

الأكاديمية العربية الدولية



الأكاديمية العربية الدولية
Arab International Academy

الأكاديمية العربية الدولية المقررات الجامعية



الجامعة الافتراضية السورية
SYRIAN VIRTUAL UNIVERSITY

النظم الموزعة

الدكتور إيهاب عويشق



Books

النظم الموزعة

الدكتور إباء عويشق

من منشورات الجامعة الافتراضية السورية

الجمهورية العربية السورية 2018

هذا الكتاب منشور تحت رخصة المشاع المبدع – النسب للمؤلف – حظر الاشتقاق (CC– BY– ND 4.0)

<https://creativecommons.org/licenses/by-nd/4.0/legalcode.ar>

يحق للمستخدم بموجب هذه الرخصة نسخ هذا الكتاب ومشاركته وإعادة نشره أو توزيعه بأية صيغة وبأية وسيلة للنشر ولأية غاية تجارية أو غير تجارية، وذلك شريطة عدم التعديل على الكتاب وعدم الاشتقاق منه وعلى أن ينسب للمؤلف الأصلي على الشكل الآتي حصراً:

إباء عويشق، النظم الموزعة، من منشورات الجامعة الافتراضية السورية، الجمهورية العربية السورية، 2018

متوفر للتحميل من موسوعة الجامعة <https://pedia.svuonline.org/>

Distributed Systems

Ibaa Oueishek

Publications of the Syrian Virtual University (SVU)

Syrian Arab Republic, 2018

Published under the license:

Creative Commons Attributions- NoDerivatives 4.0

International (CC-BY-ND 4.0)

<https://creativecommons.org/licenses/by-nd/4.0/legalcode>

Available for download at: <https://pedia.svuonline.org/>



الفهرس

١	١. لمحة عامة عن النظم الموزعة.....
٣	(١) الخصائص الأساسية للأنظمة الموزعة.....
٥	(٢) تطور الأنظمة الموزعة.....
٨	(٣) أمثلة على الأنظمة الموزعة.....
١٣	(٤) تحديات الأنظمة الموزعة.....
١٥	(٥) تطبيقات الأنظمة الموزعة واستخداماتها.....
١٦	(٦) الأنظمة الموزعة المعيارية.....
١٧	٢. النماذج المختلفة للتوزيع، توزيع الموارد والبيانات والتحكم.....
١٨	(١) تذكير بالبنيان الحاسوبي.....
٢١	(٢) التعريف التفصيلي للأنظمة الموزعة.....
٢٩	(٣) النموذج المعتمد للنظام الموحد.....
٣١	٣. المخدم الزبون.....
٣٢	(١) المبدأ التصميمي للمخدم /زبون.....
٣٣	(٢) مثال تطبيقي أولي
٣٧	(٣) الخصائص المختلفة للرسائل.....
٣٧	❖ العنوان
٤٠	❖ الرسائل المتزامنة وغير المتزامنة.....
٤٢	❖ الرسائل المخزنة.....
٤٣	❖ وثوقية تبادل الرسائل.....
٤٤	❖ الرسائل المستخدمة في بناء التطبيقات.....
٤٧	(٤) استدعاء الإجراءات عن بعد
٤٧	❖ المراحل الأساسية.....
٥٠	❖ تمرير الوسائط.....
٥٥	❖ معالجة الاعطال.....
٥٨	٤. التزامن في الأنظمة الموزعة.....
٥٩	(١) مقدمة.....
٦٠	(٢) مزامنة الميقاتية.....
٦٥	(٣) الاستبعاد المتبادل.....
٦٥	❖ خوارزمية مركزية.....
٦٦	❖ خوارزمية موزعة.....
٦٩	❖ خوارزمية حلقة.....
٧٠	❖ مقارنة الخوارزميات الثلاث.....
٧٢	(٤) المناقلات.....
٧٣	❖ نموذج المناقلات.....
٧٥	❖ تحقيق المناقلات.....
٧٩	❖ المناقلات والموارد.....

٨٣	٥. أنظمة التشغيل الموزعة.....
٨٤	(١) الإجراءات الموزعة.....
٨٧	(٢) نماذج إدارة الموارد في النظام الموزع.....
٨٧	❖ نموذج محطة العمل.....
٨٩	❖ نموذج المجمع.....
٩٠	(٣) أنظمة الملفات الموزعة.....
٩١	❖ تصميم أنظمة الملفات الموزعة.....
٩٥	❖ دلالة التشارك في الملفات.....
٩٩	٦. المراجع.....

.

:

.

:

.

.1

:

.

concurrency

.

.

()

.

.

.

.

.

.

.

.

.

	:	
	:access transparency	•
	:location transparency	•
)		
	.(
	:concurrency transparency	•
	:replication transparency	•
	:Failure transparency	•
	:mobility transparency	•
ف	:performance transparency	•
عرضياً	:scaling transparency	•

.2

:

.

.

.10Gbps

)

(100Gbps

Gbps 40

⋮

⋮

⋮

⋮

⋮

⋮

⋮

⋮

⋮

⋮

parallel systems

⋮

⋮

⋮

⋮

⋮

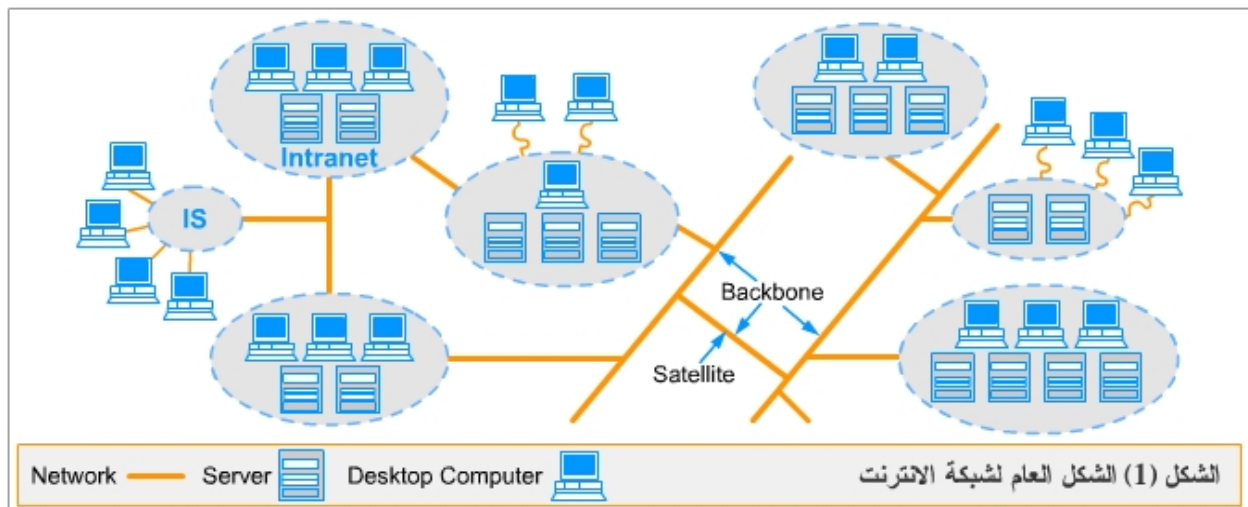
⋮

.3

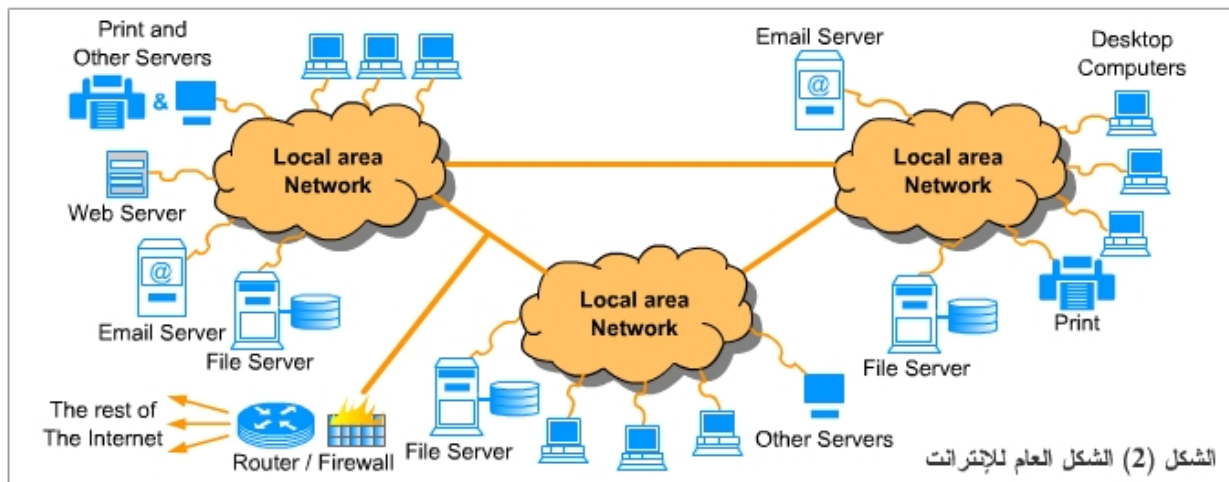
:

-
-
-

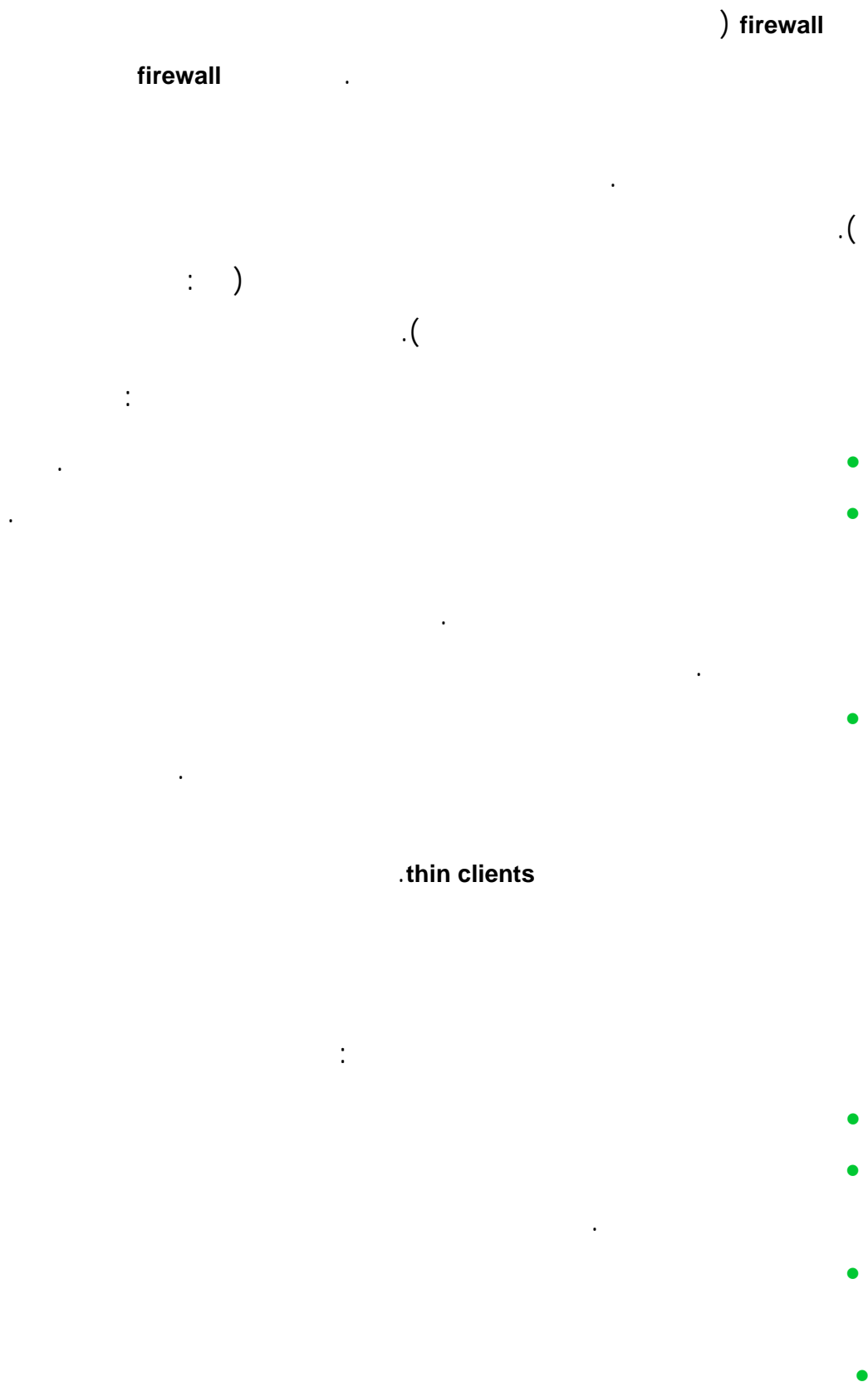
1-1



2 .()



router



()

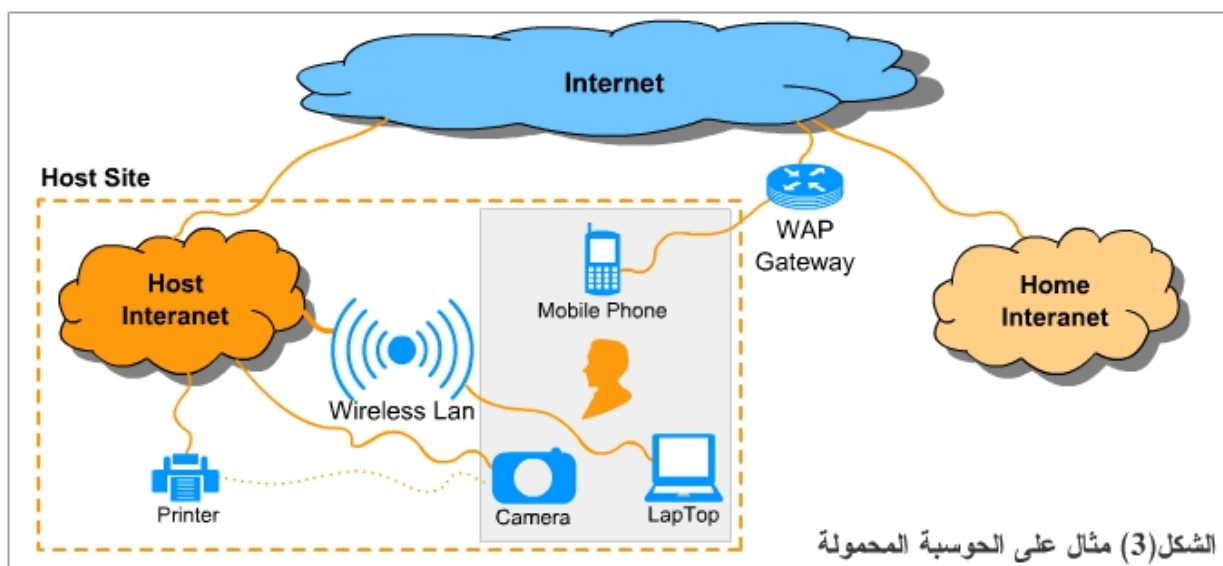
.location aware computing

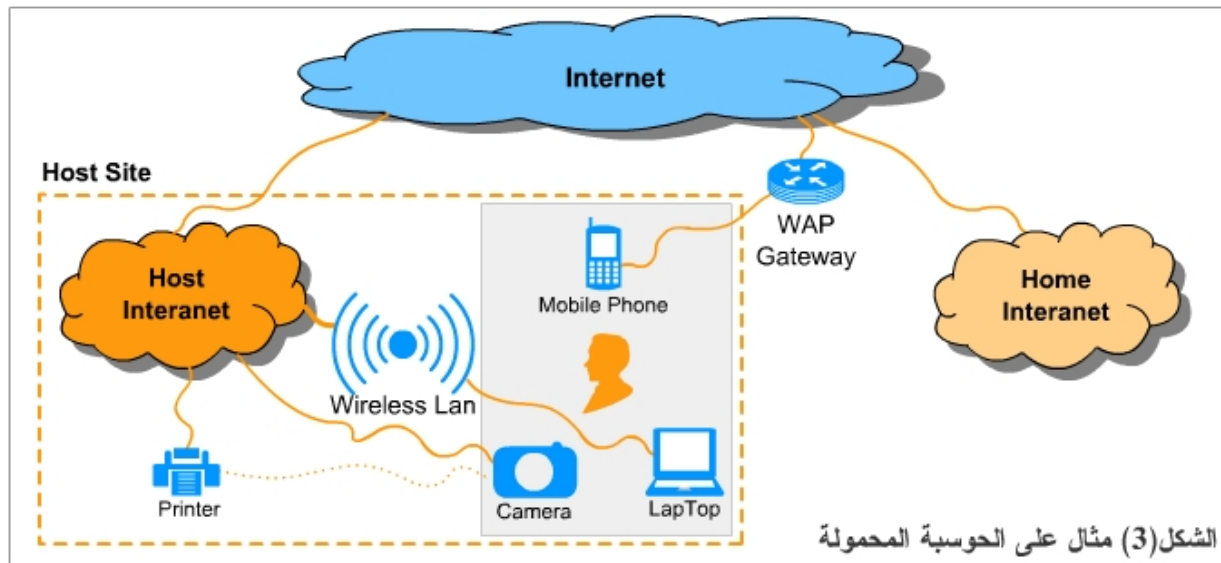
.ubiquitous computing

.(host organization

)

3





:

()

.WAP

Bluetooth

تحديات

.4

.
:Heterogeneity

.
middleware

.
:Openness

.
:security

.
Denial of service

.
:scalability

.(=)

.

:

.

.

:concurrency

.

.

()

.

(:)

.

.5

.1

.2

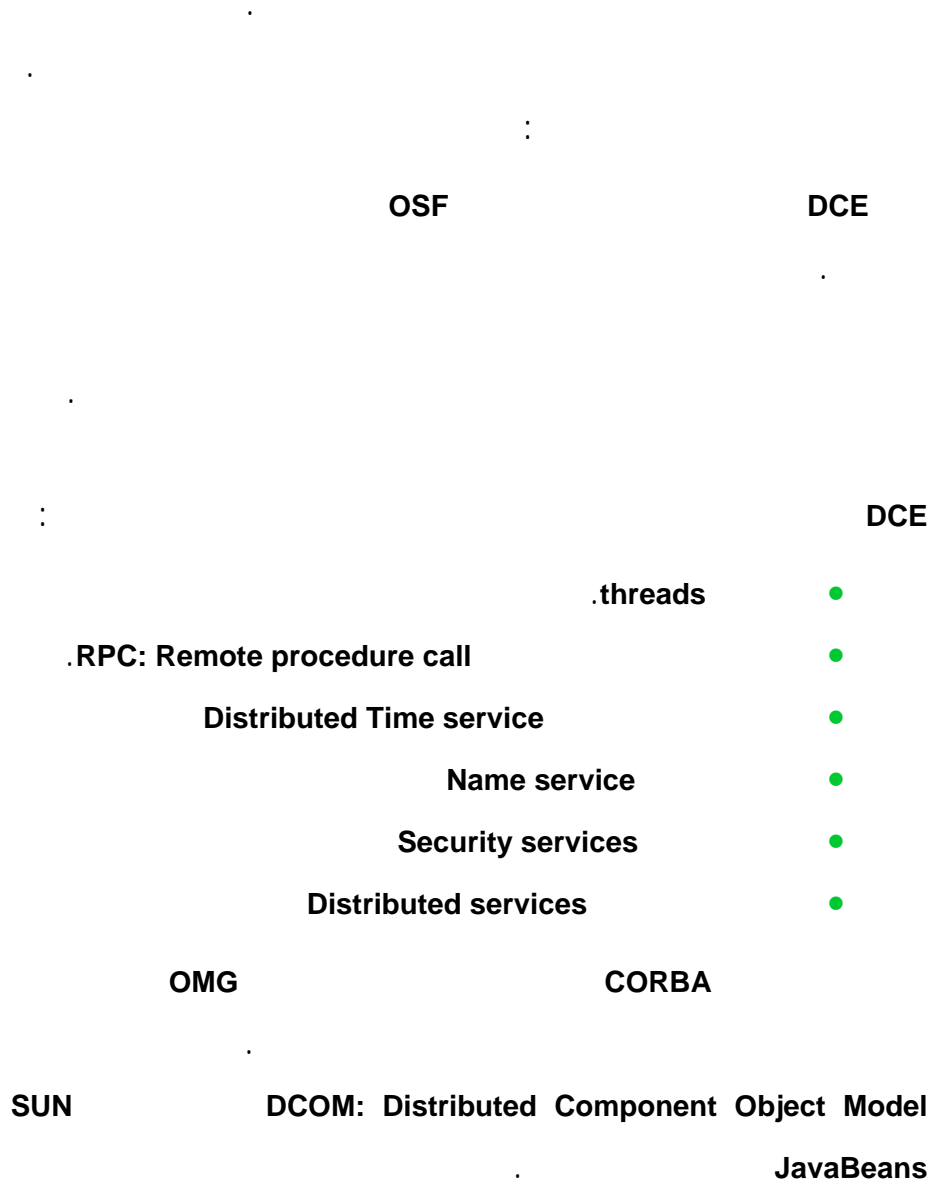
.3

.4

groupware

video conference

Interoperability

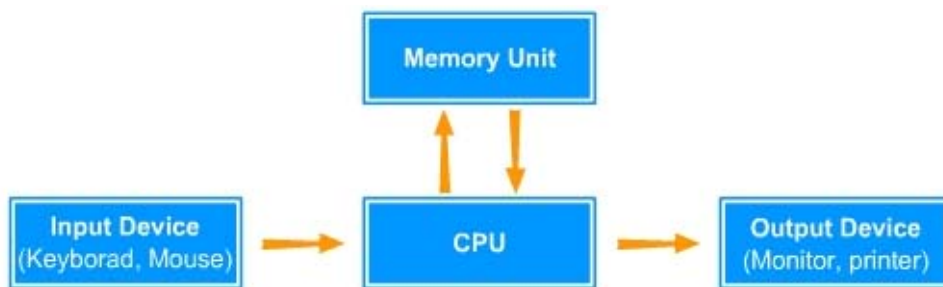


المختلفة

.1

CPU Von Neumann memory unit
غ .I/O

()



الشكل (1) مكونات الحاسوب بحسب نموذج فون نيومان Von Neumann

:

2-a

.1

.tightly coupled system

.2-b

.2

()

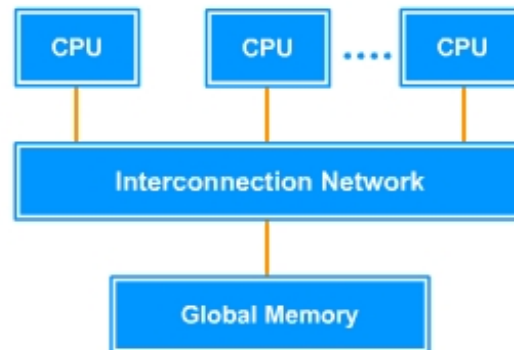
Processing Element

.loosely coupled systems

2

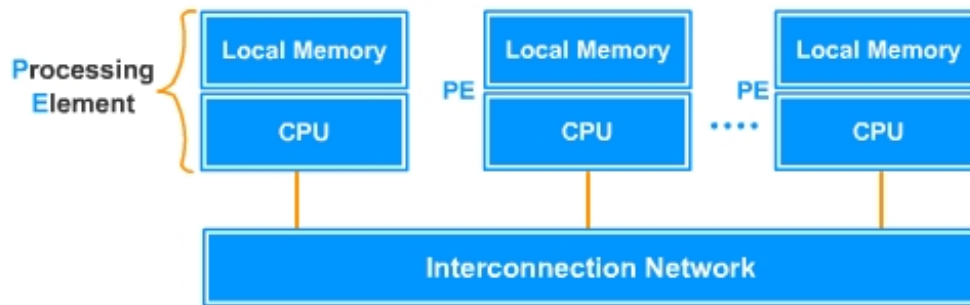
.receive

send



(a)

الشكل (2-أ): الأنظمة شديدة الترابط tightly coupled system



(b)

الشكل (2-ب): الأنظمة قليلة الترابط loosely coupled systems

الشكل (2) شكلان أساسيان للتنظيم الحاسوبي متعدد المعالجات

	Logically Shared	Logically distributed
Physically Shared	Shared Memory	Simulated Message Passing
Physically distributed	Distributed Shared Memory	Message Passing

3

shared memory

processes

critical section

monitor

semaphore

.barriers

message passing

send

simulated message passing

.receive

buffers

distributed shared memory (DSM)

distributed shared virtual memory

distributed memory system

.2

concurrent parallel network distributed :
.decentralized

)

.(

parallel

(Single-instruction, multiple-data) SIMD

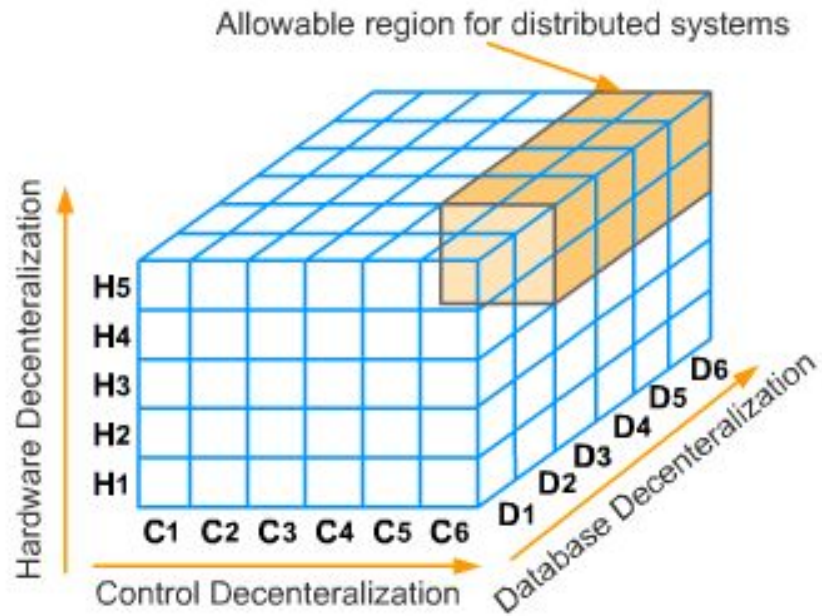
centralized

decentralized

networked

Enslow

Distributed system = distributed hardware + distributed control + distributed data



الشكل (4) نموذج إتسئو للتوزيع

:H5 H1

)

.(

.ALU

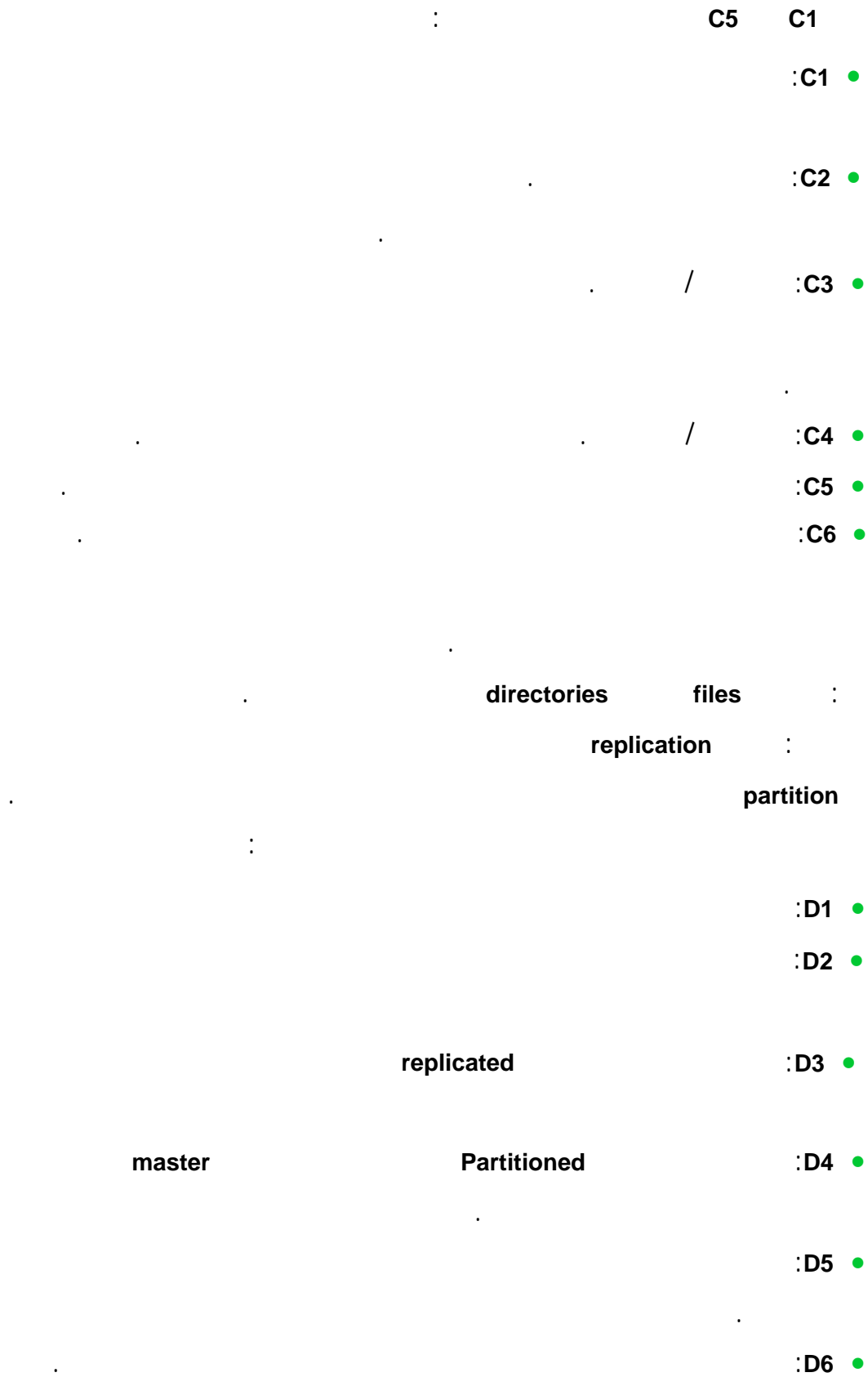
:H1 •

:H2 •

:H3 •

:H4 •

:H5 •



4

:

H4

D3

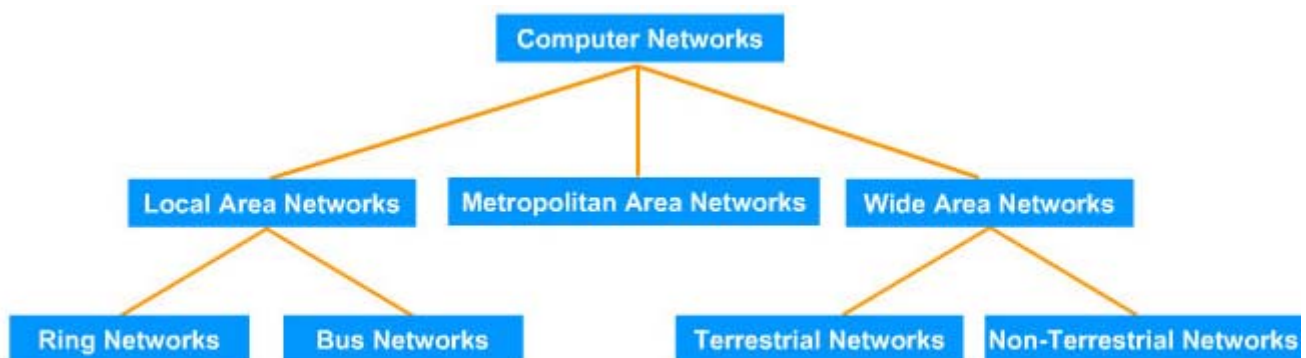
C5

-
-
-

symptoms

:

-
-
-
-



الشكل (5) تصنيف الشبكات الحاسوبية

6 5

data flow

greedy evaluation

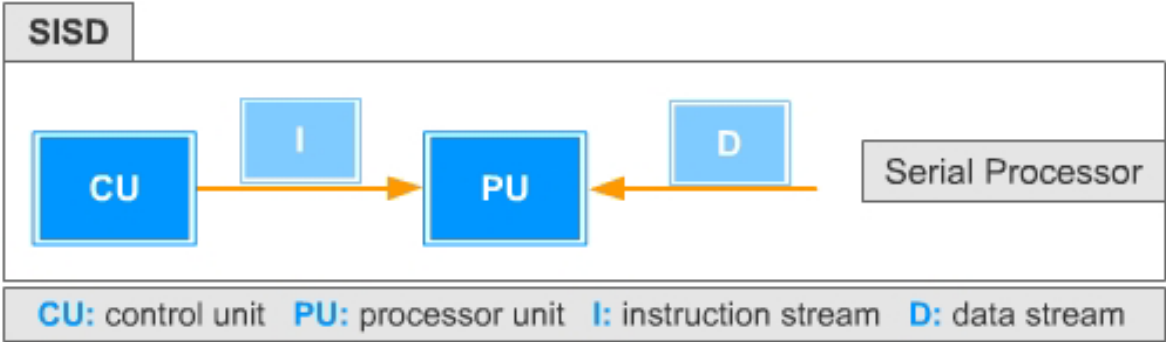
reduction model

Flynn

:

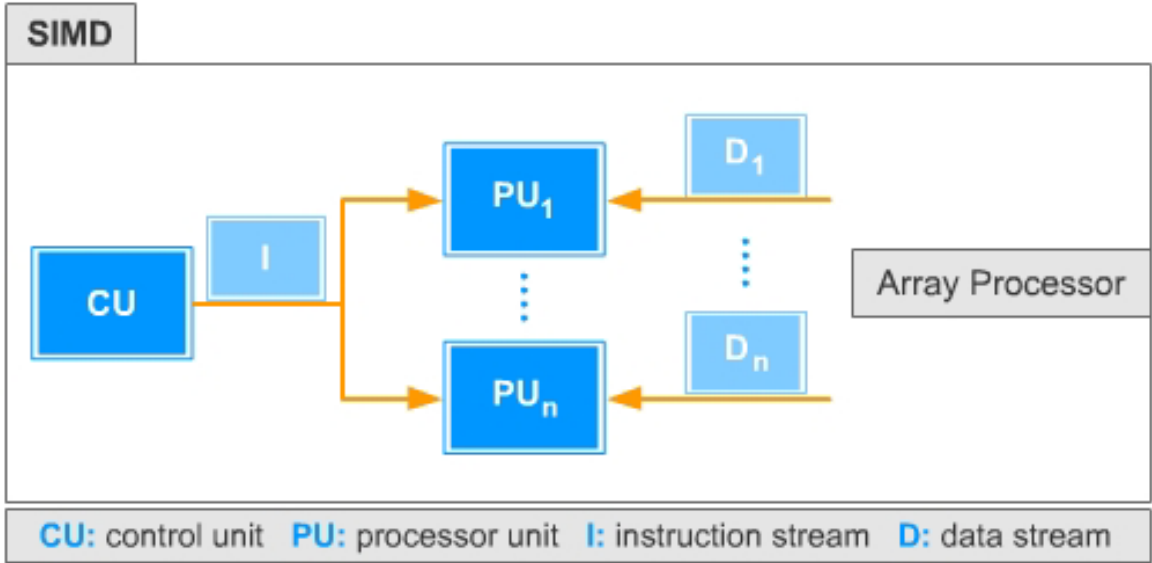
:Single Instruction Single Data (SISD) ●

pipeline

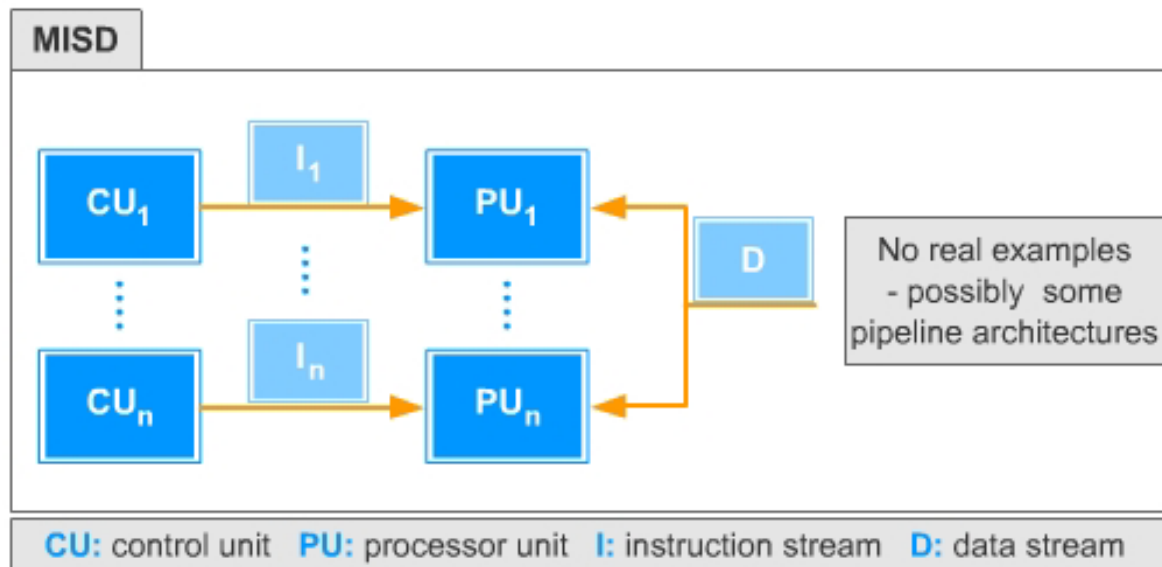


:Single Instruction Multiple Data (SIMD) ●

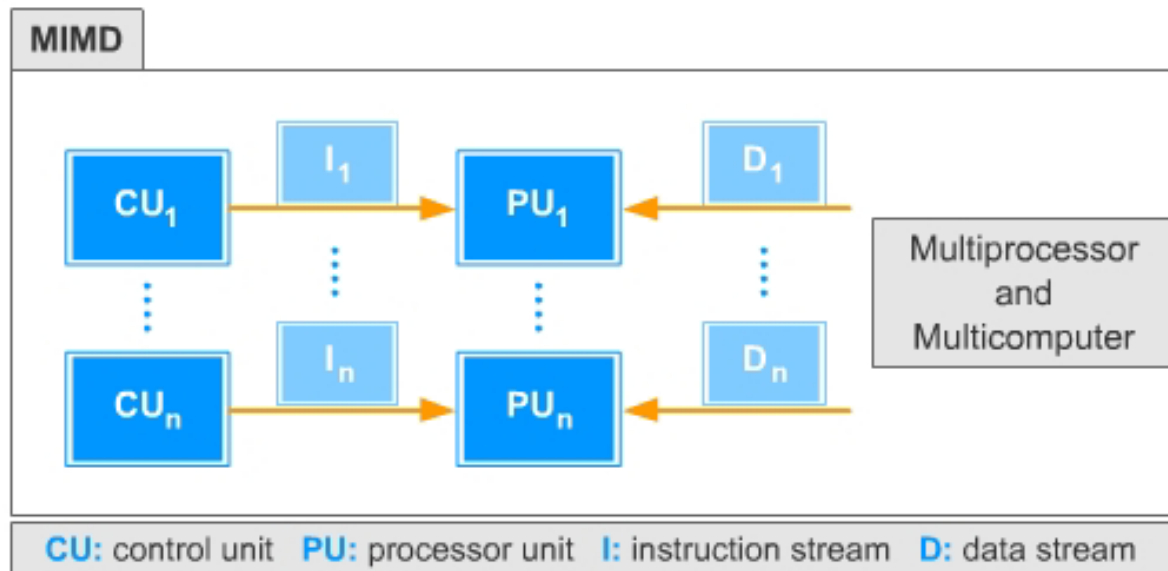
.systolic

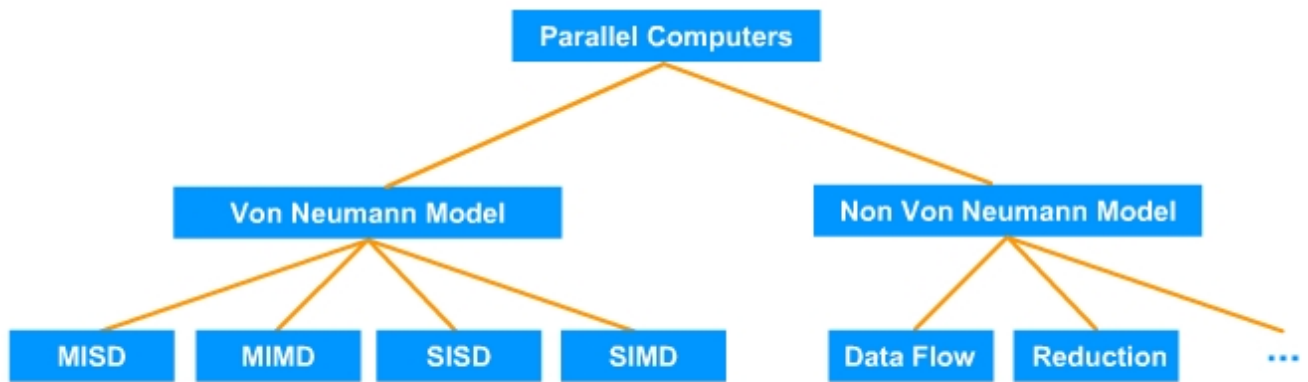


:Multiple Instruction Single Data (MISD) •



:Multiple Instructions Multiple Data (MIMD) •





الشكل (6) تصنيف الحواسيب التفرعية

6 5

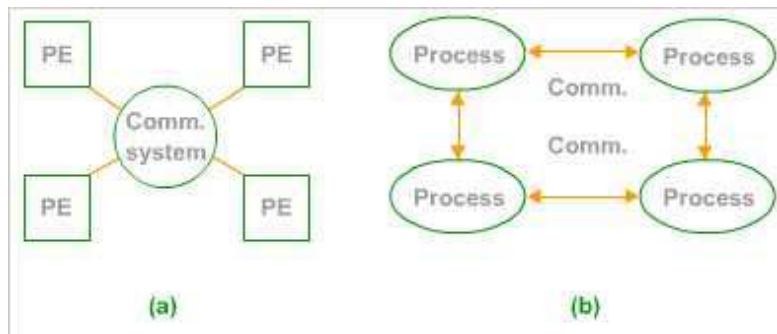


لتبادل

bus

.point-to-point –

.3



(7)

7

المنطقية

.logical resource

.physical resource

:graceful degradation

(LAN, MAN, WAN)

()

:

Tanenbaum

virtual uniprocessor

.file sharing semantics

:

•

•

•

•

/ .1

client-server /

()

(:)



send(dest, &mptr) :

dest

mptr

receive(addr, &mptr) :

```

/* Definitions needed by clients and servers */
#define TRUE 1
#define MAX_PATH 255 /* maximum length of file name */
#define BUF_SIZE 1024 /* how much data to transfer at once */
#define FILE_SERVER 243 /* file server's network address */

/* Definitions of the allowed operations */
#define CEATE 1 /* create a new file */
#define READ 2 /* read data from a file and return it */
/* */
#define WRITE 3 /* write data to a file */
#define DELETE 4 /* delete an existing file */

/* Error codes */
#define OK 0 /* operation performed correctly */
#define E_BAD_OPCODE -1 /* unknown operation requested */
#define E_BAD_PARAM -2 /* error in a parameter */
#define E_IO -3 /* disk error or other I/O error */

/* Definitions of the message format */
struct message {
    long source; /* sender's identity */
    long dest; /* receiver's identity */
    long opcode; /* requested operation */

    long count; /* number of bytes to transfer */

    long offset; /* position in file to start I/O */
    long result; /* result of the operation */
    char name[MAX_PATH]; /* name of file being operated on */
    char data[BUF_SIZE]; /* data to be read or written */
};

```

```

(
    .header.h
    MAXPATH :
    BUF-SIZE (
FILE-SERVER
    .
    .
    .
    .(
        .OK (
            .(K.BAD_OPCODE, K.BADYARAM)
10
    )
    .(

```

Field name	Field meaning	Possible values
Source, dest	Identify the sender and receiver	
Opcode	Operation code	<i>CREATE, READ, WRITE, DELETE</i>
Count, offset	Used for parameters	
Extra1, extra2	Provide space for additional parameters in case the server is expanded in the future	
Result	Holds result value for server-to-client replies	
Name (array)	Name of the file being Accessed	
Data (array)	holds the data sent back on a reply to READ or the data sent to the server on a WRITE	

: (2)

```
# include <header.h>
void main(void) {
    struct message m1, m2;           /* incoming and outgoing messages */
    int r;                           /* result code */

    while(TRUE) {                   /* server runs forever */
        receive(FILE_SERVER, &m1);  /* block waiting for a message */
        switch(m1.opcode) {         /* dispatch on type of request */
            case CREATE:             r= do_create(&m1,&m2); break;
            case READ:               r= do_read(&m1,&m2); break;
            case WRITE:              r= do_write(&m1,&m2); break;
            case DELETE:             r= do_delete(&m1,&m2); break;
            default:                 r=E_BAD_OPCODE;
        }
        m2.result=r;                 /* return result to client */
        send(m1.source, &m2);        /* send reply */
    }
}
```

(a)

```
# include <header.h>
int copy(char *src, char *dst){      /* procedure to copy file using the server */
    struct message m1;               /* message buffer */
    long position;                   /* current file position */
    long client=110;                 /* client's address */

    initialize();                    /* prepare for execution */
    position =0;
    do {
        m1.opcode = READ;            /* operation is a read */
        m1.offset = position;        /* current position in the file */
        m1.count = BUF_SIZE;        /* how many bytes to read */
        strcpy(&m1.name, src);       /* copy name of file to be read to message */
        send(FILESERVER, &m1);       /* send the message to the file server */
        receive(client, &m1);        /* block waiting for the reply */

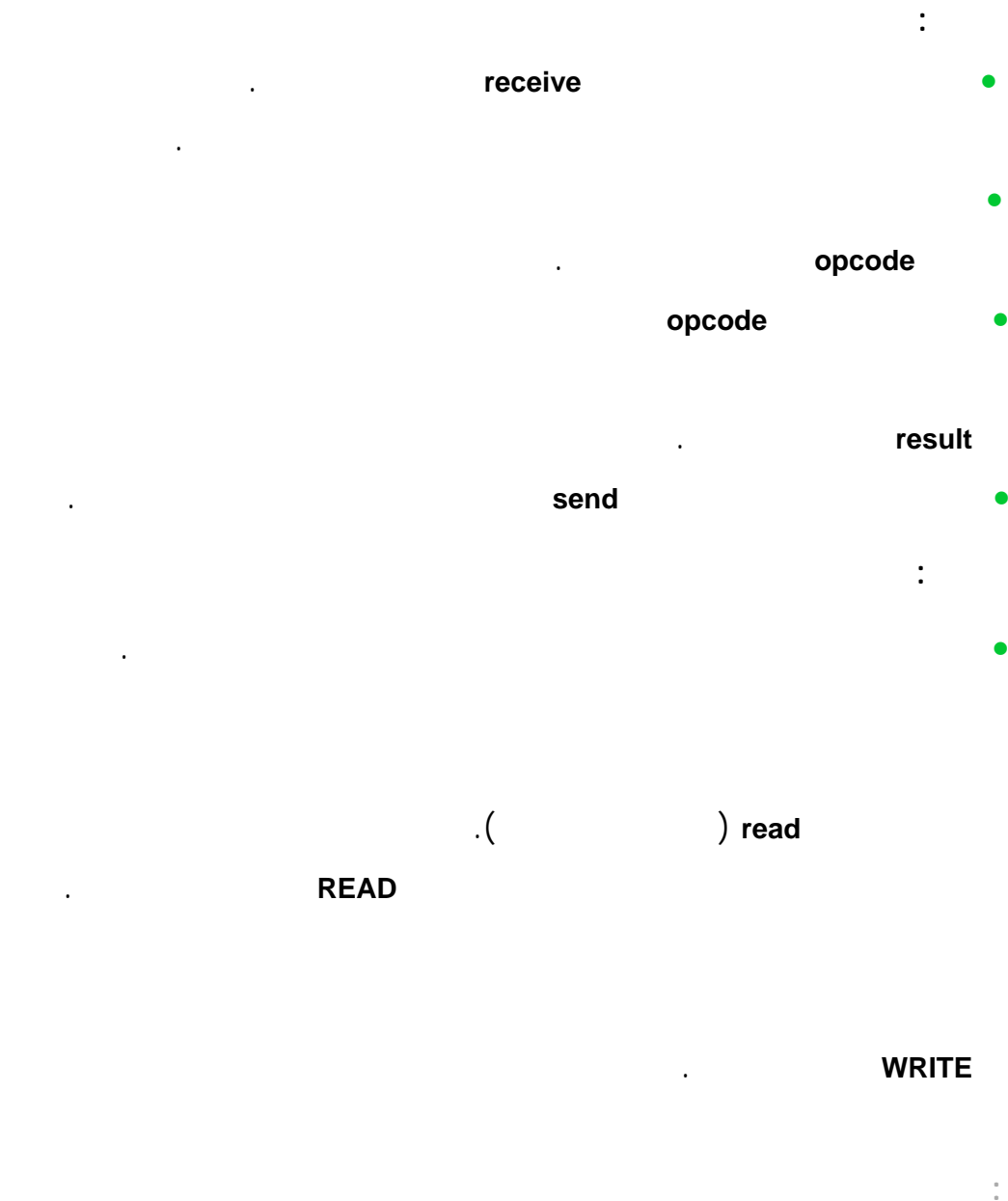
        /* Write the data just received to the destination file */
        m1.opcode = WRITE;           /* operation is a write */
        m1.offset = position;        /* current position in the file */
        m1.count = m1.result;        /* how many bytes to write */
        strcpy(&m1.name, dst);       /* copy name of file to be written to buf */
        send(FILESERVER, &m1);       /* send the message to the file server */
        receive(client, &m1);        /* block waiting for the reply */
        position +=m1.result;        /* m1.result is number of bytes written */
    } while(m1.result > 0);          /* iterate until done */
    return(m1.result>=0 ? OK : m1.result); /* return OK or error code */
}
```

(b)

(b)

(a)

: (2)



/

.3

header.h

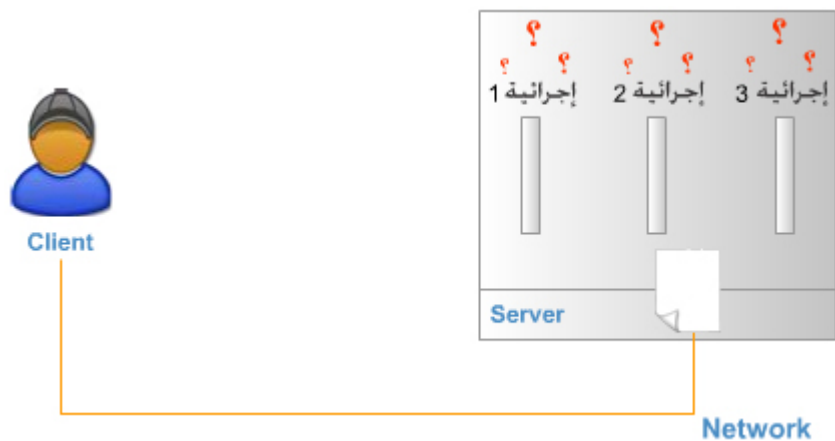
```
#define MAX_PATH 255      /* maximum length of a file name */
#define BUF_SIZE 1024     /* how much data to transfer at once */
#define FILE_SERVER 243   /* file server's network address */
```

MAC

) 2

(Address

.()

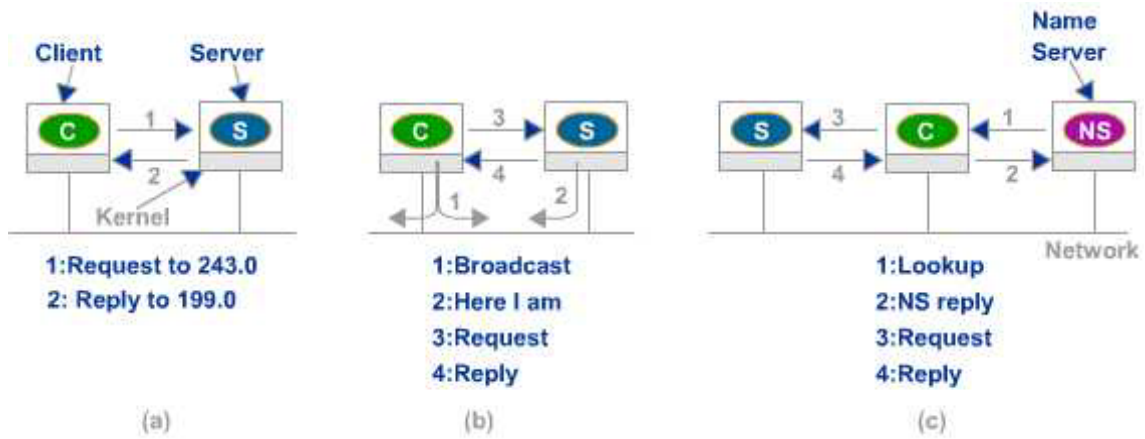


4

243@4 243.4 :) machine.process

.(a) 3

.(243



(3)

machine.local-id

local-id

machine.localid

local-id

socket

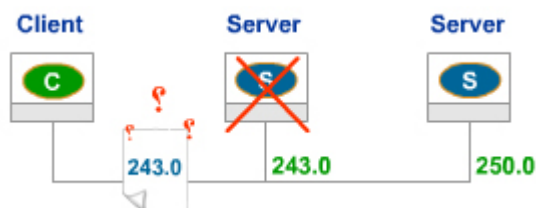
.Berkeley UNIX

()

243

250

.243



:

)

:

(64

broadcasting

locate packet

here I am

cache

.(b) 3

.name server

.(c) 3

:

	machine.procid

Synchronous

suspended

بدرجة

.Asynchronous

(b)4



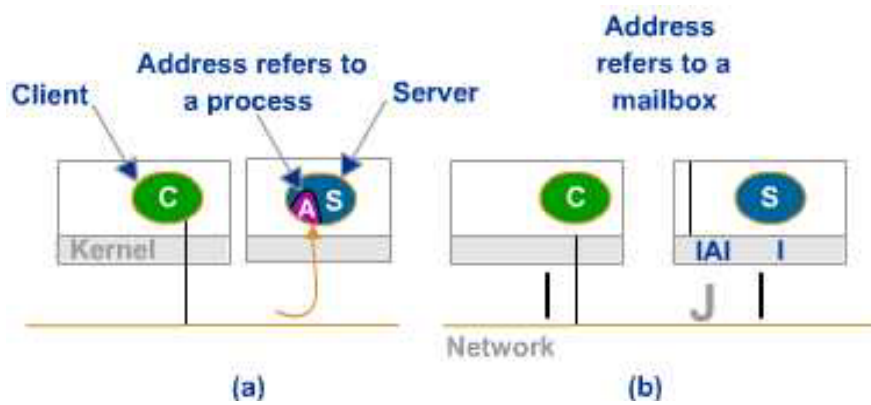
.1

.2

.3

.4-a

receive



(4)

:

.

:

.4-b

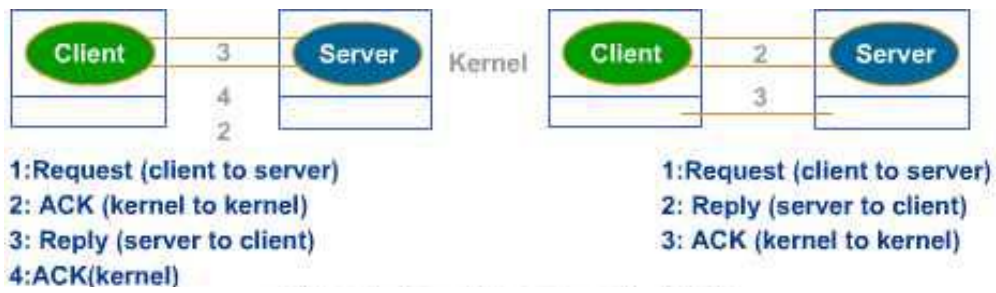
.buffered messages

.5-b

reply
request

ACK

.5-a



(5)

/

/

81

:

.

()

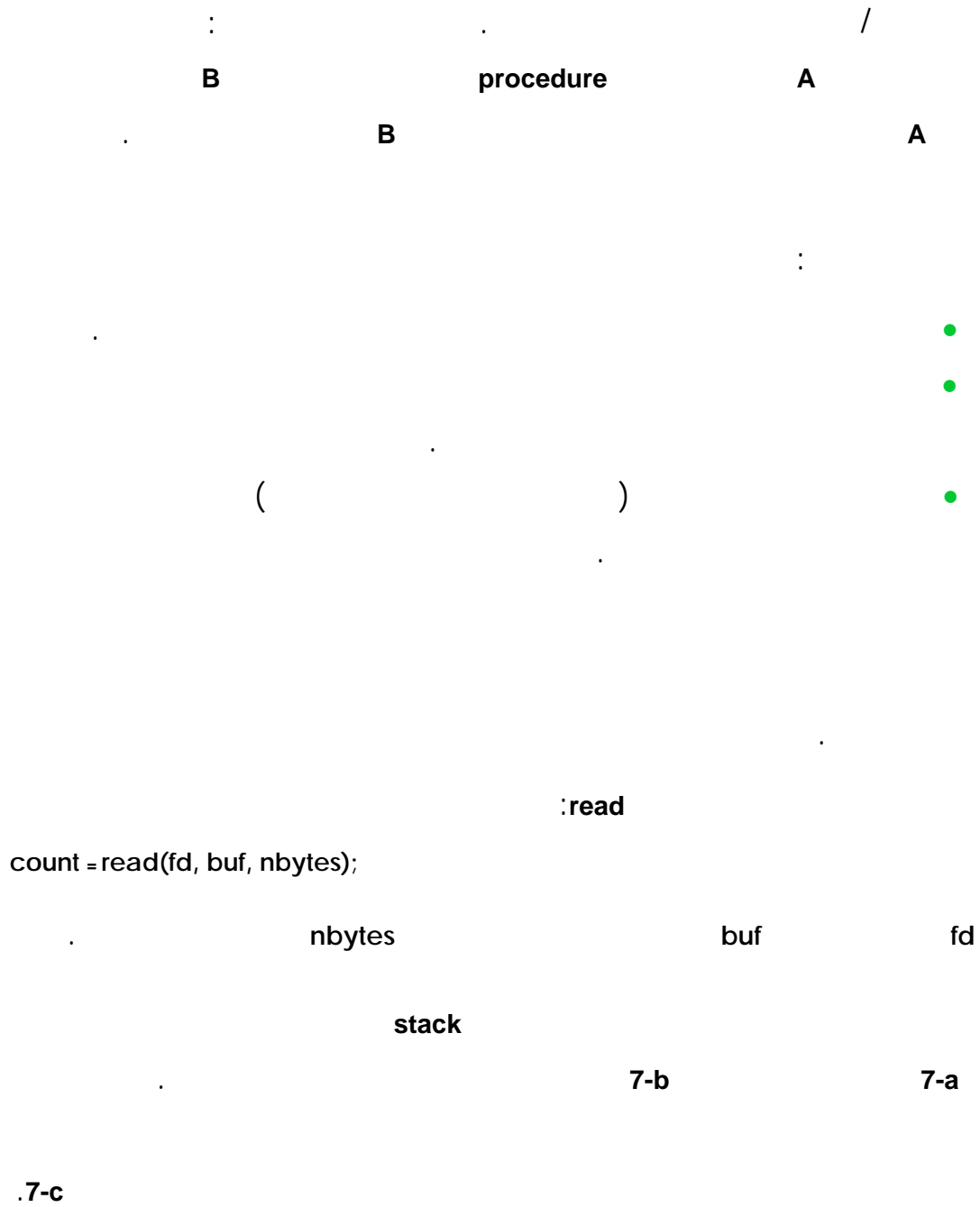


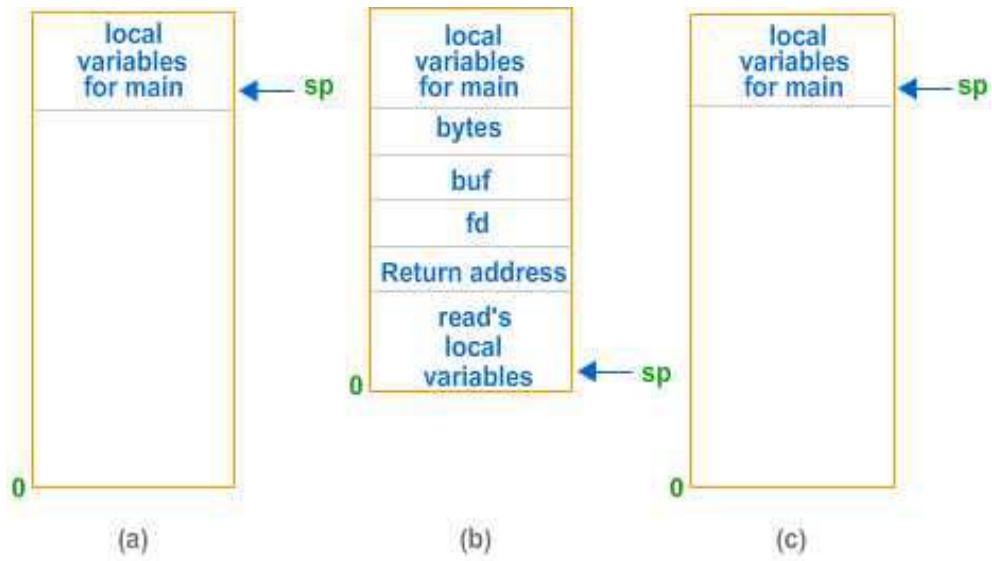
Code	Packet type	From	To	Description
REO	Request	Client	Server	The client wants service
REP	Reply	Server	Client	Reply from the server to the client
ACK	Ack	Either	Other	The previous packet arrived
AYA	Are you alive	Client	Server	Probe to see if the server has crashed
IAA	I am alive?	Server	Client	The server has not crashed
TA	Try again	Server	Client	The server has no room
AU	Address unknown	Server	Client	No process is using this address

الشكل (6) الأنواع المختلفة من الرزم المستخدمة في التخاطب
بين الزبون والمخدم

Remote Procedure Call

.4





(7)

:

:

C

.1

C

.2

stub

)

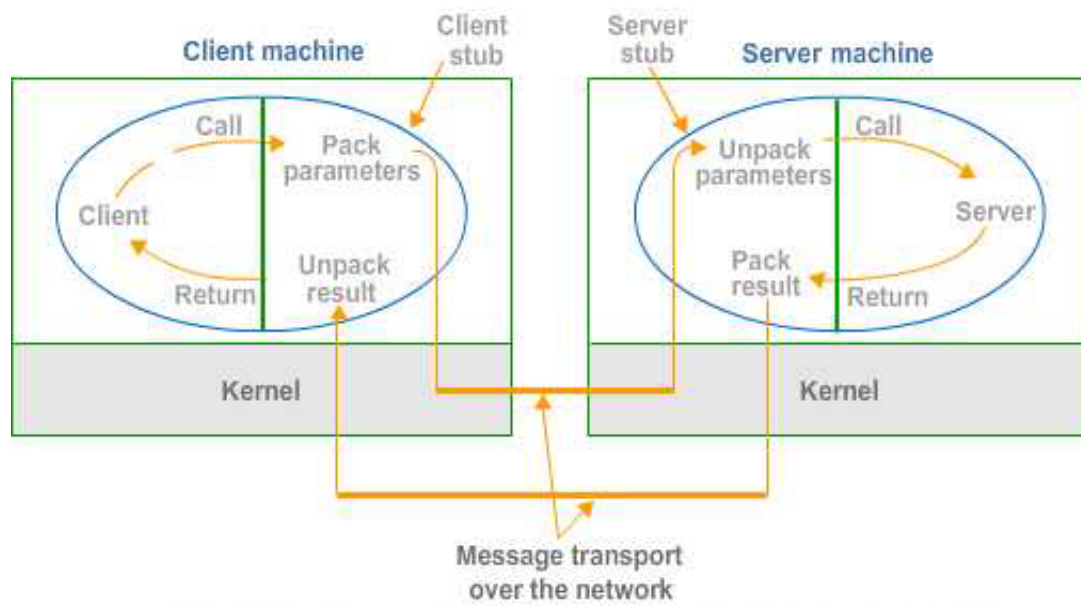
)

.(

.(

receive

.8



(8)

.3

(7)

read

.4

.5

الزبون

:

.1

.2

.3

.4

.5

.6

.7

.8

.9

.10

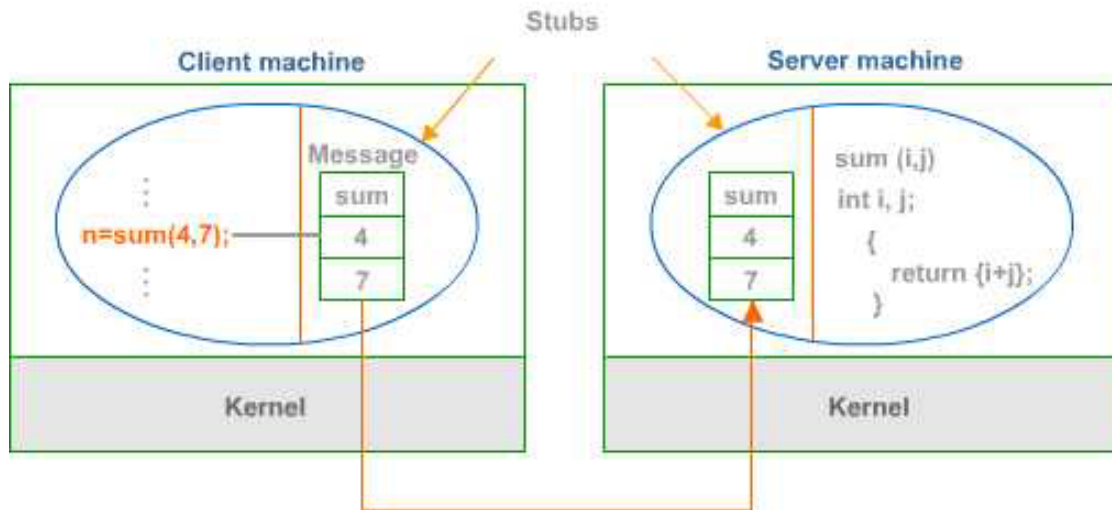
IBM

ASCII

EBCDIC

-9-

windows



sum (4,7)

(9)

floating point

10

.sun

:

a

5

.Intel

.JILL

3	2	1	0
0	0	0	5
7	6	5	4
L	L	I	J

(a)

0	1	2	3
5	0	0	0
4	5	6	7
J	I	L	L

(b)

0	1	2	3
0	0	0	5
4	5	6	7
L	L	I	J

(c)

(10)

b

SPARC

83,886,080 $2 \cdot 2^{24}$

n

n+1

$$\vdots$$

11-a

WORD

byte

.11-b

```
foobar (x,y,z)
char x;
float y;
int z [5]
{
}
```

(a)

foobar
x
y
5
z[J]
z[1]
z[2]
z[3]
z[4]

(b)

(11)

.()

ASCII

2

⋮

IEEE

1 0

.little endian

(
big endian

Windows

OIDL

.IDL: Interface Description Language

:

OIDL

IDL

appli.h



.appli_s.h appli_c.h :



```
// hello.idl: Interface specification for two procedures.  
interface hello  
{  
  
void HelloProc([in, string] unsigned char * pszString);  
  
void Shutdown(void);  
  
}
```

/

:

IDL

.1

.OIDL

.2

.()

.3

.()

.4

.

pointers

.

.

:

.

.

.

.

.



: .4

: .

.(exception)

.

.

:

at least once

at most once

0

.

.

.

exactly once

()

.

:

" "

orphan

.5

:

•

•

()

•

$$(\mathbb{I} \quad \quad \quad) \quad \quad \quad : \quad \quad \quad -$$

•

$$\cdot (\quad)$$

•

• [How to use the new Google Analytics interface](#)

•

■

■

•

T

.

•

•

.1



monitor



.mutual exclusion

.transactions :

deadlock

clock synchronization

.2

- .1
 - .2
 - .3
 - .4
- .single point of failure

B A

.B A

UNIX Make : (1)

)

.(

2150 input.o

2151 input.c

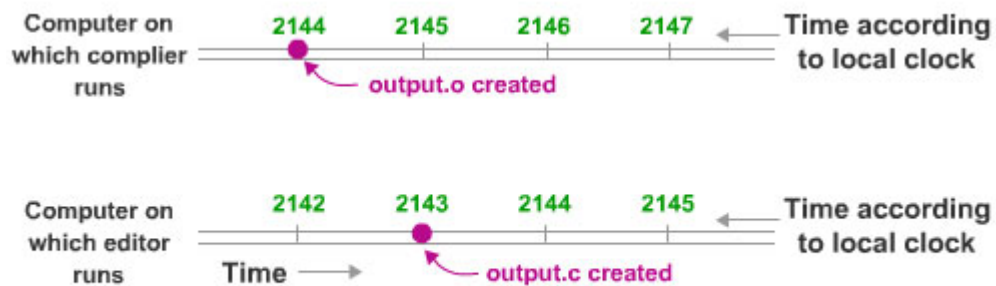
input.c make

2144 input.c

2144 output.o

output.c

.2143



الشكل (1) مثال على مشكلة الاتفاق على الوقت في نظام موزع

:

logical clock

%100

.
 .(make)
 :

1978 Lamport

.
 .
 .
 .
 .(input.o input.c) make

.logical clock

:happensbefore -

b a a ~ b
 : .a b
 b a .1
 a ~ b
 .b a
 b a .2
 a ~ b

.a ~ c b ~ c a ~ b



concurrent

b a

(b ~ a a ~ b)

C(a) a

. a ~ b => C(a) < C (b) :

b

a

.b

a

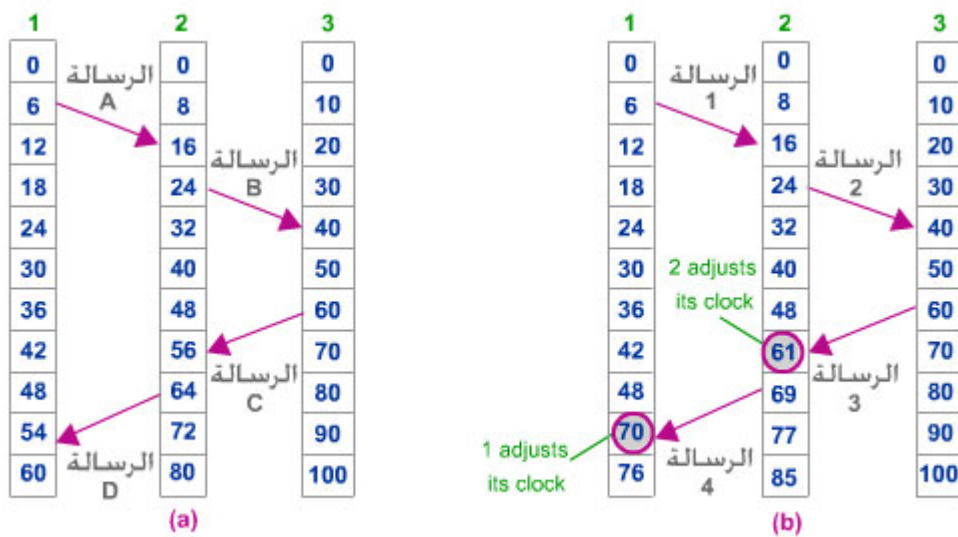
2-a

10 و 1

8

0

2



الشكل (2) مثال على خوارزمية لامبورث

.1

A

0

6

.6 :

16

2

A

10 :

12

3

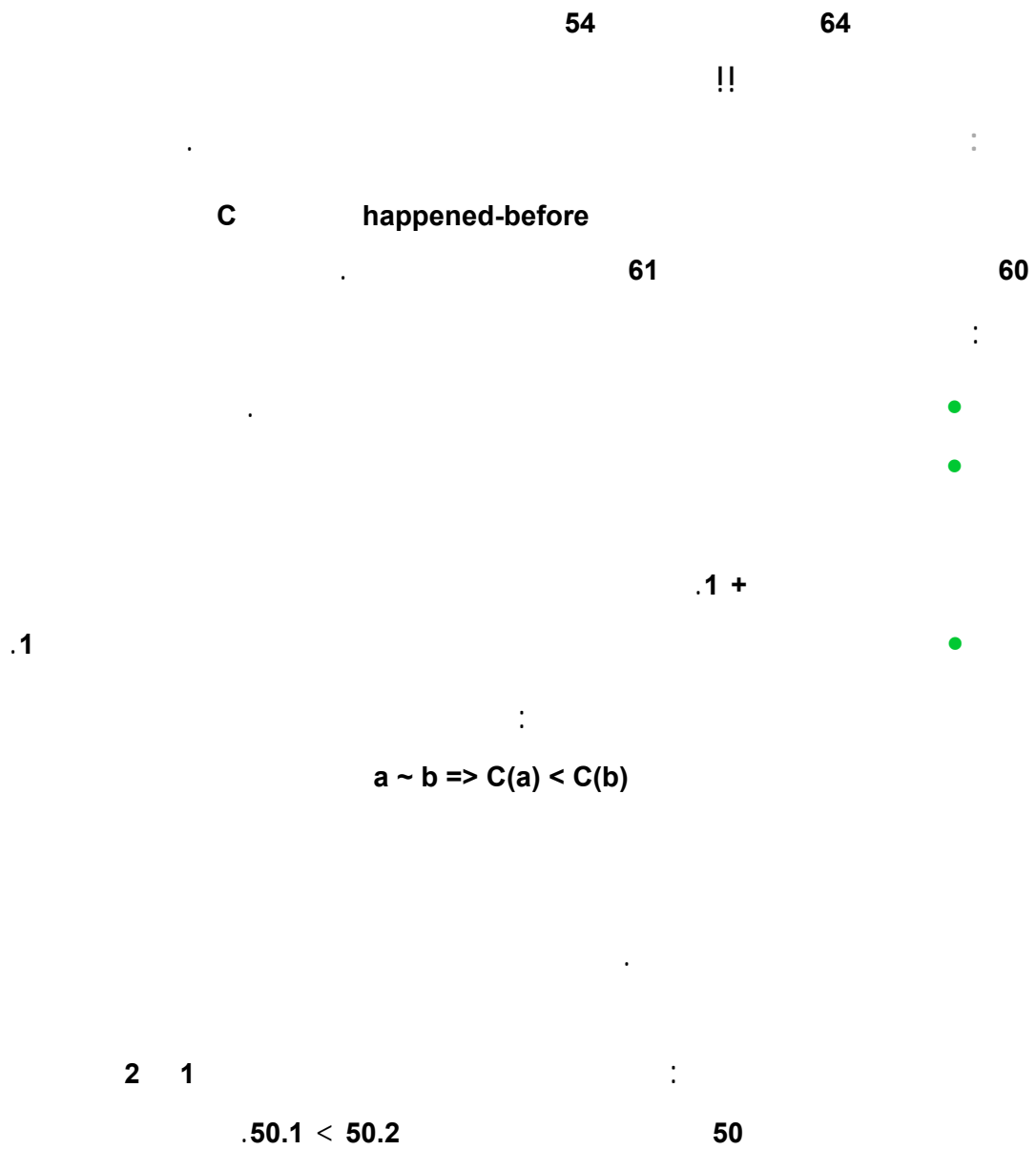
B

D

56

60

C



.3

monitor

:

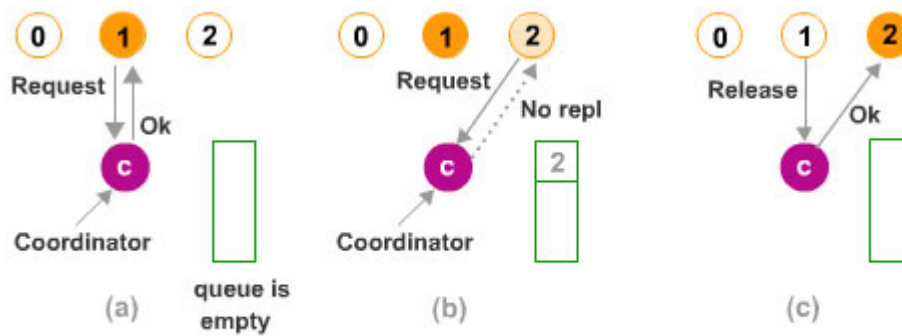
.coordinator

.1

.2

.3

.4



الشكل (3) الخوارزمية المركزية لإدارة المفّطع الحرج

.()

Ricart & Agrawala

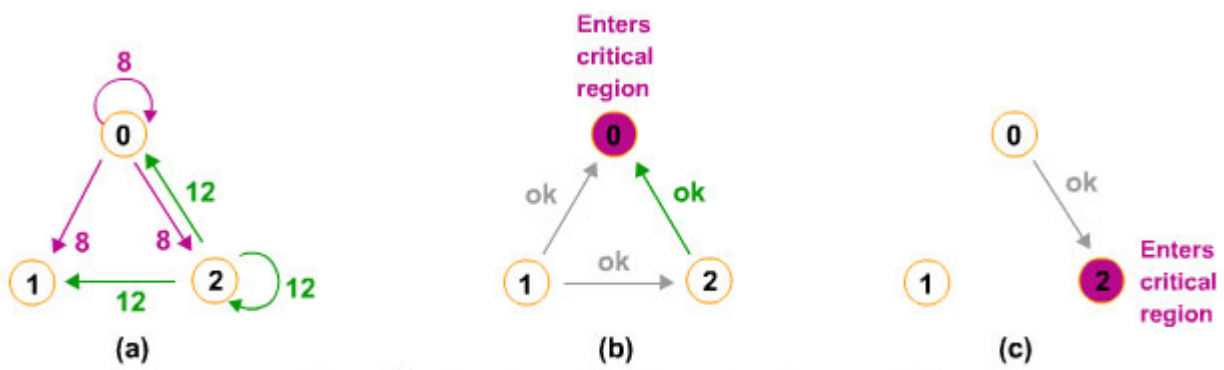
.1

.2

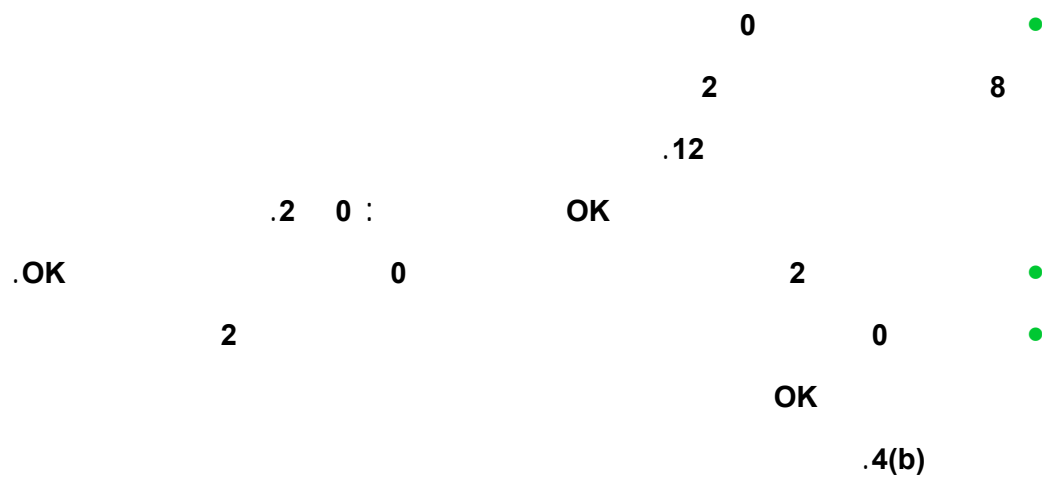
3.

4

4-a



الشكل (4) مثال على عمل خوارزمية ريكار وأغراوالا



2

OK

0



.4(c)



0

2

n

2(n-1)

n

n

n

.2(n-1)

group communication algorithms

$(1 + \dots)$

!! "

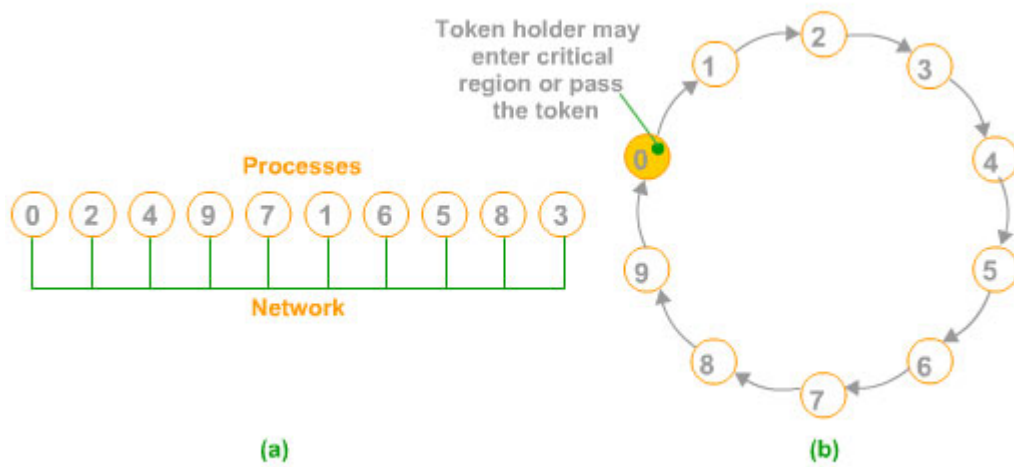
"

bus

token 0

point-to-point

* n-1



الشكل (5) مبدأ الخوارزمية الحلقية

.4

()

.

.

.transactions

:

:

atomic

.

"

"

.

.

:

.1

.

.2

.

)

!!(

.

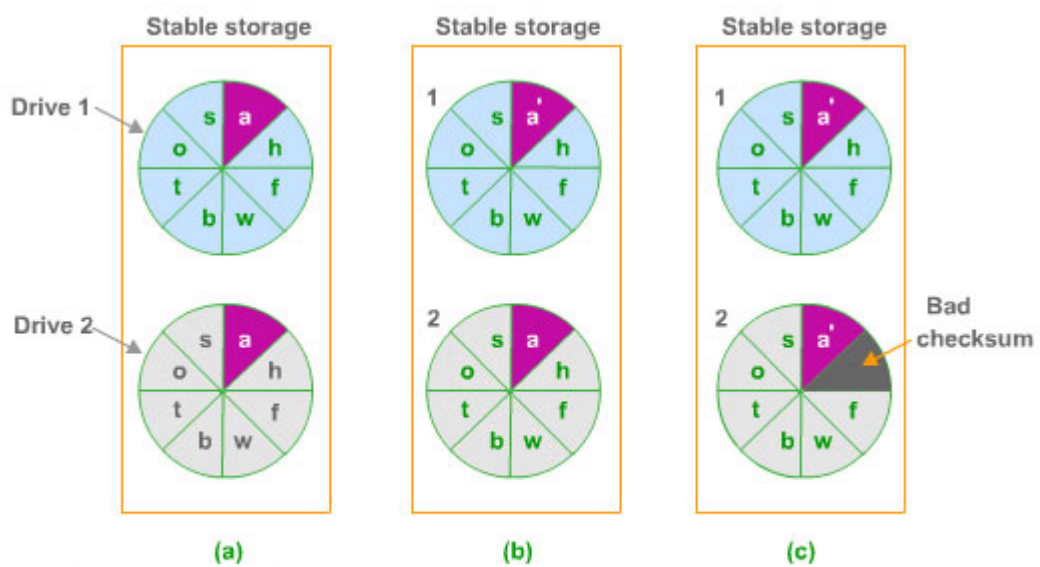


Fig. 3-15. Stable storage Crash after drive 1 is updated Bad spot.

الشكل (6) التخزين المستقر باستخدام قرصين

:

:

:BEGIN_TRANSACTION .a

:END_TRANSACTION .b

:ABORT_TRANSACTION .c

:READ .d

:WRITE .e

:

:(ACID)

:Atmoic .1

:Consistent .2

:Isolated .3

)

7 .(

:Durable .4

BEGIN_TRANSACTION

x=0;

x=x+1;

END_TRANSACTION

(a)

BEGIN_TRANSACTION

x=0;

x=x+2;

END_TRANSACTION

(b)

BEGIN_TRANSACTION

x=0;

x=x+3;

END_TRANSACTION

(c)

Time →

Schedule1	x=0;	x=x+1;	x=0;	x=x+2;	x=0;	x=x+3;	Legal
Schedule2	x=0;	x=0;	x=x+1;	x=x+2;	x=0;	x=x+3;	Legal
Schedule3	x=0;	x=0;	x=x+1;	x=0;	x=x+2;	x=x+3;	Legal

(d)

(d)

الشكل (7) عزل المناقلات والتنفيذ التسلسلي المكافئ

:

.private workspace

.blocks

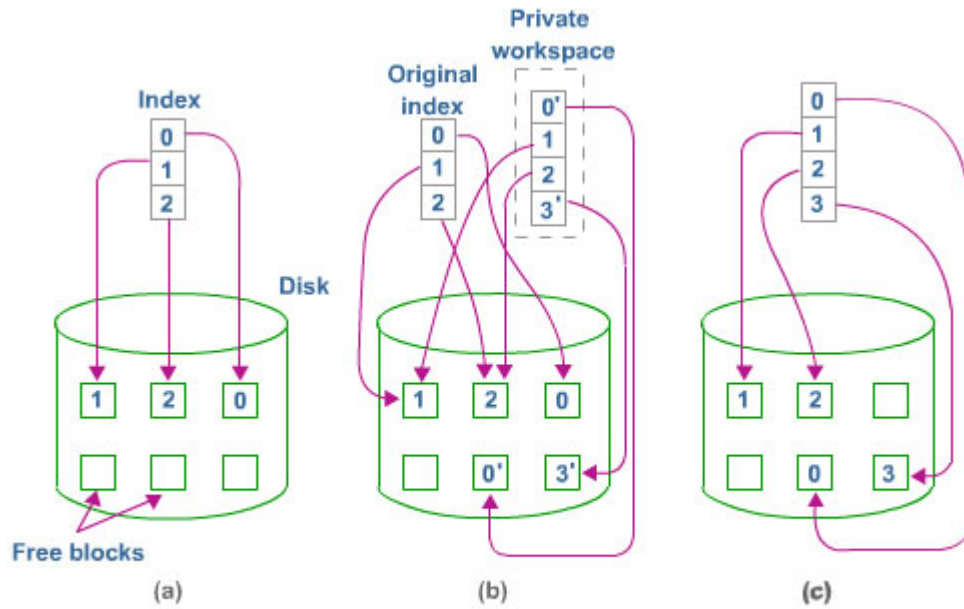
index

.shadow blocks

8-b

.8

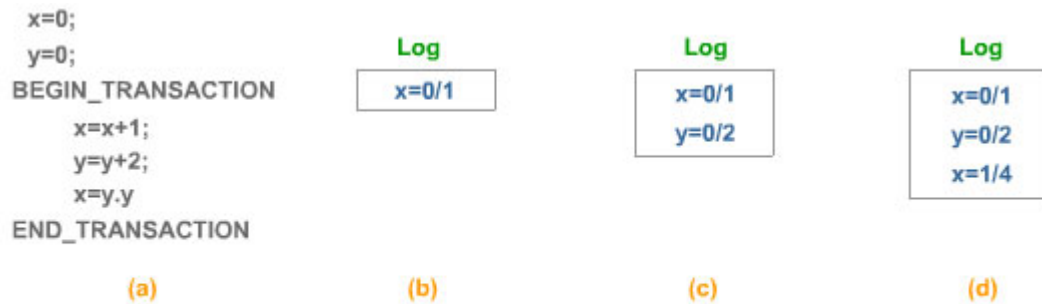
غير



الشكل (8) فضاء العمل الخاص باستخدام وحدات الظل

:

writeahead log



الشكل (9) مثال على عمل السجل الاستباقي

:

9

.0

./

.

.rollback

9-d

x

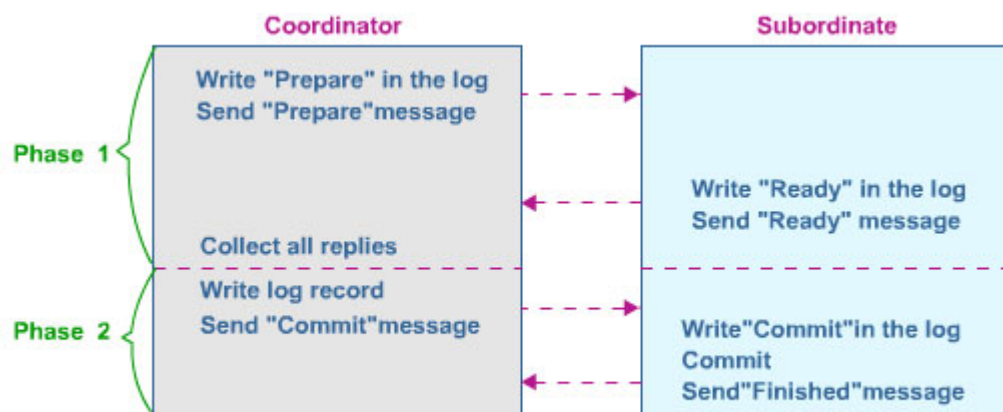
x

.

4

.4

1



الشكل (10) التثبيت على مرحلتين

10

.1

.2

subordinates

.3

.4

()

.5

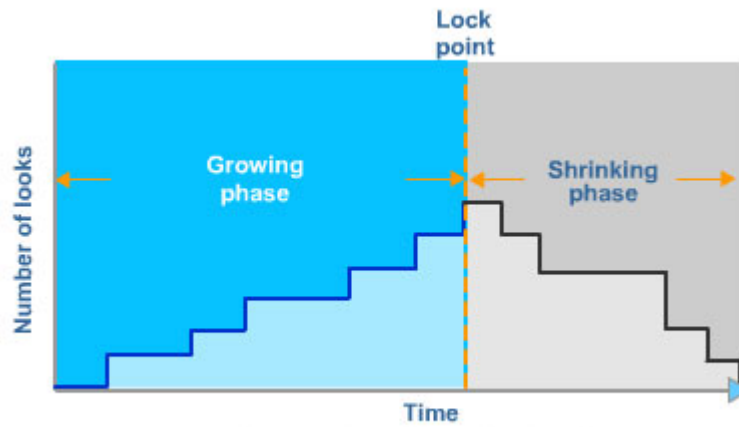
concurrency control algorithm

:locking

.lock manager

.(n-read, 1-write

grain



الشكل (11) الإقفال على مرحلتين

.11

.serializable

.strict two-phase locking

:

•

)

.(cascading aborts

•

implicit

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

)

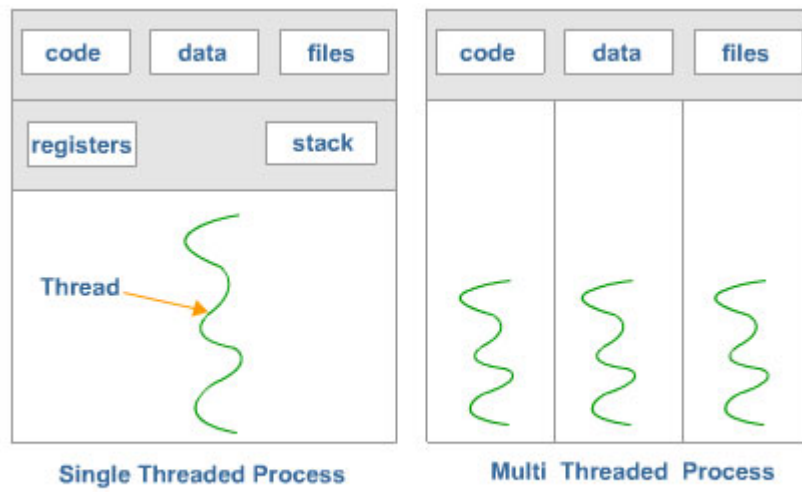
(

.

.1

.2

.threads



الشكل (1) النياسب والإجرائيات

.()

(1)

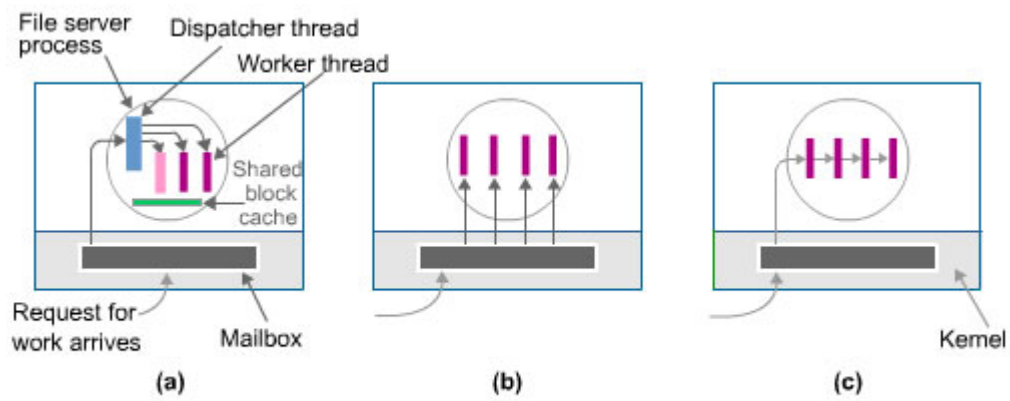
-2-

dispatcher

v :)

worker thread

.(



الشكل (2) استخدام النياسب في المخدم

scheduler

pipeline

c

-2-

b

:team

/

.3

:

:

.workstation

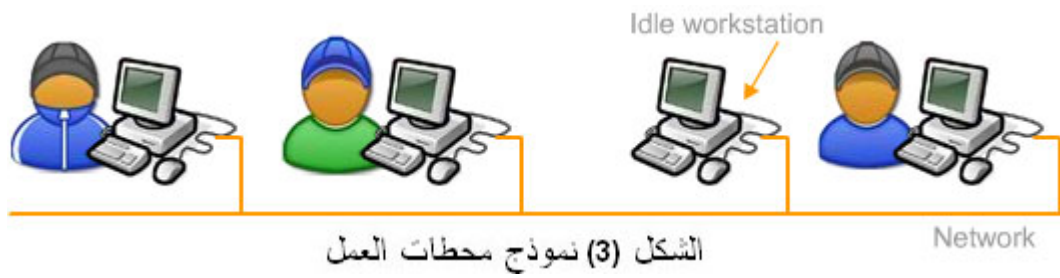
•

.pool

•

:

.3



:

paging

.1

.2

.3

.4

Disk usage	Advantages	Disadvantages
(Diskless)	Low cost, easy hardware and software maintenance, symmetry and flexibility	Heavy network usage: file servers may become bottlenecks
Paging. scratch files	Reduces network load over diskless case	Higher cost due to large number of disks needed
Paging. scratch files. binaries	Reduces network load even more	Higher cost ;additional complexity of updating the binaries
Paging.scratch files,binaries, file caching	Still lower network load;reduces load on file servers as well	Higher cost; cache consistency problems
Full local file system	Hardly any network load; eliminates need for file servers	Loss of transparency

الشكل (4) الأشكال المتخلفة لاستخدام الأقراص الصلبة
في محطات العمل

-4-

:

" "

:

)

.(

multicore

:)

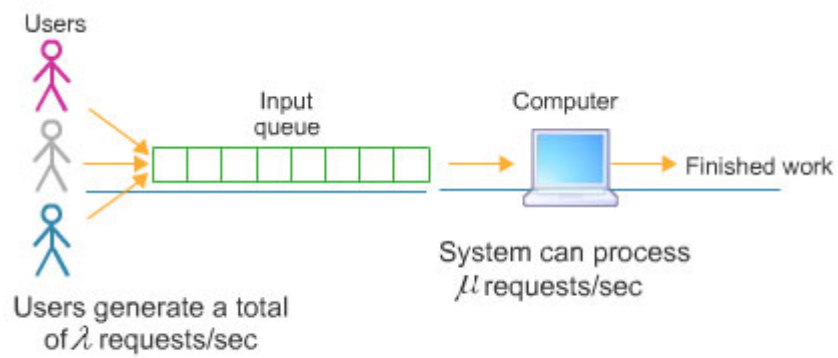
(

terminal

timesharing

:

" "



الشكل (5) النموذج الأساسي لنظرية الأوتال

queuing theory

-5-

)

: (

$$T = \frac{1}{\mu - \lambda}$$

.

:

$$T_1 = \frac{1}{n\mu - n\lambda} = T / n$$

n

n

. n

.

50

.

.

.4

.

.

.

File

File service

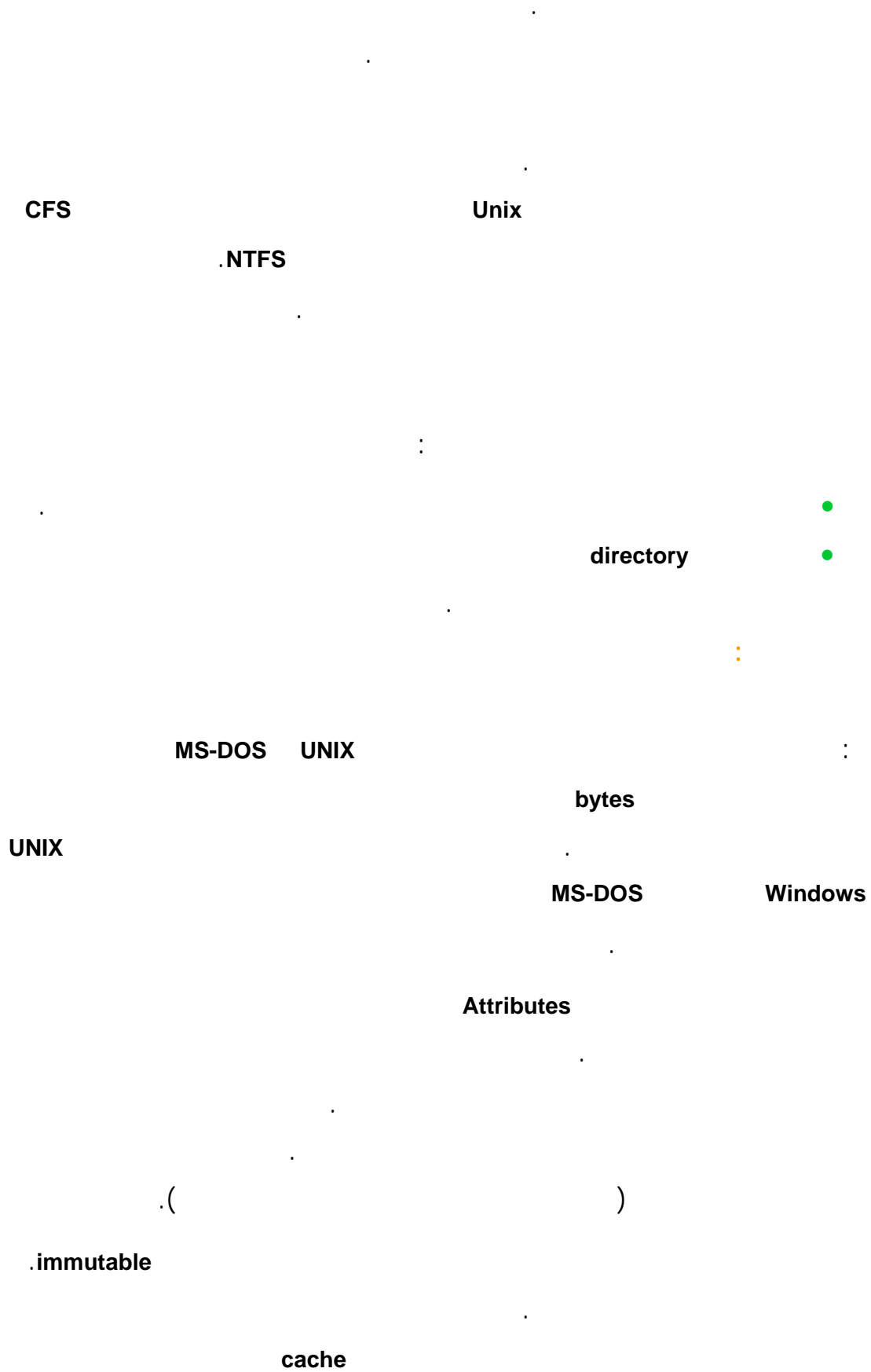
:

.server

.()

.

.



.capabilities

ACL

:

.UNIX

:

:

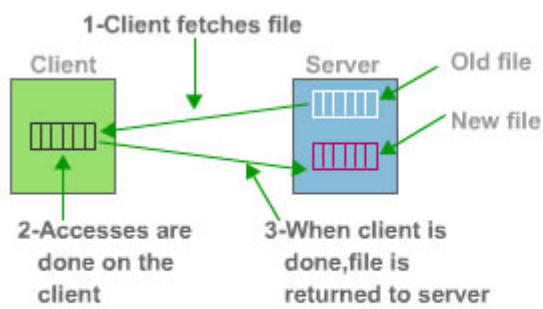
upload/ download

)

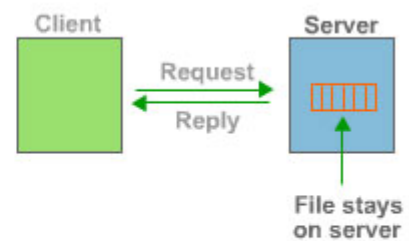
.(

(a) 1

.()



(a)



(b)

الشكل (6) الأنواع المختلفة لخدمات الملفات

.remote access model

(b) 6

procedure.c
$$\left(\begin{array}{c} \text{" " } \\ \cdot \end{array} \right)$$

nothing.txt C

subdirectories

.hierarchical file system

7

.B A

B

D

.B

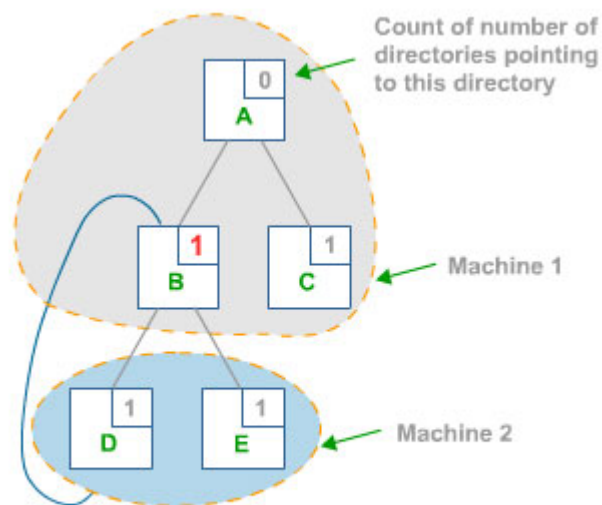
B

1

B A

(E D)

.orphaned



الشكل (7) بيان الاحتواء العشوائي للأدلة

)

(

Naming

a

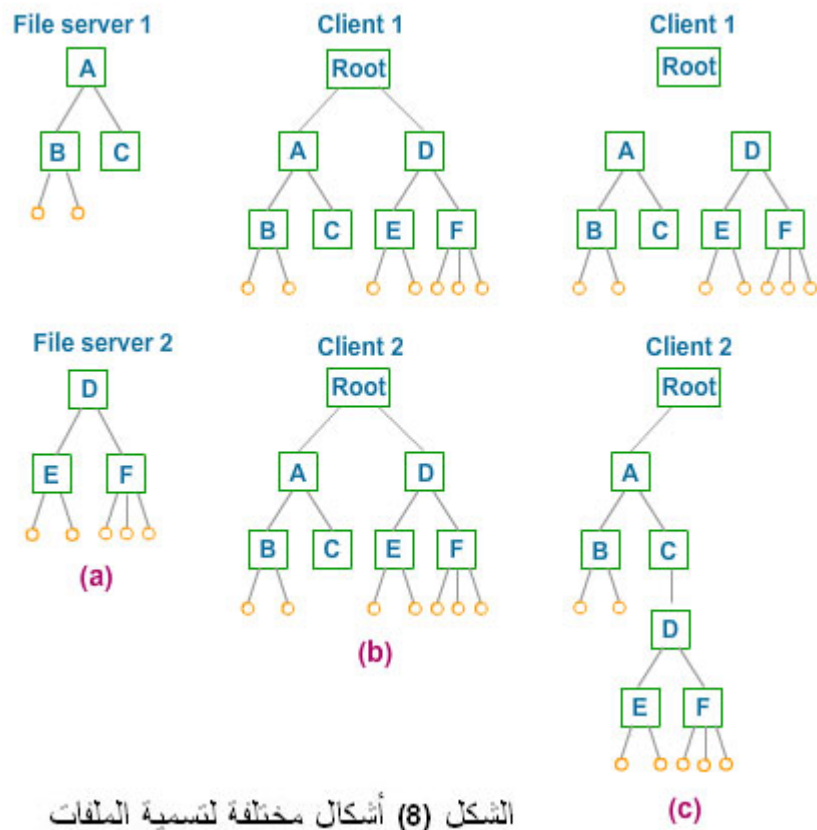
3

/D/E/x

b

c

.remote mounting



الشكل (8) أشكال مختلفة لتسمية الملفات

UNIX

UNIX

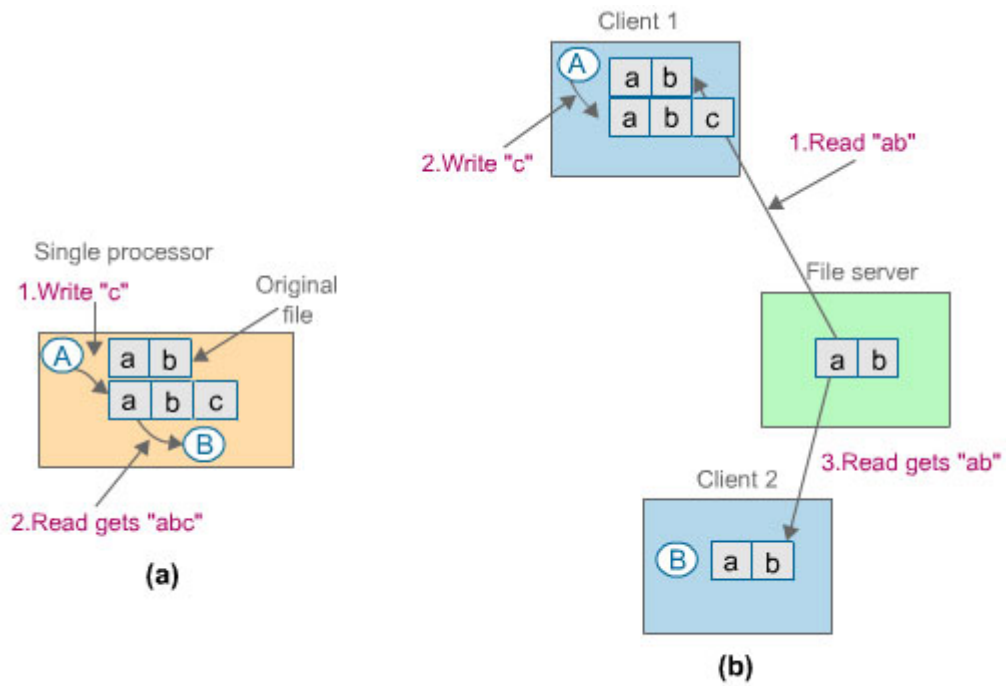
WRITE

READ

9

UNIX semantics

UNIX



الشكل (9) التشارك في ملف ضمن نظام مركزي مثل UNIX

.9-b

:

.

UNIX

session semantics

.semantics

:

.

.CREATE READ

.Immutable files

.

.

.

All-or-nothing

.COMMIT تعلیمة

عملها

Method	Comment
UNIX semantics	Every operation on file is instantly visible to all processes
Session semantics	No changes are visible to other processes until the file is closed
Immutable file	No updates are possible; simplifies sharing and replication
Transactions	All changes have the all-or-nothing property

A. Tanenbaum, Introduction to distributed systems.

J. Wu, Distributed systems design

Coulouris - Distributed Systems - Concepts and Design 3e.

A. Tanenbaum, Modern Operating systems, 2nd Edition.

Ajay D. Kshemkalyani, Mukesh Singhal,

Distributed Computing Principles, Algorithms, and Systems.