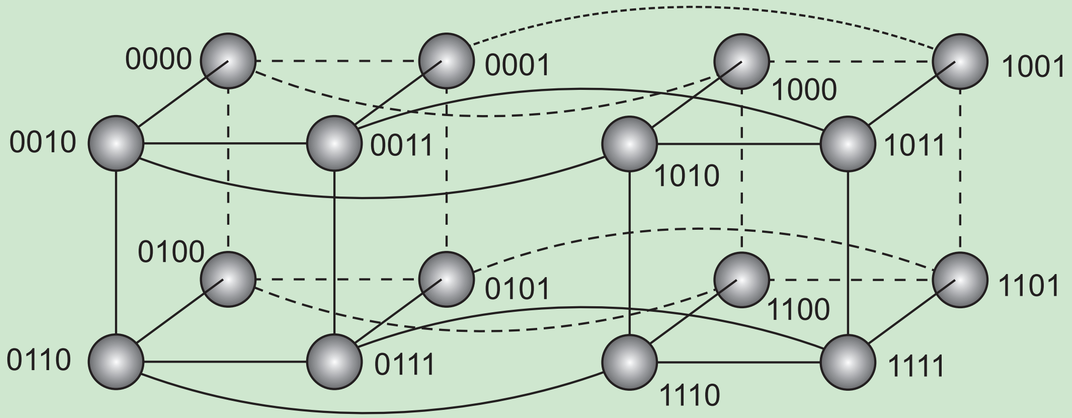
## Essence

Make use of a semantic-free index: each data item is uniquely associated with a key, in turn used as an index. Common practice: use a hash function

$$key(data\ item) = hash(data\ item's\ value).$$

P2P system now responsible for storing (*key*,*value*) pairs.

## Simple example: hypercube




Looking up $d$ with key $k \in \{0,1,2,\ldots,2^4-1\}$ means routing request to node with identifier $k$.

Architectures: System architecture

البنى: بنية النظام

Decentralized organizations: peer-to-peer systems

المنظمات اللامركزية: أنظمة الند للند

Structured P2P

P2P منظم

Essence

جوهر

Make use of a semantic-free index: each data item is uniquely associated with a key, in turn used as an index.

الاستفادة من الفهرس الخالي من الدلالات: يرتبط كل عنصر بيانات بشكل فريد بمفتاح، والذي يستخدم بدوره كفهرس.

Common practice: use a hash function

ممارسة شائعة: استخدم دالة التجزئة

key(data item) = hash(data item’s value).

المفتاح (عنصر البيانات) = التجزئة (قيمة عنصر البيانات).

P2P system now responsible for storing (key,value) pairs.

أصبح نظام P2P مسؤولاً الآن عن تخزين أزواج (المفتاح، القيمة).

Simple example: hypercube

مثال بسيط: المكعب الزائد

with identifier k. Looking up d with key k 2 {0,1,2,...,24−1} means routing request to node

مع المعرف ك. البحث عن d باستخدام المفتاح {0,1,2,...,24−1} k 2 يعني توجيه الطلب إلى العقدة

Structured peer-to-peer systems

أنظمة نظير إلى نظير منظمة

# Unstructured P2P

## Essence

Each node maintains an ad hoc list of neighbors. The resulting overlay resembles a random graph: an edge $\langle u, v \rangle$ exists only with a certain probability $\mathbb{P}[\langle u, v \rangle]$.

## Searching

- Flooding: issuing node $u$ passes request for $d$ to all neighbors. Request is ignored when receiving node had seen it before. Otherwise, $v$ searches locally for $d$ (recursively). May be limited by a Time-To-Live: a maximum number of hops.
- Random walk: issuing node $u$ passes request for $d$ to randomly chosen neighbor, $v$. If $v$ does not have $d$, it forwards request to one of *its* randomly chosen neighbors, and so on.

Architectures: System architecture

البنى: بنية النظام

Decentralized organizations: peer-to-peer systems

المنظمات اللامركزية: أنظمة الند للند

Unstructured P2P

P2P غير منظم

Essence

جوهر

Each node maintains an ad hoc list of neighbors.

تحتفظ كل عقدة بقائمة مخصصة من الجيران.

The resulting overlay resembles a random graph: an edge hu,vi exists only with a certain probability P[hu,vi].

يشبه التراكب الناتج رسمًا بيانيًا عشوائيًا: الحافة hu,vi موجودة فقط مع احتمال معين P[hu,vi].

Searching

يبحث

Flooding: issuing node u passes request for d to all neighbors.

الفيضان: إصدار عقدة u يمرر طلب d إلى جميع الجيران.

Request is ignored when receiving node had seen it before.

يتم تجاهل الطلب عندما تكون العقدة المتلقية قد شاهدته من قبل.

Otherwise, v searches locally for d (recursively).

بخلاف ذلك، يبحث v محليًا عن d (بشكل متكرر).

May be limited by a Time-To-Live: a maximum number of hops.

قد تكون محدودة بمدة البقاء: الحد الأقصى لعدد القفزات.

Random walk: issuing node u passes request for d to randomly chosen neighbor, v. If v does not have d, it forwards request to one of its randomly chosen neighbors, and so on.

المشي العشوائي: إصدار العقدة u يمرر طلب d إلى جار تم اختياره عشوائيًا، v. إذا لم يكن v يحتوي على d، فإنه يعيد توجيه الطلب إلى أحد جيرانه الذين تم اختيارهم عشوائيًا، وهكذا.

# Unstructured peer-to-peer systems

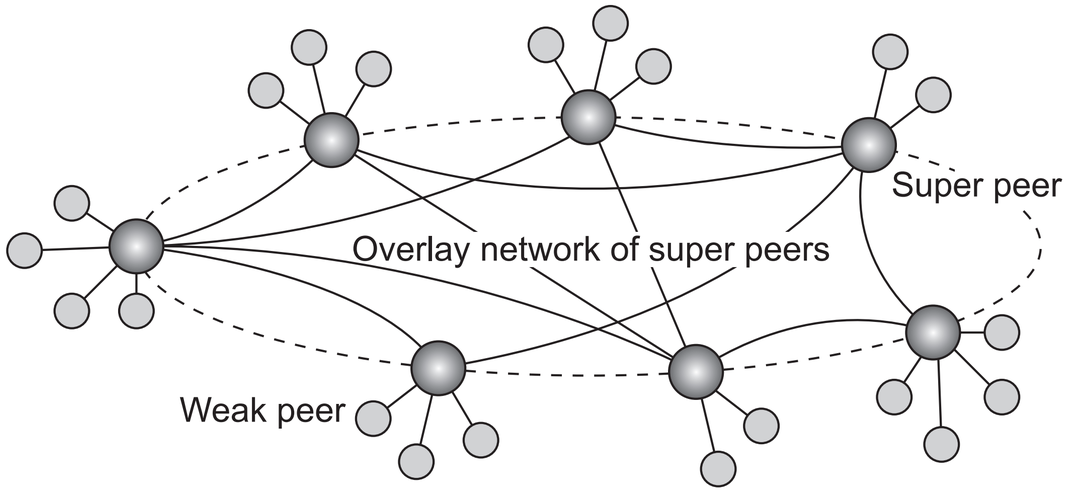
## أنظمة نظير إلى نظير غير منظمة

# Super-peer networks

## Essence

It is sometimes sensible to break the symmetry in pure peer-to-peer networks:

- When searching in unstructured P2P systems, having index servers improves performance
- Deciding where to store data can often be done more efficiently through brokers.



Architectures: System architecture

البنى: بنية النظام

Decentralized organizations: peer-to-peer systems

المنظمات اللامركزية: أنظمة الند للند

Super-peer networks

شبكات سوبر نظير

Essence

جوهر

It is sometimes sensible to break the symmetry in pure peer-to-peer networks:

من المنطقي في بعض الأحيان كسر التماثل في شبكات نظير إلى نظير الخالصة:

When searching in unstructured P2P systems, having index servers improves performance
Deciding where to store data can often be done more efficiently through brokers.

عند البحث في أنظمة P2P غير المنظمة، فإن وجود خوادم الفهرس يعمل على تحسين الأداء. غالبًا ما يمكن تحديد مكان تخزين البيانات بشكل أكثر كفاءة من خلال الوسطاء.

Super peer

سوبر نظير

Overlay network of super peers

تراكب شبكة من أقرانهم الفائقة

Weak peer

نظير ضعيف

Hierarchically organized peer-to-peer networks

شبكات نظير إلى نظير منظمة بشكل هرمي

# Skype's principle operation: *A* wants to contact *B*

## Both *A* and *B* are on the public Internet

- A TCP connection is set up between *A* and *B* for control packets.
- The actual call takes place using UDP packets between negotiated ports.

## *A* operates behind a firewall, while *B* is on the public Internet

- *A* sets up a TCP connection (for control packets) to a super peer *S*
- *S* sets up a TCP connection (for relaying control packets) to *B*
- The actual call takes place through UDP and directly between *A* and *B*

## Both *A* and *B* operate behind a firewall

- *A* connects to an online super peer *S* through TCP
- *S* sets up TCP connection to *B*.
- For the actual call, another super peer is contacted to act as a relay *R*: *A* sets up a connection to *R*, and so will *B*.
- All voice traffic is forwarded over the two TCP connections, and through *R*.

Architectures: System architecture

البنى: بنية النظام

Decentralized organizations: peer-to-peer systems

المنظمات اللامركزية: أنظمة الند للند

Skype’s principle operation: A wants to contact B

عملية Skype الأساسية: A يريد الاتصال بـ B

Both A and B are on the public Internet

كل من A و B موجودان على الإنترنت العام

A TCP connection is set up between A and B for control packets.

يتم إعداد اتصال TCP بين A و B لحزم التحكم.

The actual call takes place using UDP packets between negotiated ports.

ويتم الاستدعاء الفعلي باستخدام حزم UDP بين المنافذ التي تم التفاوض عليها.

A operates behind a firewall, while B is on the public Internet

يعمل A خلف جدار حماية، بينما يكون B موجودًا على الإنترنت العام

A sets up a TCP connection (for control packets) to a super peer S S sets up a TCP connection (for relaying control packets) to B The actual call takes place through UDP and directly between A and B

يقوم A بإعداد اتصال TCP (لحزم التحكم) إلى نظير فائق S يقوم بإعداد اتصال TCP (لترحيل حزم التحكم) إلى B ويتم إجراء المكالمة الفعلية من خلال UDP ومباشرة بين A و B

Both A and B operate behind a firewall

يعمل كل من A و B خلف جدار الحماية

A connects to an online super peer S through TCP S sets up TCP connection to B.

يتصل A بنظير فائق عبر الإنترنت S من خلال TCP S، ويقوم بإعداد اتصال TCP بـ B.

For the actual call, another super peer is contacted to act as a relay R: A sets up a connection to R, and so will B.

بالنسبة للمكالمة الفعلية، يتم الاتصال بنظير متميز آخر ليكون بمثابة مرحل R: يقوم A بإعداد اتصال بـ R، وكذلك يفعل B.

All voice traffic is forwarded over the two TCP connections, and through R.

تتم إعادة توجيه كل حركة الصوت عبر اتصالي TCP، ومن خلال R.

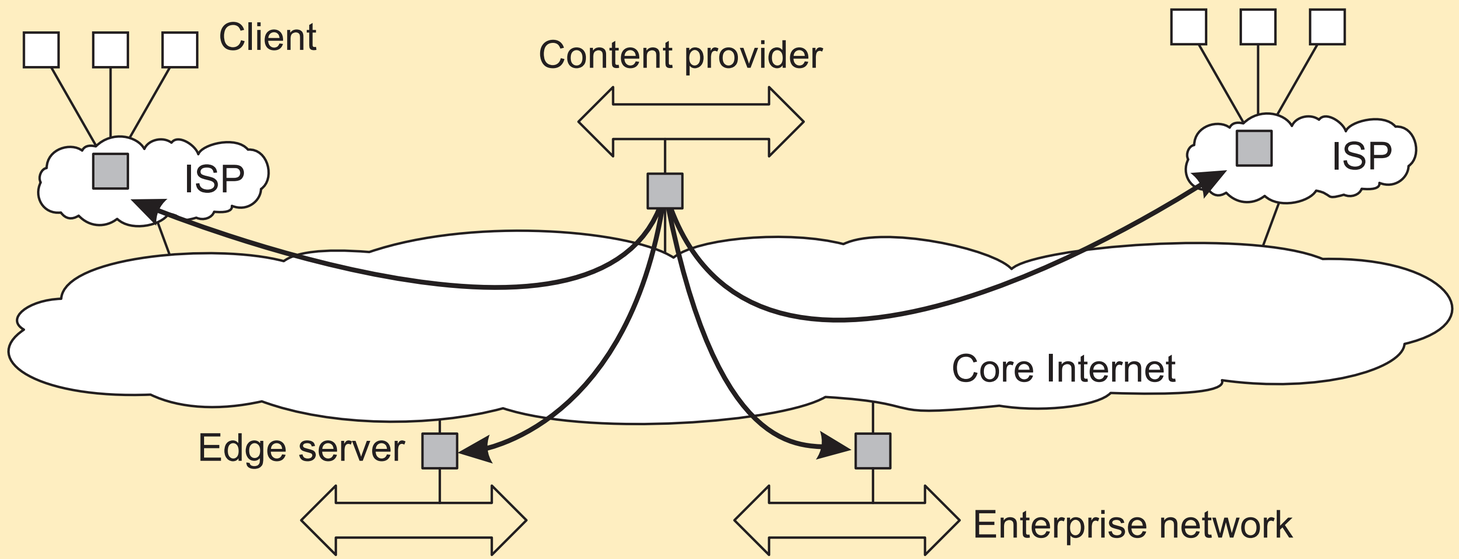
Hierarchically organized peer-to-peer networks

شبكات نظير إلى نظير منظمة بشكل هرمي

# Edge-server architecture

## Essence

Systems deployed on the Internet where servers are placed at the edge of the network: the boundary between enterprise networks and the actual Internet.



Architectures: System architecture

البنى: بنية النظام

Hybrid Architectures

البنى الهجينة

Edge-server architecture

بنية خادم الحافة

Essence

جوهر

Systems deployed on the Internet where servers are placed at the edge of the network: the boundary between enterprise networks and the actual Internet.

الأنظمة المنتشرة على الإنترنت حيث يتم وضع الخوادم على حافة الشبكة: الحد الفاصل بين شبكات المؤسسة والإنترنت الفعلي.

Client

عميل

Content provider

مزود المحتوى

ISP

مزود خدمة الإنترنت

Core Internet

الإنترنت الأساسية

Edge server

خادم الحافة

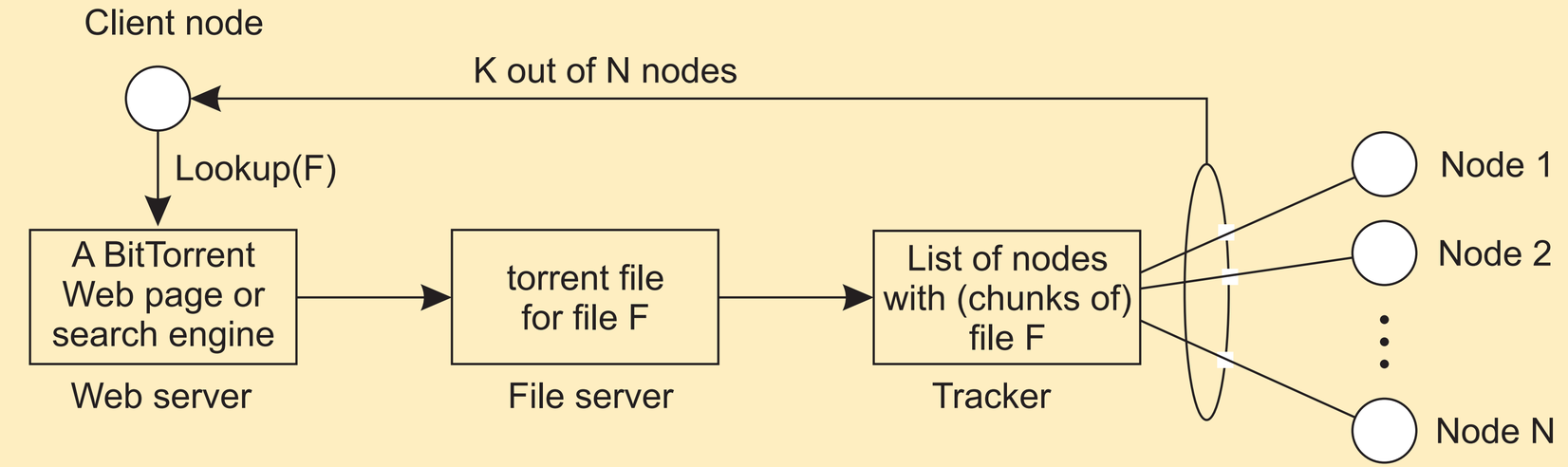
Enterprise network

شبكة المؤسسة

# Collaboration: The BitTorrent case

## Principle: search for a file *F*

- Lookup file at a global directory ⇒ returns a torrent file
- Torrent file contains reference to tracker: a server keeping an accurate account of active nodes that have (chunks of) *F*.
- *P* can join swarm, get a chunk for free, and then trade a copy of that chunk for another one with a peer *Q* also in the swarm.



Architectures: System architecture

البنى: بنية النظام

Hybrid Architectures

البنى الهجينة

Collaboration: The BitTorrent case

التعاون: قضية BitTorrent

Principle: search for a file F

المبدأ: ابحث عن ملف F

Lookup file at a global directory ) returns a torrent file Torrent file contains reference to tracker: a server keeping an accurate account of active nodes that have (chunks of) F.

ملف بحث في دليل عالمي ) يُرجع ملف تورنت يحتوي ملف تورنت على إشارة إلى المتعقب: خادم يحتفظ بحساب دقيق للعقد النشطة التي تحتوي على (أجزاء من) F.

P can join swarm, get a chunk for free, and then trade a copy of that chunk for another one with a peer Q also in the swarm.

يمكن لـ P الانضمام إلى السرب، والحصول على قطعة مجانًا، ثم استبدال نسخة من تلك القطعة بنسخة أخرى مع نظير Q أيضًا في السرب.

Client node

عقدة العميل

Lookup(F)

بحث (و)

K out of N nodes

K من العقد N

List of nodes

قائمة العقد

Node 1

العقدة 1

A BitTorrent

بت تورنت

torrent file

ملف تورنت

Node 2

العقدة 2

Web page or

صفحة ويب أو

with (chunks of)

مع (قطع)

for file F

للملف F

search engine

محرك البحث

file F

ملف ف

Web server

قاعدة بيانات للانترنت

File server

خادم الملفات

Tracker

المقتفي

Node N

العقدة ن

# BitTorrent under the hood

## Some essential details

- A tracker for file $F$ returns the set of its downloading processes: the current swarm.
- $A$ communicates only with a subset of the swarm: the neighbor set $N_A$.
- if $B \in N_A$ then also $A \in N_B$.
- Neighbor sets are regularly updated by the tracker

## Exchange blocks

- A file is divided into equally sized pieces (typically each being 256 KB)
- Peers exchange blocks of pieces, typically some 16 KB.
- $A$ can upload a block $d$ of piece $D$, only if it has piece $D$.
- Neighbor $B$ belongs to the potential set $P_A$ of $A$, if $B$ has a block that $A$ needs.
- If $B \in P_A$ and $A \in P_B$: $A$ and $B$ are in a position that they can trade a block.

Architectures: System architecture

البنى: بنية النظام

Hybrid Architectures

البنى الهجينة

BitTorrent under the hood

تورنت تحت الغطاء

Some essential details

بعض التفاصيل الأساسية

A tracker for file F returns the set of its downloading processes: the current swarm.

يقوم متتبع الملف F بإرجاع مجموعة عمليات التنزيل الخاصة به: السرب الحالي.

A communicates only with a subset of the swarm: the neighbor set NA.

يتواصل A فقط مع مجموعة فرعية من السرب: المجموعة المجاورة NA.

if B 2 NA then also A 2 NB.

إذا كان B 2 NA فأيضًا A 2 NB.

Neighbor sets are regularly updated by the tracker

يتم تحديث مجموعات الجوار بانتظام بواسطة جهاز التعقب

Exchange blocks

تبادل الكتل

A file is divided into equally sized pieces (typically each being 256 KB) Peers exchange blocks of pieces, typically some 16 KB.

يتم تقسيم الملف إلى أجزاء متساوية الحجم (عادة ما يكون حجم كل منها 256 كيلو بايت). يتبادل الزملاء كتلًا من القطع، عادةً حوالي 16 كيلو بايت.

A can upload a block d of piece D, only if it has piece D.

يمكن لـ A تحميل كتلة d من القطعة D، فقط إذا كانت تحتوي على القطعة D.

Neighbor B belongs to the potential set PA of A, if B has a block that A needs.

ينتمي الجار B إلى المجموعة PA المحتملة لـ A، إذا كان لدى B كتلة يحتاجها A.

If B 2 PA and A 2 PB: A and B are in a position that they can trade a block.

إذا كان B 2 PA و A 2 PB: A و B في وضع يمكنهم من تداول الكتلة.

# BitTorrent phases

## Bootstrap phase

$A$ has just received its first piece (through optimistic unchoking: a node from $N_A$ unselfishly provides the blocks of a piece to get a newly arrived node started).

## Trading phase

$|P_A| > 0$: there is (in principle) always a peer with whom $A$ can trade.

## Last download phase

$|P_A| = 0$: $A$ is dependent on newly arriving peers in $N_A$ in order to get the last missing pieces. $N_A$ can change only through the tracker.

Architectures: System architecture

البنى: بنية النظام

Hybrid Architectures

البنى الهجينة

BitTorrent phases

مراحل تورنت

Bootstrap phase

مرحلة التمهيد

A has just received its first piece (through optimistic unchoking: a node from NA unselfishly provides the blocks of a piece to get a newly arrived node started).

لقد استلم A للتو قطعته الأولى (من خلال التفكيك المتفائل: توفر عقدة من NA بشكل غير أناني كتل القطعة لبدء العقدة التي وصلت حديثًا).

Trading phase

مرحلة التداول

|PA| > 0: there is (in principle) always a peer with whom A can trade.

|السلطة الفلسطينية| > 0: يوجد (من حيث المبدأ) دائمًا نظير يمكن لـ A أن يتاجر معه.

Last download phase

مرحلة التحميل الأخيرة

|PA| = 0: A is dependent on newly arriving peers in NA in order to get the last missing pieces.

|السلطة الفلسطينية| = 0: يعتمد A على أقرانه الوافدين حديثًا إلى زمالة المدمنين المجهولين للحصول على آخر القطع المفقودة.

NA can change only through the tracker.

زمالة المدمنين المجهولين لا يمكن أن تتغير إلا من خلال جهاز التعقب.

# Self-managing Distributed Systems

**Observation**

Distinction between system and software architectures blurs when automatic adaptivity needs to be taken into account:

- Self-configuration
- Self-managing
- Self-healing
- Self-optimizing
- Self-*

**Warning**

There is a lot of hype going on in this field of autonomic computing.

Architectures

أبنية

2.4 Self-management in Distributed Systems

2.4 الإدارة الذاتية في الأنظمة الموزعة

Self-managing Distributed Systems

الأنظمة الموزعة ذاتية الإدارة

Observation

ملاحظة

Distinction between system and software architectures blurs when

التمييز بين بنيات النظام والبرمجيات يطمس عندما

automatic adaptivity needs to be taken into account:

يجب أن يؤخذ التكيف التلقائي بعين الاعتبار:

Self-configuration

التكوين الذاتي

Self-managing

إدارة الذات

Self-healing

الشفاء الذاتي

Self-optimizing

التحسين الذاتي

Self-*

الذات-*

Warning

تحذير

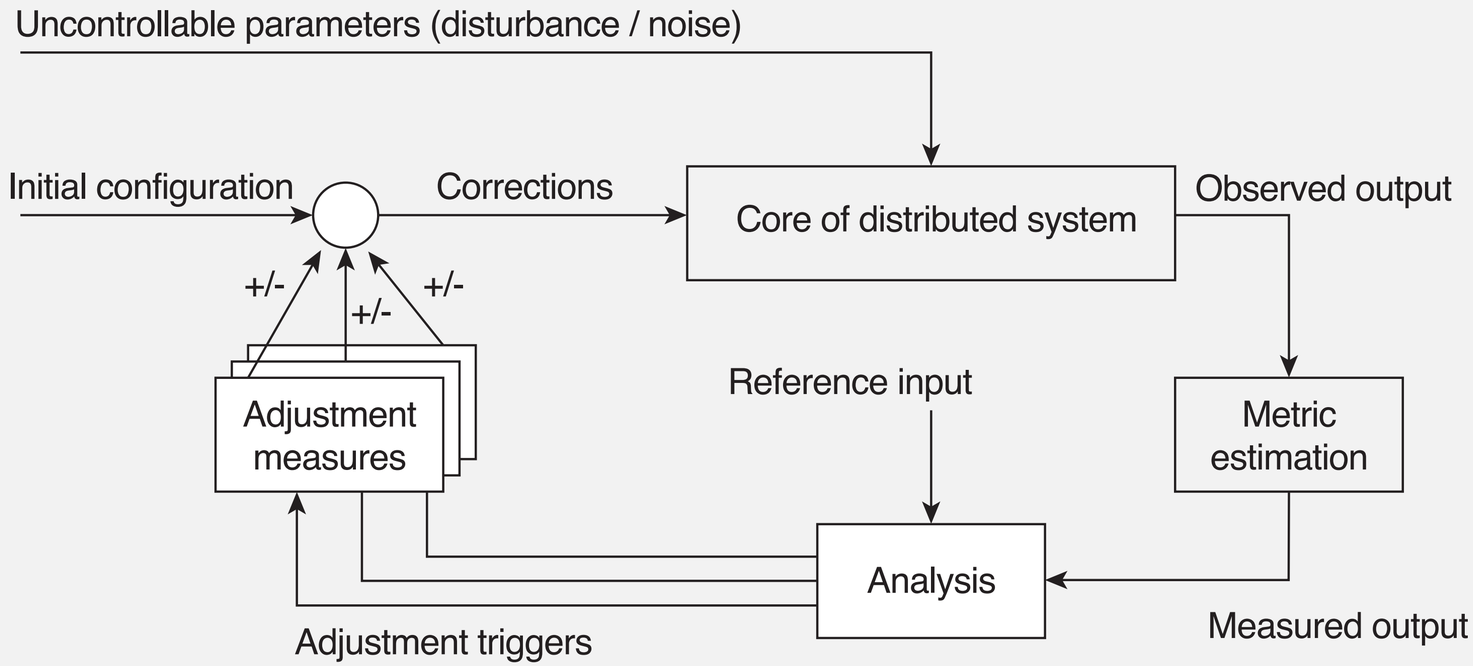
There is a lot of hype going on in this field of autonomic computing.

هناك الكثير من الضجيج الجاري في هذا المجال من الحوسبة المستقلة.

# Feedback Control Model

## Observation

In many cases, self-* systems are organized as a feedback control system.



Architectures 2.4 Self-management in Distributed Systems

البنيات 2.4 الإدارة الذاتية في الأنظمة الموزعة

Feedback Control Model

نموذج التحكم في ردود الفعل

Observation

ملاحظة

In many cases, self-* systems are organized as a feedback control

في كثير من الحالات، يتم تنظيم الأنظمة الذاتية كعنصر تحكم في ردود الفعل

system.

نظام.

Uncontrollable parameters (disturbance / noise)

معلمات لا يمكن السيطرة عليها (اضطراب / ضوضاء)

Initial configuration

الترتيب الأولي

Corrections

التصحيحات

Core of distributed system

جوهر النظام الموزع

Observed output

الإخراج الملاحظ

Reference input

المدخلات المرجعية

Metric Adjustment

التعديل المتري

measures

مقاسات

estimation

تقدير

Analysis

تحليل

Adjustment triggers Measured output

يؤدي التعديل إلى إنتاج مخرجات مقاسة