# سیستم های توزیع شده

**Reference:**

DISTRIBUTED SYSTEMS Principles and Paradigms Second Edition, ANDREW S. TANENBAUM and MAARTEN VAN STEEN, 2007 Prentice-Hall.

**مدرس: دکتر ابراهیم بهروزیان نژاد**

**فهرست مطالب:**

## فصل اول: مقدمه ای بر سیستم های توزیع شده

**تعریف سیستم توزیع شده**: مجموعه ای از کامپیوترهای مستقل و ناهمگن که برای کاربرش بصورت یک سیستم منسجم و یکپارچه ظاهر می شود.

### مقدمه

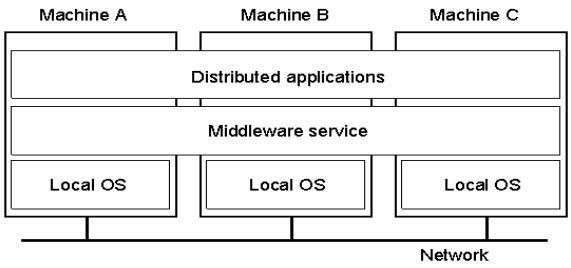
- سیستم عامل توزیع شده در یک محیط شبکه ای اجرا می شود.
- سیستم عامل های توزیع شده به مراتب پیچیده تر از سیستم عامل های شبکه هستند.
- به سیستم های توزیع شده گاهی اوقات سیستم های با ارتباط ضعیف نیز می گویند، چرا که هر پردازنده سیگنال ساعت و حافظه مستقلی دارد.
- به سیستم های چندپردازنده ای سیستم های با ارتباط قوی می گویند، چرا که پردازنده ها سیگنال ساعت یکسان دارند.
- یکی از خصوصیات مهم سیستم های توزیع شده که از دید کاربران مخفی است، تفاوت کامپیوتر های مختلف و روشهایی است که از طریق آنها این کامپیوتر ها با هم ارتباط برقرار می کنند.
- گسترش سیستم های توزیع شده نسبتاً آسان است.
- اگر قسمتهای خاصی از آن بطور موقتی خراب هم باشند معمولاً بصورت کامل در دسترس است. البته کاربران نمی بایستی متوجه تعویض و یا تعمیر آن قسمت یا اضافه کردن بخش های جدیدی شوندکه به منظور خدمات رسانی بیشتر به کاربران و برنامه های کاربردی صورت می گیرد.
- در سیستم توزیع شده اگر اطلاعاتی همزمان در چند کامپیوتر به صورت یکسان ذخیره شود و یکی از کامپیوترها خراب شود، اطلاعات را می توان از کامپیوتر های دیگر بازیابی کرد و از این نظر قابلیت اطمینان افزایش می یابد.
- یکی از مزایای مهم سیستم توزیع شده سرعت بالای اجرای برنامه هاست چرا که یک برنامه همزمان می تواند از چندین کامپیوتر برای اجرا شدنش استفاده کند.

### معایب سیستم توزیع شده در مقابل چندپردازنده ای

- امنیت پایین است.
- سرعت آن ممکن است پایین تر باشد.
- وقتی که تعداد کامپیوتر ها کم باشد سیستم چند پردازنده ای بهتر از سیستم توزیع شده است.

## اجزاء سیستم توزیع شده

- سخت افزار سیستم توزیع شده: شبکه
- نرم افزار سیستم توزیع شده: میان افزار (Middleware)




Distributed Applications : مثل IE و ... که برای ارتباط با سیستم توزیع شده مورد استفاده قرار می گیرد.

Middleware Service : نوعی نرم افزار است که درحقیقت اساس کار پیاده سازی سیستم‌های توزیع‌شده را بر عهده داشته و بر روی تمامی کامپیوترها قرار می گیرد.

## اهداف سیستم توزیع شده

۱- دسترسی به منابع (Making resource accessible)

۲- شفافیت (Transparency)

۳- باز بودن (Openness)

۴- مقیاس پذیری (Scalability)

### ۱- دسترسی به منابع

- هدف سیستم توزیع شده این است که کاربران به راحتی به منابع راه دور دسترسی داشته باشند و آنها را به روش کنترل شده به اشتراک بگذارند. دسترسی به منابع، ارتباط و تبادل اطلاعات را تسهیل می کند.

### ۲- شفافیت

- هدف عمده یک سیستم توزیع شده پنهان کردن این حقیقت است که پردازنده ها و منابع آن بصورت فیزیکی در کامپیوترهای متعدد توزیع شده اند.
- به سیستم توزیع شده ای که بتواند خود را برای کاربر و برنامه های کاربردی اش، طوری نمایش دهد که گویی سیستم تک کامپیوتری است، سیستم شفاف می گویند.

**انواع شفافیت**

- شفافیت می تواند در موارد گوناگونی مطرح شود:

۱) دسترسی (Access): تفاوت در نمایش‌ها و چگونگی دستیابی به منابع را از دید کاربران مخفی می‌کند.
۲) مکان (Location): مکان منابع را از دید کاربران مخفی می‌کند (مثلاً از طریق DNS).
۳) مهاجرت (Migration): انتقال منابع از مکانی به مکان دیگر را از دید کاربران مخفی می‌کند.
۴) جابجایی (Relocation): انتقال منابع در حال استفاده از مکانی به مکان دیگر را از دید کاربران مخفی می‌کند (مثل سیستم موبایل).
۵) تکرار (Replication): تکرار منابع را از دید کاربران مخفی می‌کند.
۶) خرابی (Failure): خرابی و ترمیم منابع را از دید کاربران مخفی می‌کند.
۷) همزمانی (Concurrency): استفاده همزمان منابع داده توسط چند کاربر را مخفی می‌کند.

**درجه شفافیت**

- با افزایش میزان شفافیت ممکن است کارایی سیستم کاهش یابد (شفافیت با کارایی سیستم در تضاد است)
- باید بین درجه شفافیت و کارایی یک مصالحه باشد. مثال: سازگاری چندین نسخه

**۳- باز بودن (Openness)**

- اینکه برروی هر سیستم عامل با هر سخت افزاری و با داشتن یک واسط مشترک (مانند Java Virtual Machine) بتوان برنامه را اجرا نمود این خاصیت قابلیت حمل و تعامل را بالا می‌برد.
- استفاده از یک زبان واسطه میانی برای نوشتن برنامه‌ها روی تمامی کامپیوترهای سیستم توزیع شده Interface Definition Language (IDL)

**۴- مقیاس پذیری**

- اینکه بتوان یک سیستم را به راحتی گسترش داد (چندکامپیوتربه آن اضافه نمود).
- مقیاس‌پذیری ( که در مقابل متمرکز بودن استفاده می‌شود) را حداقل از سه جنبه می‌توان بررسی کرد:( اندازه و جغرافیا و مدیریت )

**جنبه های مقیاس پذیری**

- اندازه: به آسانی بتوان کاربران و منابع را به سیستم اضافه کرد.
- جغرافیا: سیستمی که در آن، کاربران و منابع بتوانند از هم دور باشند.
- مدیریت: با وجود مدیریت‌های مستقل، مدیریت آن آسان باشد.

## مقیاس پذیری و توزیع شدگی

- سرویسهای متمرکز قابلیت توسعه ندارند.
- در حالیکه در سیستم توزیع شده توسعه می تواند بر روی سه دسته سرویس صورت گیرد:
  - داده - سرویس- الگوریتم
  - براین اساس می توان داده های توزیع شده، سرویسهای توزیع شده و الگوریتم های توزیع شده داشت.
- داده متمرکز: همه داده‌ها را یکجا ذخیره می‌کند.
- داده غیرمتمرکز: داده‌ها در مکانهای مختلف ذخیره می‌شود.
- سرویس متمرکز: تنها یک سرویس‌دهنده برای تمام کاربران.
- سرویس غیرمتمرکز: چندین سرویس‌دهنده برای تمام کاربران.
- الگوریتم‌های متمرکز.
- الگوریتم‌های غیرمتمرکز.

## ویژگیهای الگوریتم‌های غیر متمرکز

۱. هیچ ماشینی اطلاعات کاملی راجع به وضعیت سیستم ندارد.
۲. ماشین‌ها تنها براساس اطلاعات محلی تصمیم می‌گیرند.
۳. خرابی در یک ماشین، به الگوریتم آسیب نمی‌رساند.
۴. هیچ تصوری راجع به ساعت سراسری وجود ندارد.

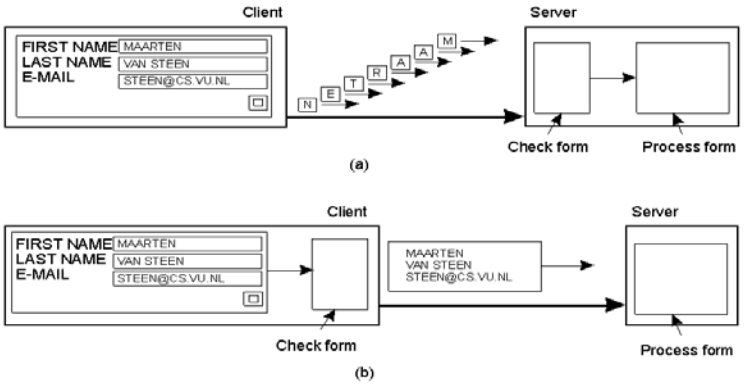
اگر سیستمی با جزئیات متمرکز شده زیادی باشد، روشن است که قابلیت ارزیابی جغرافیایی، بخاطر مسائل قابلیت اجرا و اطمینان پذیری که از ارتباط سطح وسیع ناشی می شود، محدود خواهد شد.

## تکنیک‌های مقیاس‌پذیری

۱- مخفی کردن تاخیرهای ارتباطی: برنامه کاربردی باید طوری ساخته شود که فقط از ارتباطات ناهمگام استفاده کند.
۲- توزیع: شامل شکستن یک قطعه برنامه یا داده و پخش آن در سیستم است.
۳- تکثیر: قطعات برنامه و داده در سیستم توزیع شده تکثیر شوند.

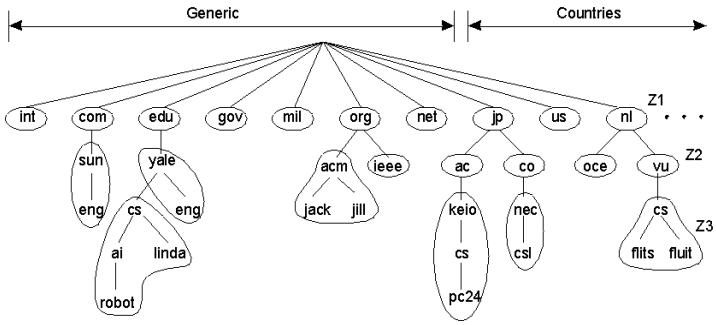
## مخفی کردن تاخیرهای ارتباطی

تفاوت بین "دادن امکان چک فرم" به یک سرور (a) و یا یک کلاینت (b) در حالی که کاربر مشغول پرکردن فرم است.




## توزیع

- سیستم نامگذاری DNS: نامها بطور سلسله مراتبی، به درختی از دامنه‌ها سازماندهی شده است که به چند منطقه Zone تقسیم می‌شود.




## تکثیر

- تکثیر نه تنها قابلیت دسترسی را افزایش می‌دهد بلکه به توزیع متوازن بار بین سیستمها کمک می‌کند تا کارایی بهبود یابد.
- مثال: استفاده از حافظه پنهان
- از آنجایی که کپی های متعددی از یک منبع وجود دارد، تغییر یک کپی باعث می شود تا با بقیه فرق کند. درنتیجه تکثیر منجر به مسائل سازگاری می شود.

**Pitfalls when Developing Distributed Systems**

**False assumptions made by first time developer:**

- The network is reliable.
- The network is secure.
- The network is homogeneous.
- The topology does not change.
- Latency is zero.
- Bandwidth is infinite.
- Transport cost is zero.
- There is one administrator.

## انواع سیستم‌های توزیع شده

۱- سیستم محاسبات توزیع شده (Distributed Computing System)

- Cluster computing systems
- Grid computing systems

۲- سیستم اطلاعات توزیع شده (Distributed Information System)

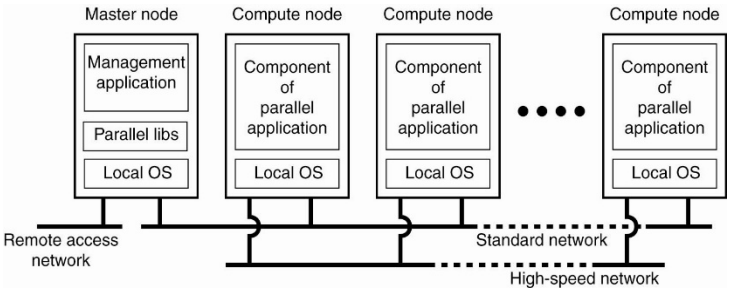
- Transaction Processing Systems
- Enterprise Application Integration (Exchange info via RPC or RMI)

۳- سیستم تعبیه شده/فراگیر توزیع شده (Distributed Pervasive (Ubiquities)/Embedded System)

- Home Systems (e.g. Smart phones, PDAs)
- Electronic Health care systems (Heart monitors, BAN: Body Area Networks)
- Sensor Networks (distributed Databases connected wirelessly)

## مشخصات یک Cluster Computing System

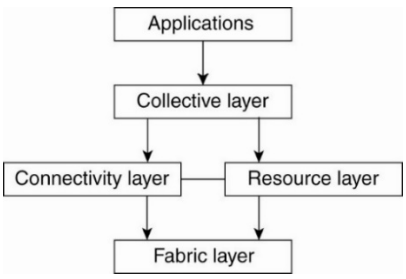
- هدف سیستمهای محاسباتی توزیع شده اجرای برنامه‌های محاسباتی با کارایی بالا بصورت توزیع شده است.
- در Cluster computing systems سخت‌افزار شامل مجموعه‌ای از PCهای معمولا همگن با سیستم عاملهای یکسان است که از طریق شبکه محلی پرسرعت به هم متصل می‌شود (ایجاد یک Supercomputer )




## سیستم‌های محاسباتی توزیع شده: گرید

- مجموعه‌ای از کامپیوترهای ناهمگن ( از نظر سخت افزار، سیستم عامل، شبکه و امنیت) که مهمترین هدف این نوع سیستم قراردادن آسان منابع در اختیار برنامه‌های کاربردی (کاربران سازمانهای مجازی .Virtual Organization V.O) به صورت استفاده اشتراکی و همکاری است.
- برای کارهای محاسباتی بهتر است از Cluster استفاده شود.
- Scalability در گرید بالاتر است.

## مثالی از معماری لایه ای گرید



**Grid Computing Layers**

**Collective layer**: access to multiple resources and typically consists of services for resource discovery, allocation and scheduling.

**Connectivity layer**: transfer data between resources or access a resource from a remote location

**Resource layer**: managing a single recourse such as creating a process or reading data

**Fabric layer**: provides interface to local resources at a specific site within a V.O.

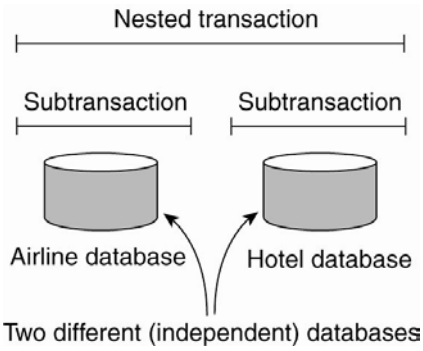
**سیستم پردازش تراکنش توزیع شده**

ایده مهم این است که یا تمام درخواست‌ها (دستورات) انجام می‌شود یا هیچکدام انجام نمی‌شود (ACID).

۱- تجزیه ناپذیری (Atomic) ۲- سازگاری (Consistent)

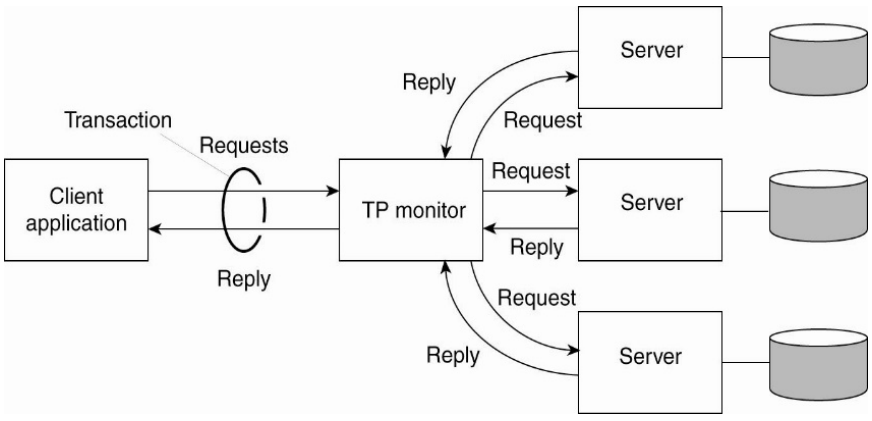
۳- ایزوله شدن (Isolated) ۴- پایداری (Durable)

**حداقل دستوراتی که یک تراکنش باید داشته باشد :** Begin-Transaction، Read، Write،
End-Transaction = Commit, Abort



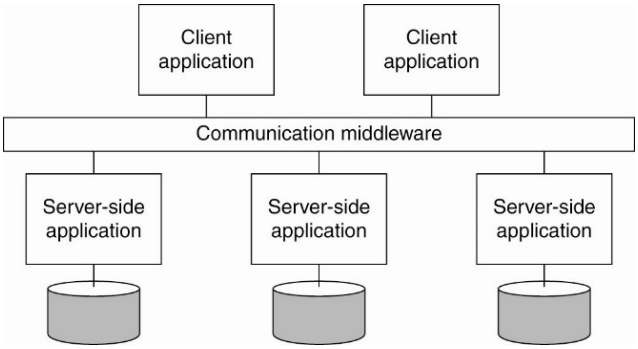

A nested transaction

**سیستم پردازش تراکنش توزیع شده**




The role of a Transaction Processing (TP) monitor in distributed systems

## Enterprise application integration (1)




Middleware as a communication facilitator in enterprise application integration

### Enterprise application integration

بسته های middleware و پروتکلهای متفاوتی برای پشتیبانی از Enterprise applications استفاده می شوند نظیر:

CORBA (Common Object Request Broker Architecture)
DCOM (Distributed Component Object Management)
RPC (Remote Procedure Call)
RMI (Remote Method Invocation)

### سیستم تعبیه شده/فراگیر توزیع شده

این سیستمها معمولاً: کوچک و قابل حمل هستند، محاسبات کمی دارند، منبع تغذیه برای آنها مهم است و با امواج رادیویی و بصورت بیسیم کار می کنند.

مثال ۱: Electronic Health Care System

مثال ۲: Wireless Sensor Network

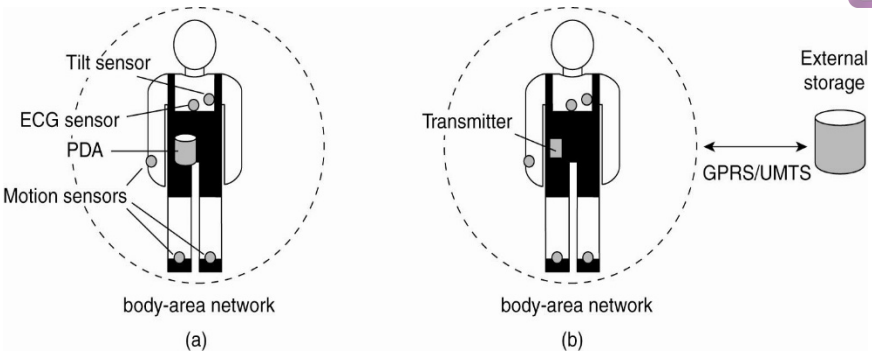
### نیازهای یک سیستم فراگیر

- Embrace contextual changes (i.e. I was a phone now I am a web access device. A device must continuously be aware of the fact that its environment may change)
- Encourage ad hoc composition (used differently by different users, e.g. PDA)
- Recognize sharing as the default (easily read, store, manage, and share info)

### Electronic Health Care Systems

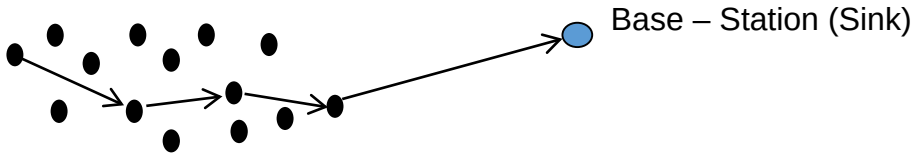
- Questions to be addressed for health care systems:
- Where and how should monitored data be stored?
- How can we prevent loss of crucial data?
- What infrastructure is needed to generate and propagate alerts?
- How can physicians provide online feedback?
- How can extreme robustness of the monitoring system be realized?
- What are the security issues and how can the proper policies be enforced?

## Electronic Health Care Systems




Monitoring a person in a pervasive electronic health care system, using (a) a local hub or (b) a continuous wireless connection.

## Wireless Sensor Network




- داده حس شده توسط حسگر به چهار روش ارسال می شود:

۱- Continues داده حس شده همواره ارسال می شود.

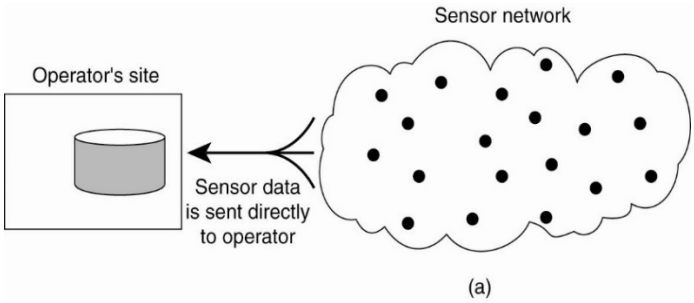
۲- Query Based هر وقتی که نیاز باشد داده حس شده ارسال می شود.

۳- Event Based وقتی عمل خاصی رخ داد داده حس شده ارسال شود.

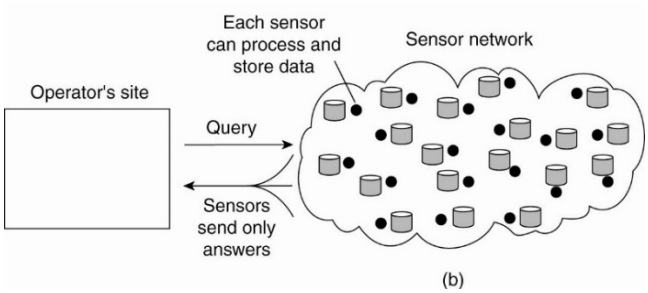
۴- ترکیبی

## Wireless Sensor Network

- Questions concerning sensor networks:
  - How do we (dynamically) set up an efficient tree in a sensor network?
  - How does aggregation of results take place? Can it be controlled?
  - What happens when network links fail?




Organizing a sensor network database, while storing and processing data (a) only at the operator's site or …



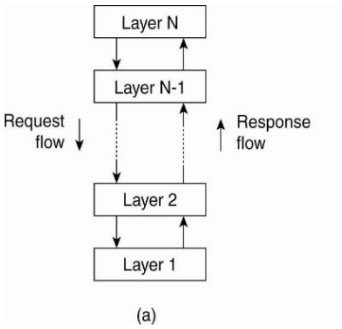

Organizing a sensor network database, while storing and processing data … or (b) only at the sensors

# فصل دوم: معماری‌ها

## سبک های معماری سیستم توزیع شده

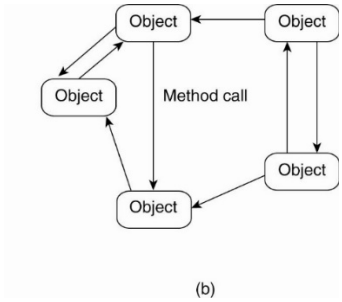
۱- معماری لایه ای (Layered architectures)
۲- معماری مبتنی بر شیء (Object-based architectures)
۳- معماری داده محور (Data-centered architectures)
۴-معماری مبتنی بر رویداد (Event-based architectures)

### ۱- معماری لایه ای



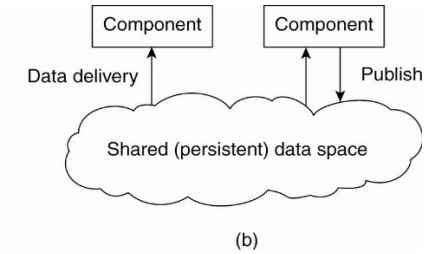

- اجزاء در لایه هایی قرار دارند و یک جز در لایه i تنها حق دارد که اجزاء لایه i-1 را فراخوانی کند.
- شماره گذاری لایه ها معمولاً از پایین به بالا است.
- درخواست ها به سمت پایین حرکت می کنند.
- نتایج درخواست ها به سمت بالا حرکت می کنند.

### ۲- معماری مبتنی بر شیء



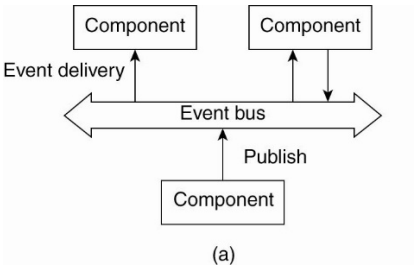

- معماری مبتنی بر شیء سازمان بندی ضعیف تری نسبت به معماری لایه ای دارد.
- اشیاء متناظر اجزاء در سیستم پراکنده هستند.
- این اشیاء از طریق فراخوانی رویه که ممکن است از راه دور نیز باشد، با هم ارتباط برقرار می کنند.

### ۳-معماری داده محور




- فرآیند ها از طریق یک فضای داده مشترک با هم ارتباط برقرار می کنند.
- مثال: سیستم توزیع شده مبتنی بر وب و یا Share distributed file systems

### ۴-معماری مبتنی بر رویداد




• Communication via propagation of events
• Mostly Publish/Subscribe
• E.g. register interest in market info

- فرآیند ها از طریق پخش رویدادها با هم ارتباط برقرار می کنند و هر رویداد تعدادی عضو دارد.
- به عبارت دیگر تعدادی فرآیند، عضو رویداد هستند.
- اگر یکی از فرآیندها رویداد را منتشر کند، میان افزار تضمین می کند که تنها اعضای آن رویداد آن را دریافت نمایند.
- لازم نیست تا فرآیند ها صریحاً به هم مراجعه کنند. اصطلاحاً گفته میشود که فرآیندها اتصال ضعیف دارند.

### انواع معماری (سازمان) سیستم توزیع شده

۱- متمرکز (Client Server)

Application Layering
Multi-tiered Architectures

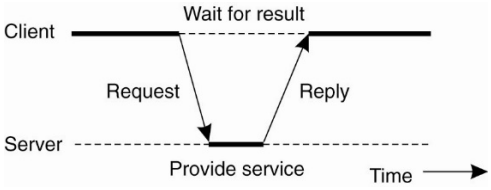
۲- غیر متمرکز

Structured P2P (Peer-to-Peer) Architectures
Unstructured P2P Architectures
Topology Management of Overlay Networks
Super peers

۳- ترکیبی

Edge-Server Systems
Collaborative Distributed Systems

## معماری متمرکز (Client Server)




- تعامل کلی بین سرویس گیرنده و سرویس دهنده
- یکی از مهمترین قسمتهای سیستم توزیع شده ارتباط بین آنها است.
- Client: همیشه نیاز به سرویس یا اطلاعات دارد.
- Server: سرویس یا اطلاعات را در اختیار دارد و اگر کسی بخواهد آنها را در اختیار آن قرار می دهد.
- در صورتی که شبکه ارتباطی قابل اطمینان و احتمال بروز خطا یا گم شدن بسته ها ناچیز باشد از یک ارتباط بدون اتصال استفاده میشود.
- در این روش اگر بسته درخواست یا پاسخ گم شود، سرویس گیرنده ممکن است تا ابد منتظر بماند و راه حل آن استفاده از Time Out است.
- دستورات تکرار شدنی (Idempotent): برخی از دستوراتی هستند که حتی اگر چندین بار تکرار شوند اثرات مخربی ندارند.
- دستورات تکرار ناشدنی (non-idempotent): برخی از دستوراتی هستند که اگر تکرار شوند اثرات مخربی دارند.
- در شبکه های با قابلیت اطمینان پایین تر، از ارتباط اتصال گرا (Connection Oriented) استفاده می شود.
- کارایی پایین تری دارد (بدلیل اعمال اضافی).
- هزینه های بالاتری دارد.
- قابل اطمینان ارسال داده بیشتر است.
- برای مثال می توان از پروتکل TCP نام برد.

### Application Layering

۱- سطح واسط کاربر (User Interface)

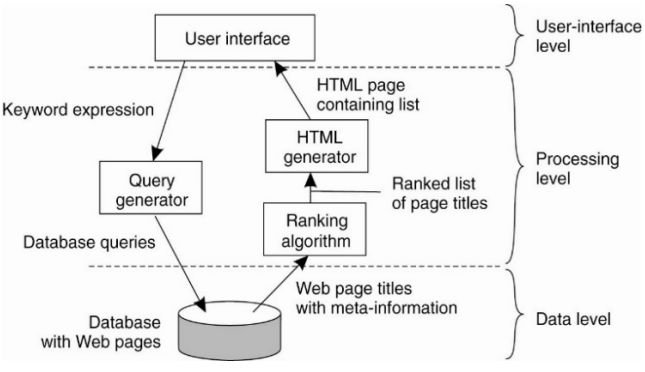
- معمولاً توسط سرویس گیرندگان پیاده سازی می شود و کارش ارتباط با کاربر است.

۲- سطح پردازش (Processing)

- سطح میانی است که عملیات اصلی برنامه کاربردی در آن انجام می پذیرد.

۳- سطح داده (Data)

- شامل برنامه هایی است که داده های واقعی را نگهداری می کنند. معمولاً این داده ها پایدار هستند.




سازمان ساده شده ای از موتور جست و جوی اینترنت در سه لایه

## معماری چندبخشی Multitiered

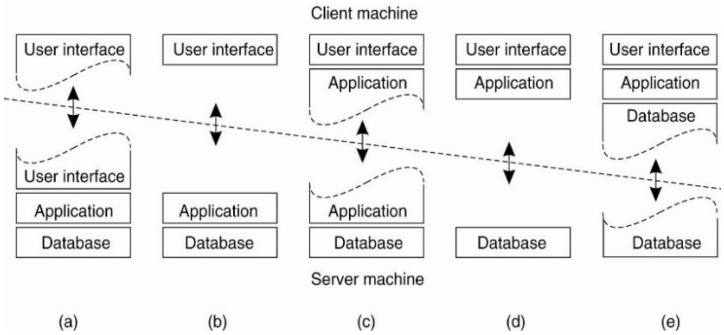
- بخشی از سطوح واسط کاربر، پردازش، داده در هر بخش قرار می گیرد.
- به توزیع عمودی معروف است.
- ساده ترین سازمان فقط دو نوع ماشین دارد (2 Tier Architecture)

۱- ماشین سرویس گیرنده شامل برنامه ای است که سطح واسط کاربر را پیاده سازی می کند.

۲- ماشین سرویس دهنده که شامل بقیه است.

- سازمان دیگر سه نوع ماشین دارد (3 Tier Architecture)

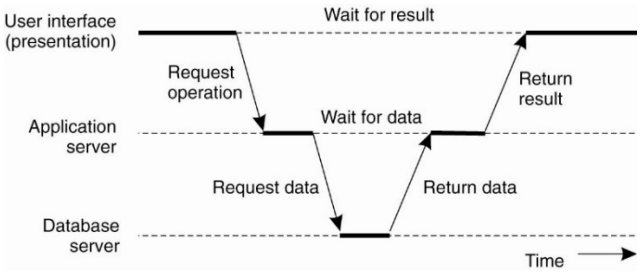
### 2 Tier Architecture




Client — Fat
Client — Thin

Alternative client-server organizations (a)–(e).

### 3 Tier Architecture

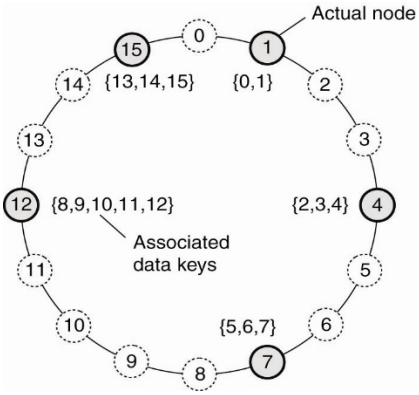



An example of a server acting as client

## معماری غیرمتمرکز

- به توزیع افقی معروف است.
- می توان چندین سرویس دهنده داشت.
- سرویس دهنده‌ها یکسان هستند.
- بار کاری به صورت متعادل در سیستم پخش می شود.
- یکی از مهمترین انواع توزیع افقی، سیستم های نظیر به نظیر (peer-to-peer) نامیده می شوند.
- انواع سیستم های نظیر به نظیر
  - ساخت یافته (Structured)
  - غیرساخت یافته (Unstructured)
- تعریف Overlay Network: یک شبکه از فرایندهای روی ماشینها صرف نظر از توپولوژی یا ساختار و ارتباط بین آنها

## معماری نظیر به نظیر ساخت یافته

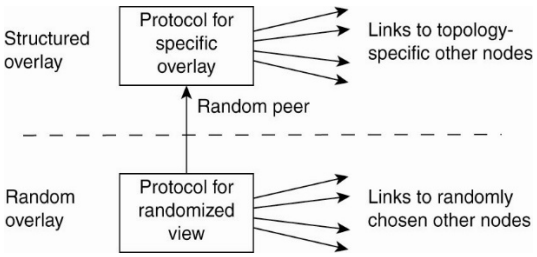



- هر فرآیند فقط فرآیند کناری خود را می شناسد.
- برای ورود یک گره (فرآیند) جدید به حلقه یک عدد تصادفی تولید می شود.
- برای خروج یک گره مانند id از حلقه، این گره رفتن خود را به گره بعدی و گره پیشین خود خبر می دهد و اقلام داده در اختیار خود را به گره بعدیش ارسال می کند.

## معماری نظیر به نظیر غیرساخت یافته

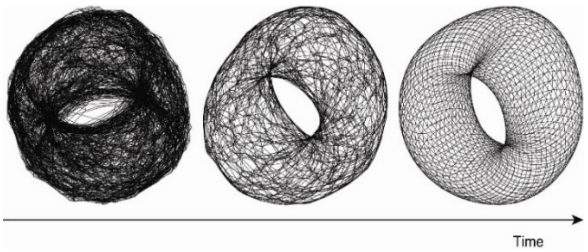
- فرآیندها به ترتیب منطقی قرار گرفته نشده اند و بصورت تصادفی هستند.
- تولید شبکه های همپوشانی به الگوریتم های تصادفی وابسته است.
- معمولاً در هر گره (فرآیند) لیستی از همسایگان (دیگر فرآیندها) نگهداری می شود.
- اقلام داده به صورت تصادفی در گره‌ها قرار می گیرند.

### Topology Management of Overlay Networks




روش دو لایه ای برای ساخت و نگهداری توپولوژی‌های خاص با استفاده از تکنیک هایی از سیستم های نظیر به نظیر غیر ساخت یافته.

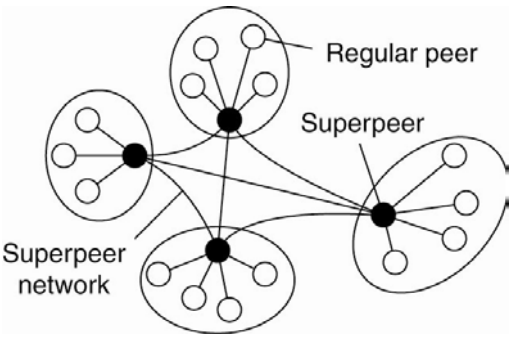
### Topology Management of Overlay Networks




تولید شبکه همپوشانی خاص با استفاده از سیستم نظیر به نظیر غیر ساخت یافته دو لایه ای.

**Super peers**

- یکی از مشکلات روش غیر ساخت یافته، جستجوی اقلام است و برای جستجو باید پیام جستجو به همه فرایندها ارسال شود.
- برای بالا بردن کارایی، ساختار متقارن کمی به هم می خورد و از برخی گره‌ها برای ذخیره سازی داده هایی خاص، مانند شاخص اقلام داده استفاده می شود. به این گره ها Super peers گفته می شود.
- Super peers با هم ارتباط برقرار می کنند و باعث تسریع در رد و بدل کردن داده ها میشوند.

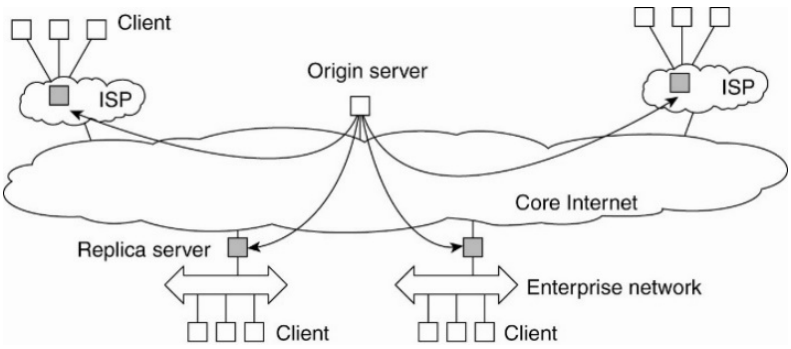



سازمان سلسله مراتبی گره ها در شبکه سوپر همتا

## معماری ترکیبی

- با ترکیب معماری متمرکز و غیرمتمرکز می توان به یک معماری ترکیبی رسید.

**Edge-Server Systems**

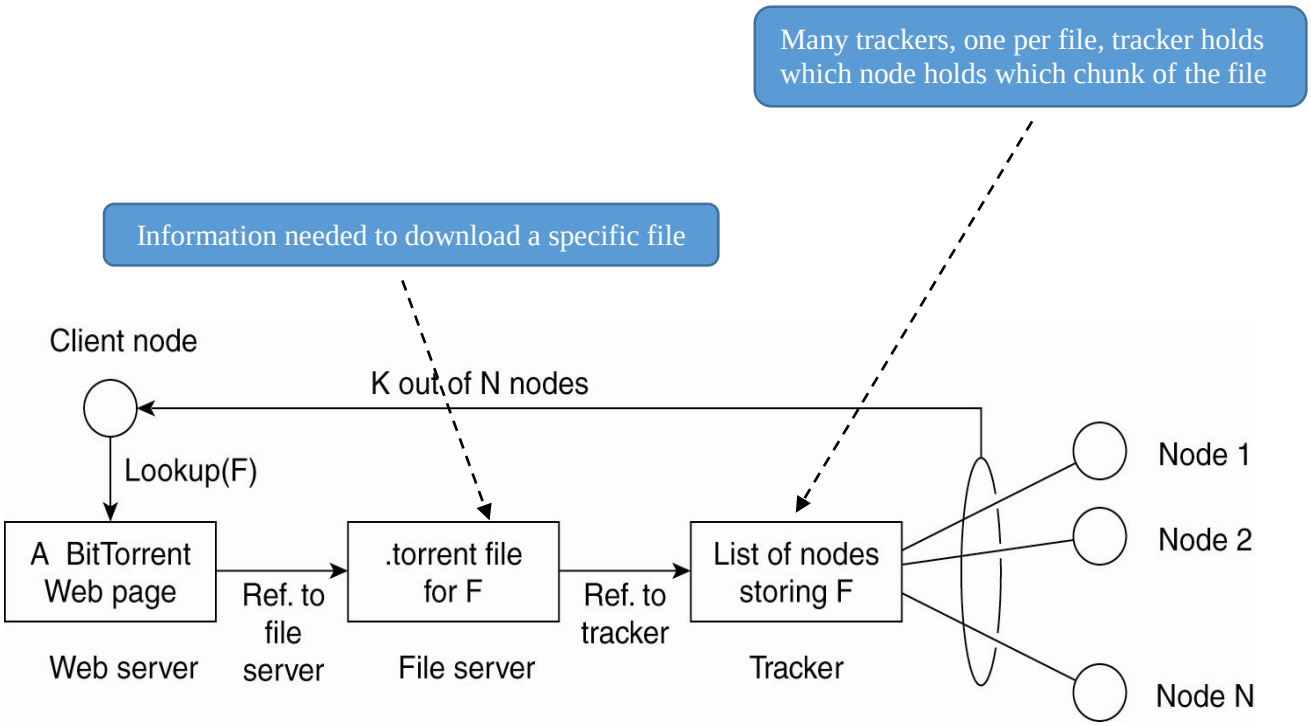



- معمولاً در اینترنت مستقر می شوند. سرویس دهنده ها در لبه شبکه قرار می گیرند. آغاز به کار با یک ساختار سرویس گیرنده- سرویس دهنده سنتی و متمرکز است اما در ادامه با اتصال گره‌ها به سیستم از ساختار غیرمتمرکز برای همکاری استفاده می کنند.

**Collaborative Distributed Systems**

مثال: سیستم اشتراک فایل Bit Torrent

Bit Torrent: is a P2p2 File downloading system. It allows download of various chunks of a file from other users until the entire file is downloaded
Globule: A Collaborative content distribution network. It allows replication of web pages by various web servers




Nodeها غیر متمرکز و Trackerها متمرکز هستند.

**Collaborative Distributed Systems**

Components of Globule collaborative content distribution network: (replication of web pages by various users)

- A component that can redirect client requests to other servers.
- A component for analyzing access patterns.
- A component for managing the replication of Web pages.
- Example:
  - Alice has a web server; Bob has a web server
  - Alice's server can have replicated contents of the Bob's server and vice versa
- Good if your server goes down
- Good if too much traffic that your server can not handle or server gets too slow
- Better Geographic diversity
- End users voluntarily provide enhanced web servers that are capable of collaborating in the replication of web pages

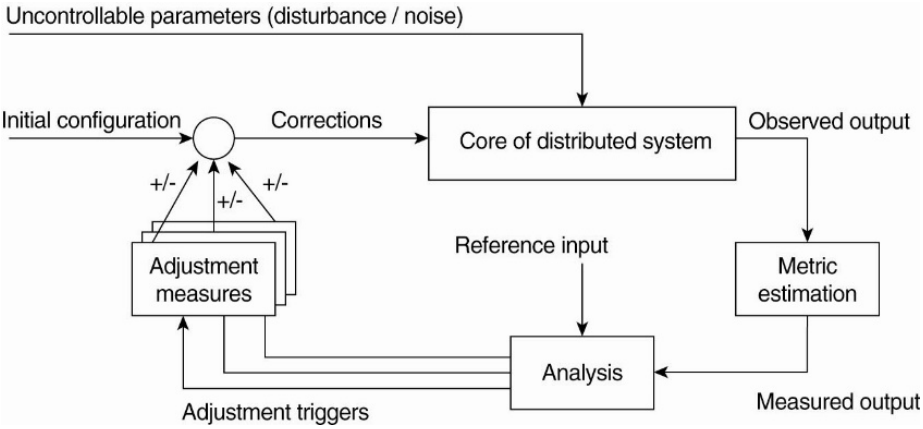
**معماری غیرمتمرکز در مقابل متمرکز**

- مدیریت سرویس دهنده های متمرکز نسبت به سرویس دهنده های غیر متمرکز ساده تر است.
- در سرویس دهنده های غیر متمرکز برعکس متمرکز معمولاً گلوگاه وجود ندارد.
- سرویس دهنده های متمرکز برعکس غیر متمرکز تحمل پذیری خطا ندارند.

## سیستم کنترل بازخورد

- Self-Management in Distributed systems:
  - Shield undesirable features (hacks, garbage, spam)
  - Self-Adaptive
  - Monitoring and adjustments should be possible
- Allowing automatic adaptation to changes (also called “autonomic Computing” or “Self-star systems”)
  - Self-managing
  - Self-healing (through replication)
  - Self-configuring
  - Self-Optimizing

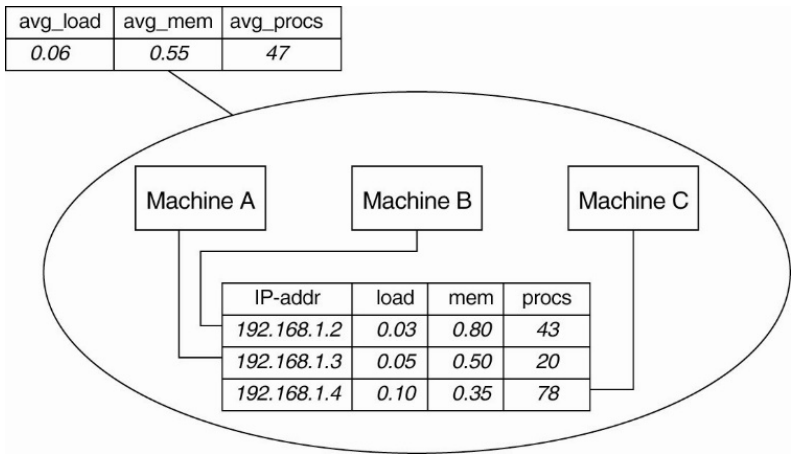
## سیستم کنترل بازخورد



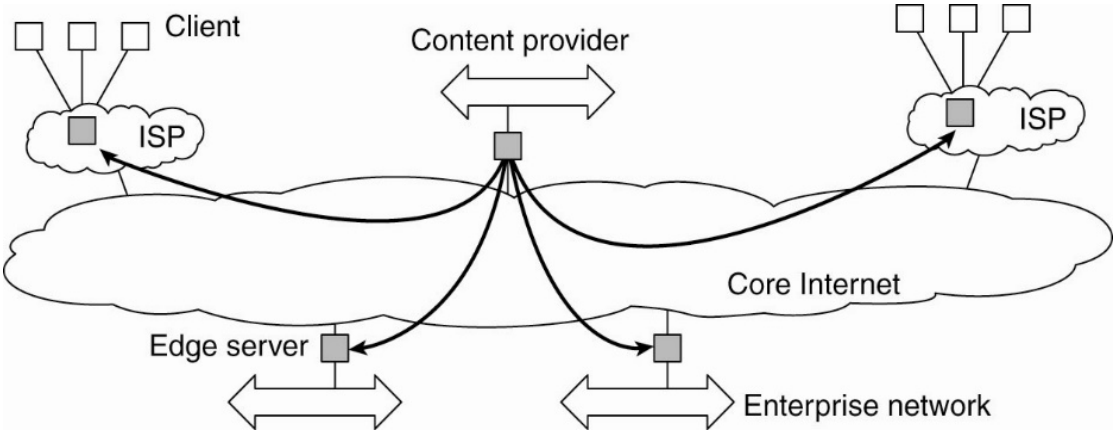

سازمان منطقی یک سیستم کنترل بازخورد

- استفاده از حلقه کنترل بازخورد متداولترین روش در سیستم های خود گردان است.

### مثال ۱: سیستم نظاره گر با Astrolabe



**مثال ۲: Differentiating Replication Strategies in Globule**



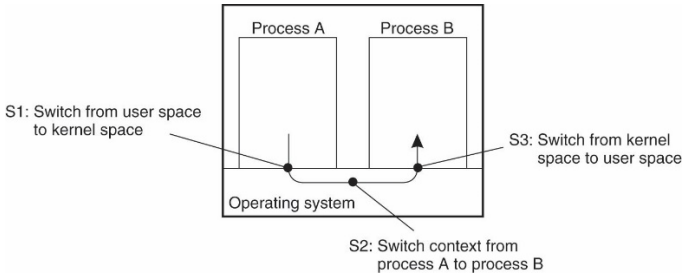

این سیستم در اینترنت کار میکند و برای توزیع داده ها بر مبنای سیستم سرویس دهنده لبه به کار می رود.

# فصل سوم: فرآیندها

## فرآیند چیست؟

- یکی از مهمترین قسمتها در کامپیوتر فرآیندها هستند، زیرا قسمت اعظم کارها توسط فرآیندها صورت می گیرد.
- فرآیند یک برنامه در حال اجرا است! که معمولاً دارای مالکیت منابع خاصی است.
- سیستم عامل برای فرآیند یک PCB (که همه اطلاعات فرآیند در آن است) ایجاد میکند.
- برای امکان چند برنامه ای سیستم عامل بین فرآیندها سوییچ میکند.

## تعویض فرآیند




Context switching as the result of IPC.

## نخ چیست؟

- هر نخ شامل مجموعه ای از دستورات است.
- نخها قسمتی از یک برنامه هستند و یک فرآیند در حال اجرا می تواند یک یا چند کار داشته باشد.
- دلیل استفاده از نخ ها: در حالتی که در یک فرآیند چند نخ همزمان اجرا شوند، امکان فراهم آوردن پردازش موازی و همچنین کار با برنامه های بزرگ ساده تر می شود.

## پیاده سازی نخ

- عموماً نخ ها به شکل بسته کوچک نخی با قابلیت‌های ایجاد، اتمام، انحصار استفاده از داده و همزمانی و نهایتاً مدیریت خطا ارائه می شوند.
- پیاده سازی یک بسته کوچک نخی : سطح کاربر ، سطح هسته ، ترکیبی

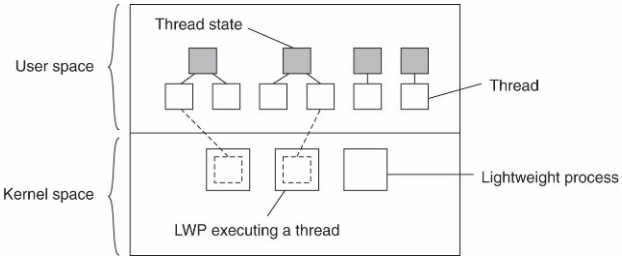
## مزایا و معایب نخ سطح کاربر

- ایجاد و تخریب نخ در این سطح ارزان است.
- غالباً سوئیچ کردن روی اجرای نخ دیگر فقط می تواند در چند دستورالعمل انجام گیرد (چون از حافظه مشترک استفاده میکنند).
- با بکار گیری System call مسدود کننده، بلافاصله کل فرآیندی را که نخ به آن تعلق دارد مسدود خواهد کرد و در نتیجه همه نخ های دیگر این فرآیند مسدود خواهند شد.

## مزایا و معایب نخ سطح هسته

- مشکلات سطح کاربر را ندارد.
- هزینه برای هر عمل نخ (ایجاد، حذف، همزمانی و غیره) که به System call نیاز دارند بالا است.
- سوئیچ کردن برای اجرای نخ می تواند به گرانی سوئیچ کردن برای فرآیند های دیگر باشد.

## پیاده سازی نخ ترکیبی




ترکیب فرایندهای سبک وزن سطح هسته و نخ های سطح کاربر

## مزایا و معایب نخ ترکیبی

- هنگام System call مسدود کننده، اگر LWP کافی باشد، کل فرآیند مسدود نمیشود.
- Application اطلاعی در مورد LWP ها نداشته و فقط نخهای سطح کاربر را میبینند.
- LWP ها به آسانی در محیط چند پردازشی مورد استفاده قرار میگیرند.
- ایجاد و حذف LWPها به گرانی سطح هسته است.

## نخ ها در سیستم های توزیع شده

- در سیستم های توزیع شده معمولاً از فرآیندهای چند نخی در سمت سرویس گیرنده و سرویس دهنده استفاده میشود.
- سرویس گیرنده های چند نخی (Multithreaded Clients)
- سرویس دهنده های چند نخی (Multithreaded Servers)

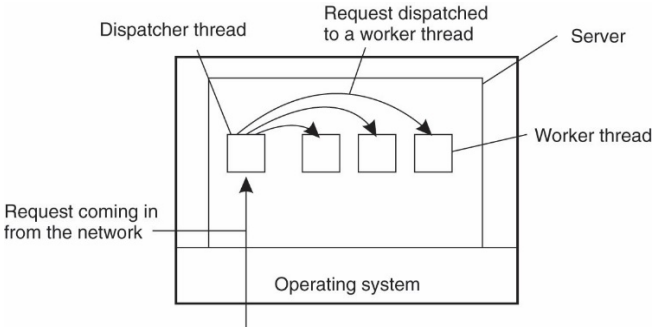
## مثالی از سرویس گیرنده چند نخی

- مرورگر وب چند نخی
- به محض اینکه فایل HTML اصلی واکشی شد، در صورت وجود عکس در فایل HTML یک نخ ایجاد شده و هر نخ یک اتصال جداگانه به سرویس دهنده محل قرار گرفتن عکس ایجاد می کند و داده ها را دریافت میکند.
- کاربر با کارایی (سرعت) بالاتری صفحات وب را دریافت می کند.
- همزمانی دریافت عکس ها و متن

## مثالی از سرویس دهنده چند نخی

- سرویس دهنده فایل چند نخی
- مدل توزیع کننده/کارگر: که در آن فرآیند سرویس دهنده یک نخ توزیع کننده دارد و چند نخ کارگر. هر درخواستی که به سرویس دهنده می رسد، توسط توزیع کننده دریافت می شود و بین کارگران تقسیم می شود.
- همزمانی ارسال فایل ها برای چند کاربر

## سرویس دهنده چند نخی توزیع کننده/کارگر

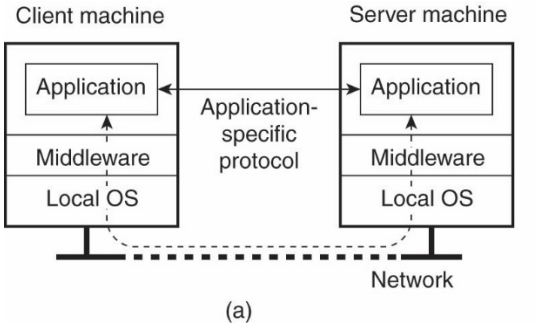



یک سرور چند نخی شده که در مدل کارگر / توزیع کننده سازماندهی شده است

## سرویس گیرنده ها در سیستم توزیع شده

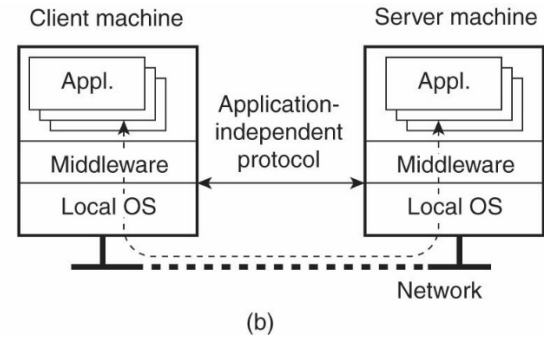
۱- واسطه های کاربر شبکه شده(Networked User Interfaces)

۲- نرم افزار طرف سرویس گیرنده برای شفافیت توزیع شده(Client-Side Software for Distribution Transparency)




(a) A networked application with its own protocol





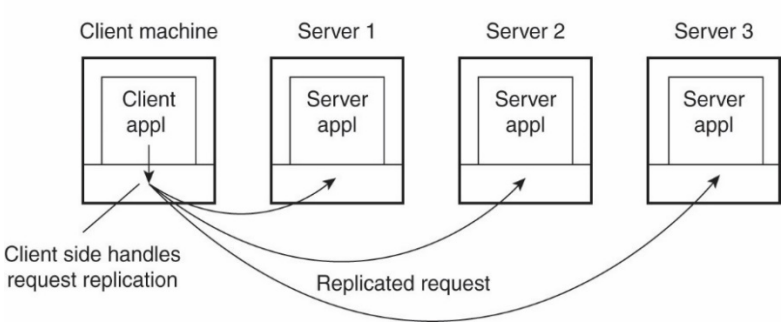

(b) A general solution to allow access to remote applications

### ۱- واسطه های کاربر  شبکه شده

- کار اصلی اکثر سرویس گیرنده ها، تعامل با کاربر انسانی و سرویس دهنده های راه دور است.
- پشتیبانی از واسطه کاربر، از خصوصیات کلیدی بیشتر سرویس گیرنده ها است.

### ۲- نرم افزار طرف سرویس گیرنده برای شفافیت توزیع شده

- نرم افزار سرویس گیرنده شامل چیزهای بیشتری از واسطه های کاربر است.
  - مثال : ماشین های خود پرداز(ATM) ،  ثبت کننده های چک، بار کد خوان ها
- یک سرویس گیرنده نباید از اینکه در حال ارتباط با پردازش های راه دور است آگاه شود.
- در بدترین حالت، برنامه های کاربردی سرویس گیرنده ممکن است متوجه افت موقتی قابلیت اجرا بشود.
- خیلی از سیستمهای توزیع شده شفافیت تکثیر را بوسیله نرم افزار طرف سرویس گیرنده پیاده سازی می کنند.




Transparent replication of a server using a client-side solution

**سرویس دهنده ها در سیستم توزیع شده**

- یک سرویس دهنده فرایندی است که سرویس خاصی را به نفع یک مجموعه از سرویس گیرنده پیاده سازی می کند.

**۱- انواع سرویس دهنده ها از دیدگاه روش پیاده سازی:**

- تکراری (Iterative Server)
  - خود سرویس دهنده درخواست را اداره نموده و در صورت لزوم پاسخی را به سرویس گیرنده می فرستد.
- همروند (Concurrent Server)
  - همانند چند نخی ها، درخواست های رسیده توسط سرویس دهنده اداره نمیشود بلکه آن را به فرایند یا نخ دیگری میفرستد و خود منتظر درخواست بعدی می شود.

**۲- انواع سرویس دهنده ها از دیدگاه نگهداری وضعیت سرویس گیرنده**

- Stateless Server: هیچ وضعیتی از سرویس گیرنده نگهداری نمیشود و در صورت خرابی یا قطعی سرویس دهنده، عملیات از اول شروع میشود.
- Tasteful Server: در Tasteful حالت و وضعیت سرویس گیرنده نگهداری میشود.
- Soft state Server: در Soft State تا مدت محدودی اطلاعات سرویس گیرنده نگهداری میشود.

**۳- انواع سرویس دهنده ها از دیدگاه نوع ارتباط با سرویس گیرنده ها**

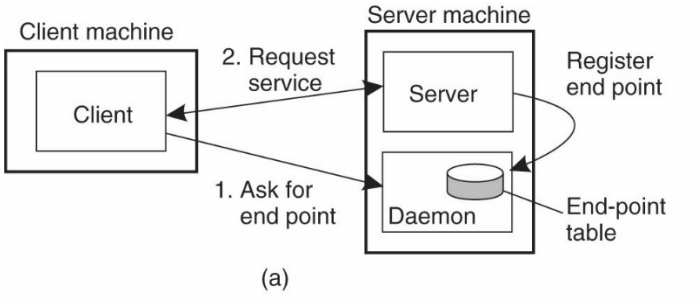
- End point(port)

در تمامی موارد سرویس گیرنده در خواست ها را به یک نقطه انتهایی به نام پورت در ماشینی که سرویس دهنده اجرا میشود می فرستد و هر سرویس دهنده به نقاط انتهایی گوش می کند.

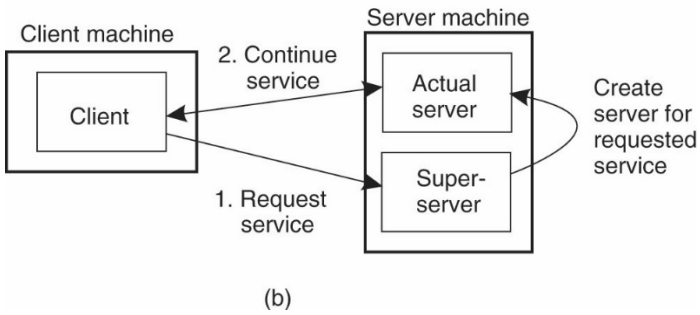
- Super server

دارای فقط یک سرویس دهنده است. در واقع ایجاد این ارتباط را بر عهده می گیرد و درخواست سرویس گیرنده را گرفته و به سرویس دهنده اختصاص میدهد.

**Client-to-server binding using a daemon**




**Client-to-server binding using a Super server**




## مهاجرت کد در سیستمهای توزیع شده(Code Migration)

۱- دلایل مهاجرت کد

- تا اینجا در بیشتر در مورد ارسال و جابجایی داده ها در سیستم توزیع شده بحث شد.
- ارسال و جابجایی برنامه ها (حتی در حین اجرای آنها) طراحی سیستم های توزیع شده را ساده تر می نماید.
- البته جابجایی برنامه ها بین سیستم های نامتجانس مشکل تر خواهد بود.

**Migrating Code Examples**

- Example 1: (Send Client code to Server)
  - Consider a client-Server system where server holds a huge database.
  - If a client application needs to perform many database operations, it may be better to ship part of the client application to the server and server sends only the results across the network.
- Example 2: (Send Server code to Client)
  - Data validation at the Database level:
  - In many interactive DB applications, clients need to fill in forms that are subsequently translated into a series of DB operation where validation at server side is required.

## مزایای مهاجرت کد

۱- اجرا شدن به صورت موازی
۲- استفاده از قدرت پردازنده ها و بالا رفتن کارایی
۳- توازن و تعادل بار
۴- انعطاف پذیری
۵- کاهش ارتباطات شبکه

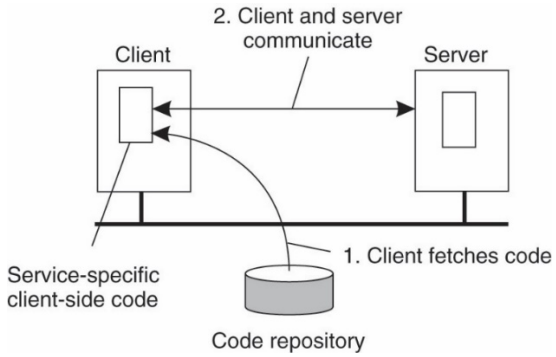
## معایب مهاجرت کد

۱- امنیت پایین می آید (زیرا این کد ممکن است مخرب باشد)
۲- خیلی وقتها مهاجرت کد یا پروسس به جای بهبود کارایی باعث کاهش کارایی می شود ( مثلا بدلیل وجود هزینه های ارتباطی)

## مهاجرت کد

- انتقال کد های برنامه به دلایلی در سیستم های توزیع شده صورت می پذیرد.
- انتقال کدهای برنامه جایگزین انتقال فرایندها شده است (Client-Side Programming)
- در انتقال فرایندها کل یک فرایند از یک ماشین به ماشین دیگر منتقل می شود.

## مهاجرت کد

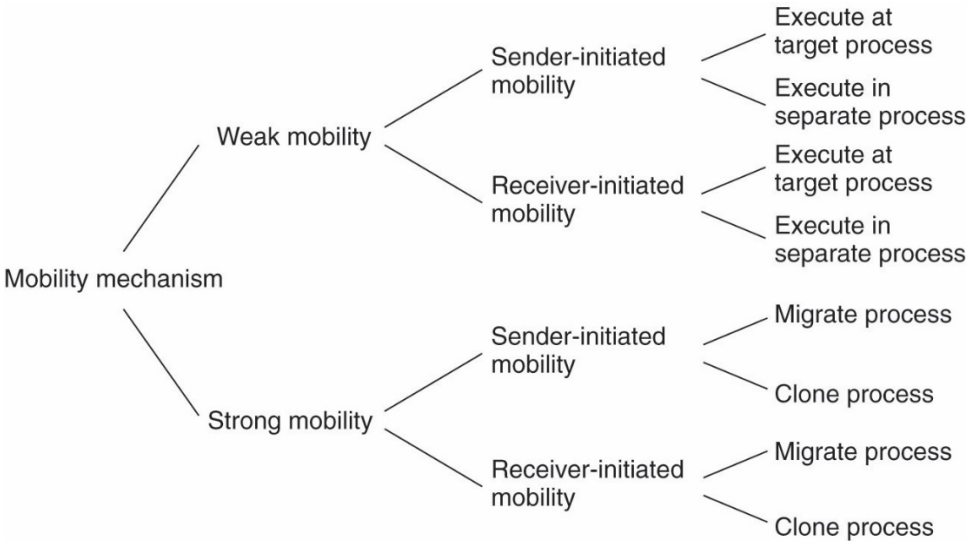



اصول پیکر بندی خودکار یک کلاینت برای برقراری ارتباط با یک سرور. کلاینت ابتدا نرم افزار لازم را واکشی می کند و سپس آن سرور را فراخوانی می کند.

## اجزای یک فرایند

۱- قطعه کد: مجموعه دستورالعمل ها که فرایند در حال اجرا را می سازد.
۲- قطعه منبع: شامل ارجاع به منابع مورد نیاز فرآیند.
۳- قطعه اجرا: جایی برای ذخیره حالت فعلی اجرای فرایند (PCB)

**انواع مدلهای مهاجرت پروسس**




**انواع مهاجرت فرایندها:**

**جابجایی ضعیف Weak mobility**

- در این حالت تنها قطعه کد قابل جابجایی است و قطعه اجرا منتقل نمیشود.
- در این حالت همیشه فرایند از وضعیت ابتدایی شروع به کار می کند. مثال: اپلت های جاوا
- مهمترین حسن این روش سادگی آن است.

**جابجایی قوی Strong mobility**

- در این حالت علاوه بر قطعه کد، قطعه اجرایی نیز قابل انتقال است.
- امکان اجرای ادامه فرایند مهاجرت کننده را دارد (می تواند از ادامه اش اجرا شود).
- این مدل قوی تر و در عین حال پیچیده تر است.
- این مدل زیاد کاربردی نیست.

**آغاز شده توسط فرستنده Sender-initiated**

- جابجایی توسط ماشینی که کد روی آن است آغاز می شود.

مثال: ارسال برنامه جستجو در اینترنت ، برنامه با قابلیت پردازش موازی (شطرنج)

**آغاز شده توسط گیرنده Receiver-initiated**

- در این مدل این ماشین مقصد است که کار جابجایی را آغاز می کند.
- این مدل معمولاً ساده تر از مدل قبلی پیاده سازی می شود. مثال: اپلت های جاوا

**مهاجرت قطعه منابع: (اتصال فرایند به منبع)**

۱- By Value: فرایند با مقدار منابع سروکار دارد (فرایندی که از پایگاه داده یا فایل می خواند).
۲- By Type: فرایند از نوع خاصی از منبع استفاده می کند. مثلاً مانیتور
۳- By Identifier: فرایند که با شناسهٔ منبع سروکار دارد. مثلاً Port یا web service ویا URL

**مهاجرت قطعه منابع: (اتصال منبع به ماشین)**

۱- Unattached (غیر متصل): منبع براحتی قابل جدا شدن از ماشین و قابل انتقال است. مثل DLL و فایل کوچک
۲- Fastened( بسته شده): منبع قابل جدا شدن از ماشین و قابل انتقال است ولی هزینه بر است. مثل پایگاه داده بزرگ
۳- Fixed (ثابت): منبع از ماشین جدا نمی شود. مثل پرینتر Local disk drives, communication ports.

**راه حل مهاجرت قطعه منابع**

GR منبع عمومی به گستردگی سیستم ایجاد میکند.
MV منبع را نقل مکان میدهد.
CP مقدار منبع را کپی میکند.
RB فرایند با منابع در دسترس محلی را دوباره اختیار میکند.

**Resource-to-machine binding**

| Process-to-resource binding | Unattached | Fastened | Fixed |
|---|---|---|---|
| By identifier | MV (or GR) | GR (or MV) | GR |
| By value | CP (or MV,GR) | GR (or CP) | GR |
| By type | RB (or MV,CP) | RB (or GR,CP) | RB (or GR) |

GR Establish a global systemwide reference
MV Move the resource
CP Copy the value of the resource
RB Rebind process to locally-available resource

**مهاجرت در سیستم های ناهمگن**

- در سیستمهای ناهمگن ممکن است انتقال به ماشینی صورت گیرد که ساختار متفاوتی (نرم افزاری و سخت افزاری) از ماشین مبدأ داشته باشد.
- باید اطمینان وجود داشته باشد که پس از انتقال کد به ماشین مقصد، کد ها قابل اجرا باشند.
- ممکن است نیاز به ترجمه مجدد کد ها وجود داشته باشد.
- راه حل مهاجرت در سیستم های ناهمگن استفاده از Middleware ماشین مجازی مثل (JVM) است. که بصورت مجازی همه را همگن می کند.

# فصل چهارم: ارتباطات

- ارتباطات بین فرآیندها یکی از مهمترین مسائل در سیستم های توزیع شده است.
- ارتباطات بین فرآیندها قلب یک سیستم های توزیع شده است.
- سه مدل ارتباطی پر استفاده
  - فراخوانی رویه راه دور(RPC)
  - میان افزار پیام گرا (MOM)
  - جریان داده ها

## انواع پروتکل ها

۱- اتصال گرا

- قبل از مبادله داده ها، باید بین فرستنده و گیرنده اتصال برقرار شود.
- تلفن، سیستم ارتباطی اتصال گرا است.

۲- بدون اتصال (غیر اتصال گرا)

- لازم نیست از قبل اتصال برقرار شود.
- قرار دادن نامه در صندوق پستی نمونه ای از ارتباط بی اتصال است

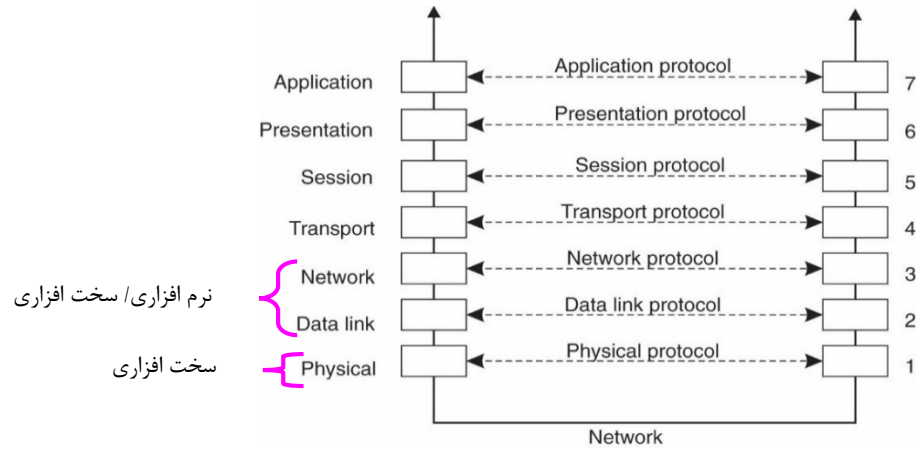
## تفاوتهای اتصال گرا و بدون اتصال

۱- در اتصال گرا برای بسته های به هم مرتبط، تنها یکبار مسیریابی میشود ولی در بدون اتصال، برای هر بسته مسیر یابی میشود.
۲- در اتصال گرا بسته ها به ترتیب به مقصد می رسند ولی در بدون اتصال بدون ترتیب می رسند.
۳- در اتصال گرا چون بسته ها به ترتیب می روند فقط بسته اول اطلاعات کامل درباره مقصد دارد و بقیه بسته ها فقط شناسه مورد نظر را دارد ولی در بدون اتصال همه بسته ها اطلاعات کامل درباره مقصد دارند.
۴- در اتصال گرا چون مسیر بسته ها را می دانیم کنترل ترافیک بهتر است.

## معایب اتصال گرا

۱- در برابر تغییرات ناگهانی ترافیک شبکه ضعیف است و در صورت وجود ترافیک شدید امکان تغییر مسیر وجود ندارد.
۲- اگر بدلیلی مسیر ارتباطی قطع شود، هیچ راهی جز اینکه دوباره بسته ها را از اول بفرستد، وجود ندارد.

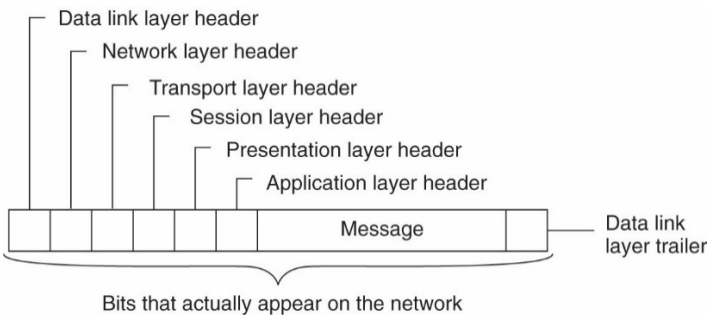
## لایه ها، واسط ها، و پروتکل ها در مدل OSI




## مدل OSI

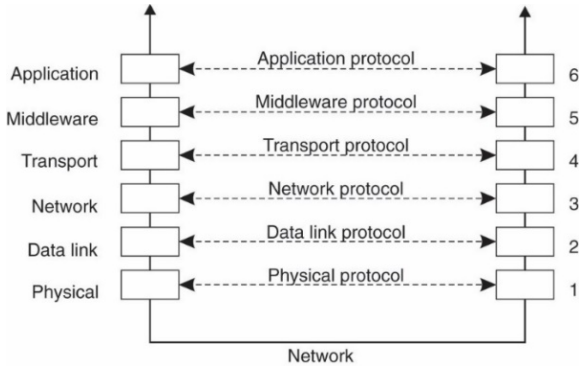
- مدل OSI یک مدل مرجع برای اداره و کنترل سطوح مختلف موجود در ارتباطات است.
- در مدل OSI هر لایه، واسطی را به لایه بالاتر از خودش ارائه می کند.
- وقتی فرآیند A در ماشین ۱ می خواهد با فرآیند B در ماشین ۲ ارتباط برقرار کند، پیامی میسازد و آن را به لایه کاربرد در ماشین خودش می فرستد و نرم افزار لایه کاربرد، سرآیندی را به جلوی پیام اضافه می کند و پیام از طریق واسط به لایه نمایش ارسال می شود و به همین ترتیب ادامه می یابد.

## نمونه ای از پیام در شبکه




## مدل مرجع برای ارتباطات سیستم توزیع شده

- در سیستم توزیع شده یک Middleware اضافه می شود.



## ارتباطات شبکه در سیستم توزیع شده

- لایه کاربردی با Middleware در ارتباط است و Middleware تمامی کارها را از دید لایه کاربردی مخفی نگه می دارد.
- Middleware وظیفه فراهم کردن سرویس ارتباطات را برای لایه کاربردی فراهم می کند.

## مزایای Middleware

۱- شفافیت (Transparency)
۲- همگن سازی.
۳- در اختیار گذاشتن منابع بصورت آسان برای کاربر.

## انواع ارتباطات

پایدار (Persistent) - ناپایدار (Transient) - همگام (Synchronous) - ناهمگام (Asynchronous)

## ارتباط پایدار

- در ارتباط پایدار وقتی که دو شیئ برای همدیگر انتقال داده میکنند، پیغام گم نمیشود.
- در ارتباط پایدار نیازی نیست که گیرنده روشن باشد.
- این ارتباط به سیستم صف نیز معروف است.
- مثال ارتباط پایدار مثل SMS و Email

## ارتباط ناپایدار

- در ارتباط ناپایدار وقتی که دو نفر برای همدیگر انتقال داده می کنند، ممکن است پیغام گم شود.
- در ارتباط ناپایدار باید گیرنده روشن باشد.

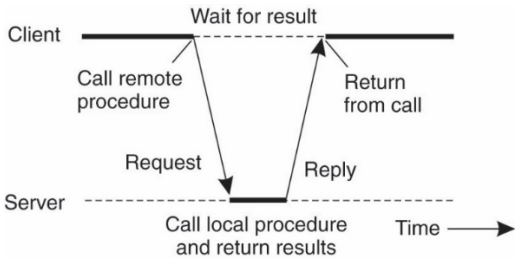
## ارتباط همگام و ناهمگام

- در ارتباط همگام فرستنده بعد ازتحویل دادن پیام متوقف میشود تا مطمئن شود درخواست او پذیرفته شده است.
- در ارتباط ناهنگام فرستنده به محض تحویل دادن پیام به کارش ادامه می دهد.

## فراخوانی رویه از راه دور (RPC)

- ایده ماورای RPC این است که فراخوانی رویه راه دور را حتی الامکان مشابه با فراخوانی محلی انجام دهد و می خواهیم RPC شفاف باشد. (رویه فراخوان نباید اطلاع داشته باشد که کار در کامپیوتر دیگری اجرا می شود و برعکس)
- تابع یا پروسیجری را که در کامپیوتر خودمان نیست صدا بزنیم و ارتباطات را در تابع قرار می دهیم بدون اینکه کامپیوتر مبدا متوجه شود.

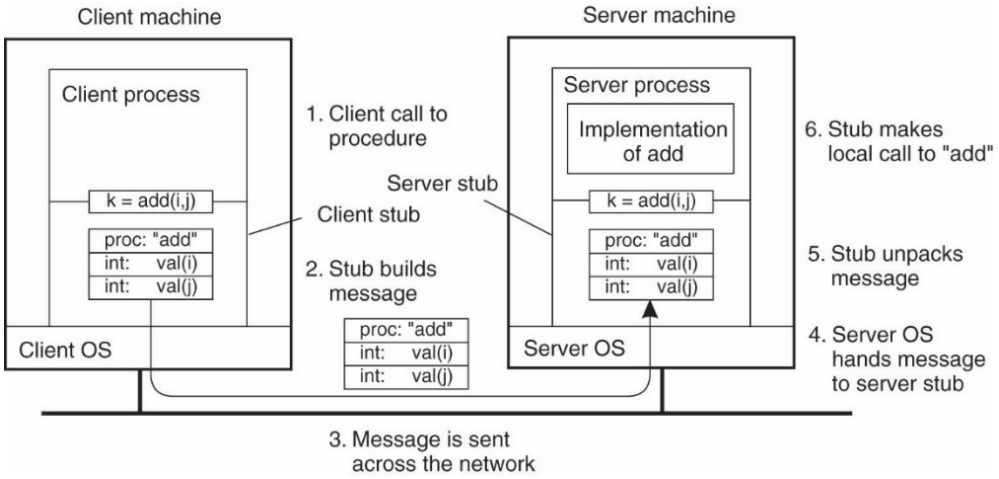
## نمونه های سرویس گیرنده و سرویس دهنده




اصل RPC بین برنامه سرویس دهنده و سرویس گیرنده

## مراحل فراخوانی رویه راه دور

۱- رویه سرویس گیرنده، نمونه سرویس گیرنده را به روش عادی فراخوانی میکند.
۲- ریشه (Stub) سرویس گیرنده، پیامی را می سازد و سیستم عامل راه دور را فراخوانی میکند.
۳- سیستم عامل سرویس گیرنده، پیام را به سیستم عامل راه دور می فرستد.
۴- سیستم عامل راه دور، پیام را به ریشه سرویس دهنده میدهد.
۵- ریشه سرویس دهنده پارامتر را استخراج می کند و سرویس دهنده را فراخوانی می نماید.
۶- سرویس دهنده کار را انجام می دهد و نتیجه را به ریشه بر می گرداند.
۷- ریشه سرویس دهنده، نتیجه را در پیام بسته بندی می کند و سیستم عامل محلی خود را فراخوانی می نماید.
۸- سیستم عامل سرویس دهنده پیام را به سیستم عامل سرویس گیرنده می فرستد.
۹- سیستم عامل سرویس گیرنده، پیام را به ریشه سرویس گیرنده میدهد.
۱۰- ریشه پیام را استخراج می کند و به سرویس گیرنده برمی گرداند.




مراحل انجام محاسبات راه دور از طریق RPC.

## ارسال پارامتر در RPC

1- Call by value: مقدار پارامتر ها را می فرستد.
2- Call by reference: در این ارسال، آدرس مکان پارامتر فرستاده می شود.
3- Call by copy/restore (Name)

## ارسال پارامترها با مقدار

- مدل فوق تا زمانی که ماشین های سرویس گیرنده و سرویس دهنده یکسان باشند و تمام پارامترها و نتایج از نوع اسکالر باشند، به خوبی کار میکند. اما در سیستم های توزیع شده بزرگ (معمولاً ماشین های ناهمگن)، هر ماشین برای مقادیر عددی، کاراکتری و...، نمایش خاصی دارد و ارسال پارامتر با مشکل مواجه خواهد شد. بعنوان مثال در کامپیوترهای بزرگ IBM با کامپیوترهای شخصی IBM.

## ارسال پارامترها با مرجع

- در این ارسال، آدرس مکان فرستاده می شود. و چون آدرس‌ها در دو کامپیوتر مختلف یکسان نیست این روش امکانپذیر نیست.

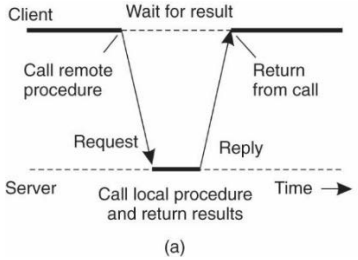
راه حل ۱): اجازه ارسال شدن پارامترهای مرجع را ندهیم و تنها پارامترهای مقدار اجازه ارسال شدن داشته باشند.
راه حل ۲): ارسال پارامترها با کپی/بازیابی

## ارسال پارامترها با کپی و بازیابی

- داده ای که پارامتر مرجع به آن اشاره میکند برای سرویس گیرنده ارسال شود و آن تغییرات روی آن انجام پذیرد و بعد کل داده باز گردد.
- به دلیل اینکه داده هایی که به آنها ارجاع داده می شود اغلب بزرگ هستند (مثل آرایه) این عمل ترافیک مسیر ارتباطی را بالا میبرد.

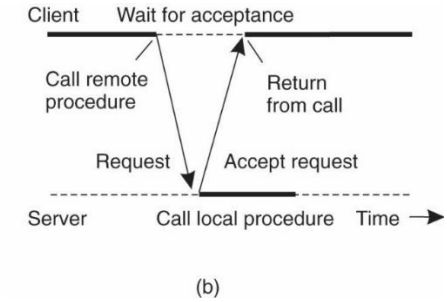
## RPC همگام

همانند فراخوانی های رویه معمولی، سرویس گیرنده پس از درخواست RPC، مسدود میشود.
مثال) انتقال پول از یک حساب به حساب دیگر، افزودن داده هایی به بانک اطلاعاتی، پردازش دسته ای و...



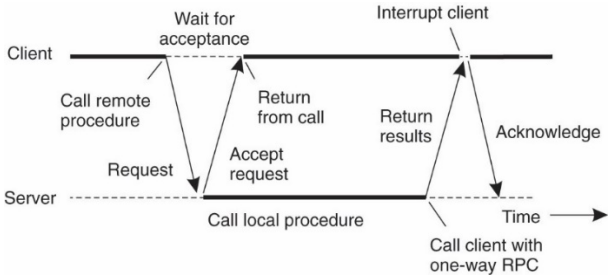

تعامل بین سرویس دهنده و سرویس گیرنده در RPC سنتی.

- سرویس گیرنده پس از درخواست RPC، فوراً به کارش ادامه می دهد.
- با RPCهای ناهمگام، سرویس دهنده بلافاصله پس از دریافت فراخوانی RPC پاسخی را ارسال می کند.




تعامل با استفاده از RPC ناهمگام.

## RPC ناهمگام




تعامل سرویس دهنده و سرویس گیرنده از طریق دو RPC ناهمگام.

## ارتباط ناپایدار پیام گرا (Transient Message-oriented)

- سوکت برکلی
- واسط مبادله پیام (MPI)

## سوکت برکلی

- توجه ویژه ای به استاندارد سازی واسط لایه انتقال انجام شد تا به برنامه نویسان اجازه دهد از طریق مجموعه ای از عملیات های پایه، از کل پشته پروتکل ها استفاده کنند.
- واسط های استاندارد، انتقال کاربرد را به ماشین مختلف آسان می سازد.

## عملیات های پایه سوکت برای TCP/IP

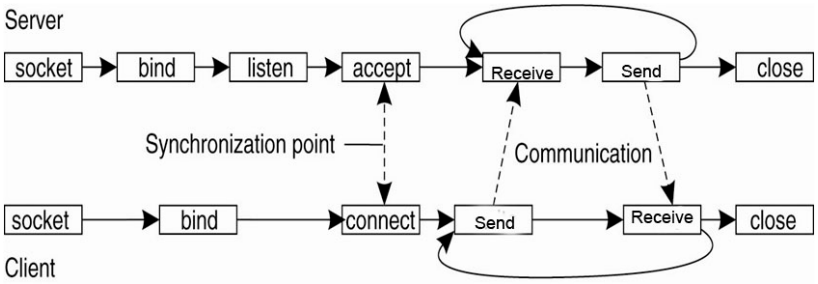
| Primitive | Meaning |
|---|---|
| Socket | Create a new communication end point |
| Bind | Attach a local address to a socket |
| Listen | Announce willingness to accept connections |
| Accept | Block caller until a connection request arrives |
| Connect | Actively attempt to establish a connection |
| Send | Send some data over the connection |
| Receive | Receive some data over the connection |
| Close | Release the connection |

server فقط

client فقط

سرویس دهنده ها معمولاً چهارعملیات پایه اول را به همین ترتیب اجرا می کنند.

## الگوی ارتباطی اتصال گرا با استفاده از سوکت ها

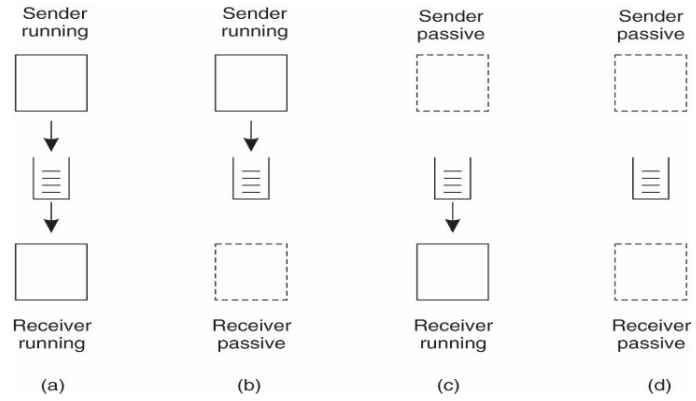



## ارتباط پایدار پیام گرا

- سیستم صف بندی پیام (میان افزار پیام گرا(Message-Oriented Middleware(MOM )
- معماری کلی صف بندی پیام
- کارگزاران پیام

## سیستم صف بندی پیام

- برنامه های کاربردی با قرار دادن پیام ها در صف خاص، با یکدیگر ارتباط برقرار می کنند.
- پیام معمولاً مستقیماً به سرویس دهنده مقصد منتقل می شود.
- هر کاربرد دارای صف اختصاصی خودش است که کاربردهای دیگر می توانند پیام هایی را به آن بفرستند.
- صف فقط می تواند توسط کاربرد خودش خوانده شود، اما چندین کاربرد می توانند در یک صف مشترک باشند.
- جنبه مهم این مدل این است که به فرستنده اطمینان داده می شود که پیامش سرانجام در صف گیرنده قرار خواهد گرفت.
- فرستنده و گیرنده می توانند کاملاً مستقل از یکدیگر اجرا شوند. (ارتباط از نوع اتصال ضعیف)
- وقتی پیامی در صف قرار گرفت، در آنجا می ماند تا حذف شود و کاری ندارد که فرستنده و گیرنده اش در حال اجرا است یا خیر.



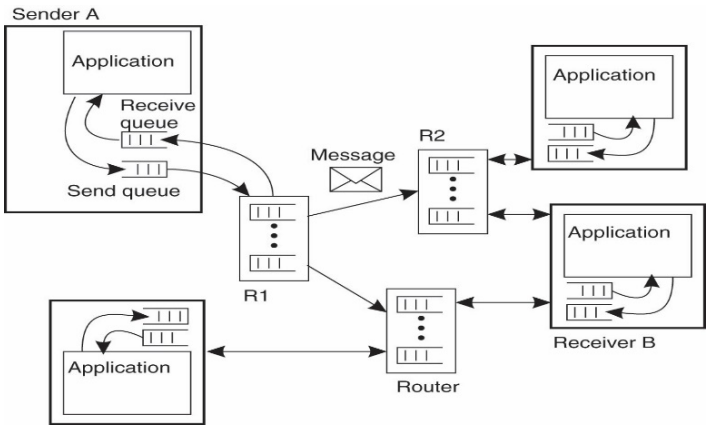

چهار ترکیب برای ارتباط با انسجام ضعیف با استفاده از صف

پیام ها می توانند شامل هر داده ای باشند و تنها باید به درستی آدرس دهی شوند.

| Primitive | Meaning |
|---|---|
| Put | Append a message to a specified queue |
| Get | Block until the specified queue is nonempty, and remove the first message |
| Poll | Check a specified queue for messages, and remove the first. Never block |
| Notify | Install a handler to be called when a message is put into the specified queue |

## معماری کلی سیستم صف بندی پیام

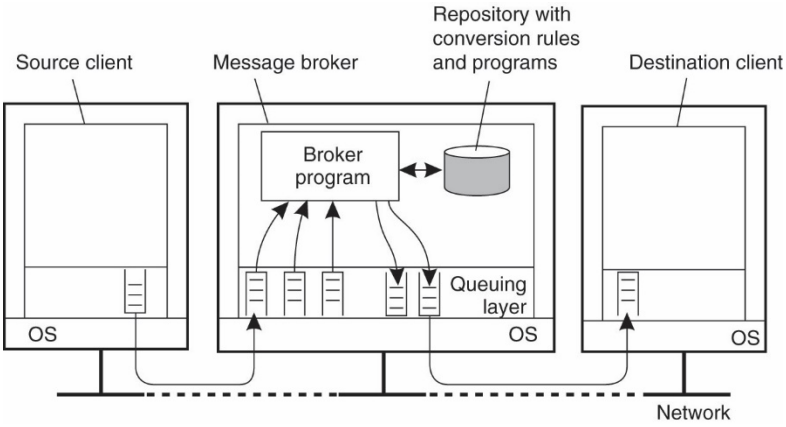
- مجموعه ای از صف ها در چندین ماشین توزیع شده اند و برای اینکه سیستم صف بندی پیام بتواند پیام را منتقل کند، باید نگاشتی از صف ها به مکان های شبکه را نگهداری نماید.
- این نگاشت شبیه استفاده از DNS در اینترنت است.
- صف ها به وسیله مدیران صف اداره می شوند.
- مدیر صف مستقیماً با کاربردی که پیام را می فرستد یا میگیرد، تعامل دارد. اما مدیران صف خاصی وجود دارند که مثل مسیریاب یا تقویت کننده عمل می کند.




سازمان کلی سیستم صف بندی پیام با مسیر یاب ها.

## کارگزاران (Broker) پیام

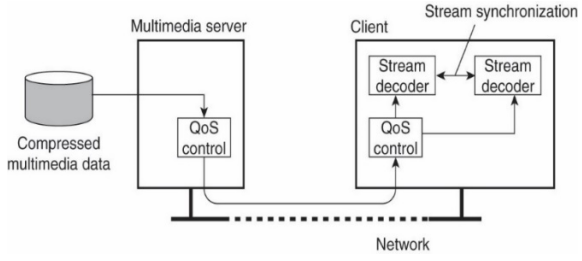
- هدف اصلی تبدیل پیام های ورودی است، به طوری که توسط کاربرد مقصد قابل فهم باشند.
- کارگزاران پیام بخشی از سیستم صف بندی محسوب نمیشود.
- استفاده از کارگزاران پیام برای جامعیت کاربرد است.
- در قلب کارگزاران پیام، مخزنی از قوانین و برنامه ها وجود دارد که می تواند پیامی از نوع T1 را به T2 تبدیل کند.




سازمان کلی کارگزاران پیام در سیستم صف بندی پیام.

## جریان داده (Data Stream)

- جریان داده، دنباله ای از واحد های داده است.
- جریان های داده می توانند به رسانه پیوسته و گسسته تقسیم شوند.
- زمان برای جریان های داده پیوسته مهم است. مثال : نمایش فیلم برای پیوسته. مثال تصاویر ثابت برای گسسته
- برای دستیابی به جنبه های زمانی معمولاً بین حالت های مختلف انتقال، تمایز قائل می شویم.
- در حالت انتقال ناهمگام asynchronous، اقلام داده ها در یک جریان داده بدون محدودیت زمانی یکی پس از دیگری منتقل میشوند، این کار معمولاً در جریان های داده گسسته صورت می گیرد.
- در حالت انتقال همگام synchronous، حداکثر تأخیر انتها به انتها وجود دارد که برای هر واحد داده تعریف شده است. مهم نیست که واحد داده سریع تر از حداکثر تأخیر منتقل شود یا خیر. مثال: شبکه حسگر برای فهم رخداد آتش
- در حالت انتقال همزمان isochronous، باید الزاما یک تاخیر زمانی ثابت رعایت شود. مثال: کاربرد مولتی مدیا
- جریان ها می توانند ساده یا پیچیده باشند.
- جریان ساده simplex فقط شامل یک دنباله از داده ها است،در حالی که جریان پیچیده complex شامل چند جریان ساده مرتبط، به نام زیر جریان ها sub stream است.
- ارتباط بین زیر جریان ها در جریان پیچیده، به زمان وابسته است.
- مثال) صدای استریو یا انتقال فیلم.




معماری کلی برای جریان داده های چند رسانه ای ذخیره شده روی شبکه.

## کیفیت سرویس

- نیازمندیهای زمانی، معمولاً به عنوان نیازمندی های کیفیت سرویس (QoS) نامیده میشوند.
- کیفیت خدمات معمولا از سه جنبه سنجیده می شود:

۱- زمان

۲- حجم و اندازه

۳- قابلیت اطمینان (تحمل پذیری خطا)

## خصوصیاتی برای QoS:

۱- نرخ بیتی مورد نیاز که داده ها باید منتقل شوند.

۲- حداکثر تأخیر برای برقراری نشست (یعنی زمانی که کاربرد می تواند شروع به ارسال داده ها کند)

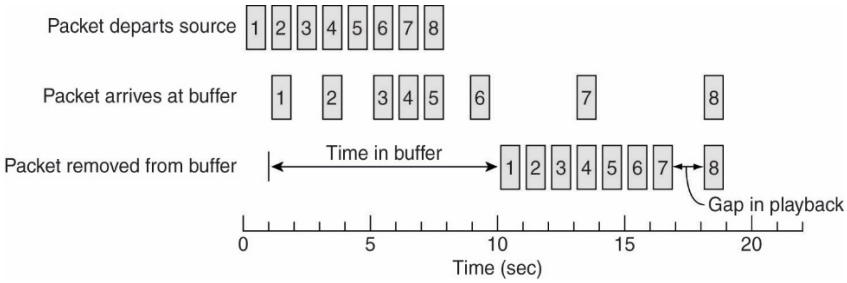
۳- حداکثر تأخیر انتها به انتها (چقدر طول می کشد تا داده ها به گیرنده برسد)

۴- حداکثر واریانس تأخیر یا لرزش Jitter

۵- حداکثر تأخیر رفت و برگشت

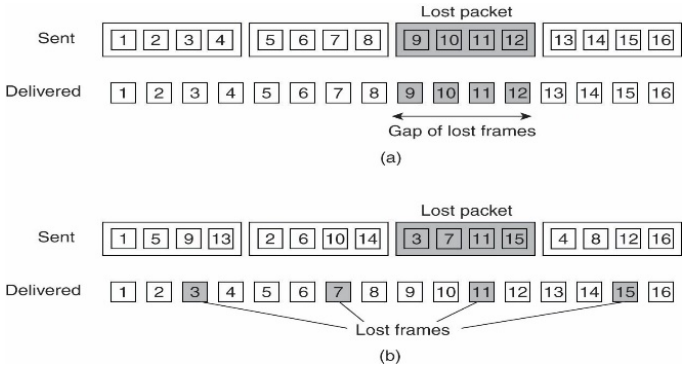
۶- نرخ خطا

## فراهم کردن QoS




Using a buffer to reduce jitter

## اثر مفقود شدن بسته




a) انتقال بدون فاصله گذاری  b) انتقال با فاصله گذاری

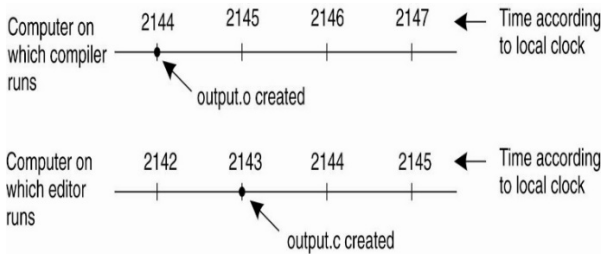
# فصل ششم: همگام سازی

۱- همگام سازی ساعت‌ها
۲- همگام سازی فرآیندها

- در دستیابی انحصاری، فرآیندها نباید به طور همزمان و همروند به منبع مشترکی، مثل چاپگر دستیابی داشته باشند.
- در بسیاری موارد، مهم است گروهی از فرآیندها بتوانند برای دستیابی انحصاری به منابع، یک فرآیند را به عنوان هماهنگ کننده بپذیرند، که می تواند بر اساس الگوریتم انتخاب، صورت گیرد.

## همگام سازی ساعت

- در سیستم متمرکز زمان مبهم نیست؛ هر پروسس با درخواست زمان میتواند آخرین زمان را دریافت نماید. ولی در سیستم توزیع شده، توافق بر سر زمان، ساده نیست.
  مثال) روش کار در برنامه یا یوتیلیتی make یونیکس (ترکیب فایلهای یک پروژه)
- وقتی برنامه نویس تغییر در تمام فایلهای مبدأ را به اتمام رساند، برنامه make را اجرا می نماید. این برنامه، با بررسی زمان اصلاح تمام فایلهای مبدأ(source)و مقصد(object) متوجه میشود کدام فایل مبدا نیاز به ترجمه مجدد دارد.
- اگر فایل مبدأ(input.c) دارای زمان ۲۱۵۱ و فایل مقصد آن(input.o)، دارای زمان ۲۱۵۰ باشد، make متوجه می شود که input.c پس از ایجاد input.o تغییر کرده است و در نتیجه input.c باید دوباره ترجمه شود.
- از طرف دیگر، اگر input.c دارای زمان کمتری از input.o باشد نیاز به ترجمه مجدد نیست.




- وقتی هر ماشین دارای ساعت خاص خودش است، رویدادی که پس از رویداد دیگری اتفاق افتاد، ممکن است زمان زودتری به آن نسبت داده شود.

## انواع همگام سازی ساعت

۱- فیزیکی : واقعاً همه ساعت های کامپیوترها یکسان می شوند.
۲- منطقی : لزومی ندارد که ساعت همه یکسان باشد ممکن است ساعت هایشان کم و زیاد باشد ولی وضعیت بگونه ای است که کارشان به درستی انجام می شود.

**Clock Tick (تیک ساعت):**

- هر دفعه یک Interrupt رخ می دهد که باعث می شود شمارنده ساعت یک واحد افزایش یابد.
- یک Counter برای ساعت وجود دارد که به آن Holding Register نیز میگویند و کارش این است که ساعت را نشان میدهد.

## ساعت های فیزیکی

- تمام کامپیوترها داری مداری برای نگهداری زمان هستند (تایمر).
- با یک کامپیوتر و یک ساعت، مهم نیست که این ساعت برای مدتی خاموش شود. چون تمام فرآیندها در ماشین از یک ساعت استفاده می کنند، از نظر داخلی نیز سازگار خواهند بود.
- به محض معرفی چندین کامپیوتر که هر کدام ساعت مخصوص به خود را داشتند، وضعیت کاملاً تغییرخواهد کرد.
- وقتی شبکه دارای n کامپیوتر باشد، تمام n کریستال تا حدودی با نرخ های متفاوت اجرا می شوند و به این ترتیب ساعت ها به تدریج از همگامی خارج می شوند و مقادیر متفاوتی را ارائه می کنند.
- این تفاوت در مقادیر زمان، انحراف ساعت (Clock drift) نام دارد.
- به دلایل افزایش کارایی و افزونگی، وجود چندین ساعت فیزیکی مطلوب است که منجر به دو مسئله می شود:
  ۱- چگونه آن ها را با ساعت های دنیای واقعی همگام کنیم
  ۲- چگونه ساعت ها را با یکدیگر همگام کنیم

## ساعت خورشیدی

- Some Basic definitions:
- Transit of the Sun:
  - Sun rises from east
  - Rises to its max height in the sky (noon)
  - Sun sets in west
- Solar Day: Interval between 2 consecutive transits of the sun (noon-to-noon)
- Solar Day varies due:
  - Core activities of earth
  - Rise & Tide of oceans
  - Gravity
  - Orbit around the sun, not a perfect circle

## ساعت اتمی

- با اختراع ساعت اتمی در سال ۱۹۴۸، مستقل از حرکت و لرزش زمین، اندازه گیری دقیق تر ساعت امکان پذیر شد.
- IAT (International Atomic Time) فقط تعداد میانگین تیک های ساعت ۱۳۳ سزیم از نیمه شب ۱۹۵۸، Jan،۱ (شروع زمان) تقسیم بر ۹۱۹۲۶۳۱۷۷۰ است.
- زمانی بر اساس ثانیه های IAT به نام زمان هماهنگ شده ی جهانی یا UTC خوانده می شود. UTC مبنایی برای تمام ساعت‌های مدرن است.
- UTCجایگزین استاندارد های قدیمی، یعنی زمان میانگین گرینویچ شد که زمان اختر شناسی است.
- کار UTC ایجاد و ارسال پالس ساعت بود و کافی بود هر کسی یک wwv receiver داشته باشد.
- GMT ساعت استاندارد جهانی از نظر فیزیکی است.
- امروزه همه ترجیح می دهند از GMT استفاده کنند.

## سیستم تعیین موقعیت جهانی

- مسئله تعیین موقعیت، از طریق سیستم توزیع شده خاصی به نام GPS حل می شود (GPS سیستم ماهواره ای است).
- GPS از ۲۹ ماهواره استفاده می کند که ماهواره ها بطور پیوسته موقعیت خود را پخش می کنند. و هرکدام در ارتفاع تقریبی ۲۰۰۰۰ کیلومتری می چرخند. هر ماهواره تا چهار ساعت اتمی دارد، که دائماً از ایستگاههای خاصی در زمین تنظیم میشوند. نیاز به حداقل سه ماهواره برای تعیین طول، عرض و ارتفاع یک موقعیت میباشد.
- دو حقیقت در دنیای واقعی وجود دارند که باید در نظر گرفته شوند:

۱- قبل از این که داده های موجود در موقعیت ماهواره به گیرنده برسند، مدتی زمان صرف می شود.

۲- ساعت گیرنده با ساعت ماهواره همگام نیست.

## الگوریتم های همگام سازی ساعت

- اگر یک ماشین دارای گیرنده WWV باشد، هدف این است که تمام ماشین های دیگر با آن همگام شوند.
- اگر هیچ ماشینی دارای گیرندهWWV نباشد، هر ماشین زمان خاص خودش را نگهداری می کند.
- الگوریتم های متعددی برای انجام این همگام سازی پیشنهاد شدند.
- در اینترنت از GMT استفاده می شود.
- استفاده از Time Server یکی از روش هایی است که برای همگام سازی ساعت استفاده می شود.
- اشکال این روش این است که Event Base است یعنی وقتی که ما به اینترنت وصل می شویم این کار را انجام می دهد.

۱- الگوریتم پروتکل زمان شبکه (NTP)

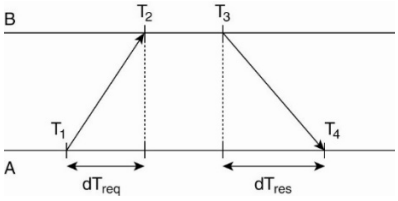
۲- الگوریتم برکلی (Berkeley)

۳- الگوریتم همگام سازی ساعت در شبکه های بی سیم(RBS)

## الگوریتم زمان شبکه (NTP)

- یک روش متداول در بسیاری از پروتکل ها که توسط کریستین(۱۹۸۹) پیشنهاد شد، این است که به سرویس گیرنده ها اجازه داده شود که با سرویس دهنده ساعت تماس بگیرند.
- سرویس دهنده می تواند زمان فعلی را به طور دقیق فراهم کند.
- در سیستم توزیع شده یک کامپیوتر (سرویس دهنده) هست که ساعتش Update است (دارای گیرنده WWV است).
- کاری که انجام می شود این است که هر کامپیوتر در فواصل زمانی خاصی که دوست ندارد بیش از آن Drift rate اش بالاتر برود از کامپیوتر سرویس دهنده زمان درخواست میکند و پاسخ را دریافت میکند.
  - مشکل ۱: زمان به گذشته باز نمیگردد.
  - مشکل ۲: هنگام تماس با سرویس دهنده، تأخیر های پیام، زمان گزارش شده را قدیمی میکند.

## حل مشکل ۲ در الگوریتم زمان شبکه (NTP)

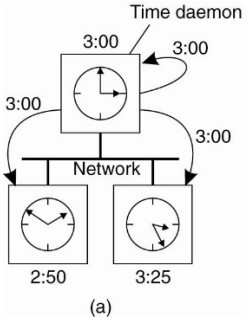



دریافت زمان فعلی توسط A از سرویس دهنده زمان B

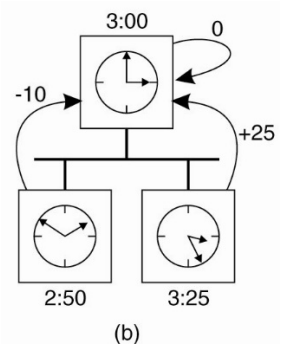
- در NTP ، سرویس دهنده زمان(Time Server) غیر فعال است یعنی شروع کننده نیستند و سایر ماشین ها به طور دوره ای زمان را از آن می پرسند و آن نیز پاسخ می دهد (Client فعال و همیشه شروع کننده است).

## الگوریتم برکلی(Berkeley)

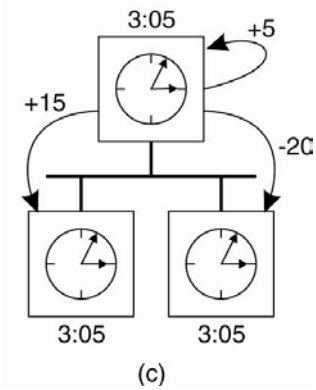
- در اینجا سرویس دهنده زمان، فعال است به طوری که در فواصل منظم از هر ماشین، زمان را می پرسد. بر اساس پاسخ ها، میانگین (میانه) را محاسبه میکند و به تمام ماشین‌ها میگوید ساعت‌های خود را به زمان جدید جلو ببرند (با سریع کردن تیک ساعت) یا ساعت خود را به عقب بکشند! (با کند کردن تیک ساعت).
- این الگوریتم برای سیستمی مفید است که در آن، هیچ ماشینی، گیرنده WWV ندارد.




a) دمون زمان، مقادیر ساعت تمام ماشین ها را سوال میکند.




b) ماشین ها پاسخ میدهند.




C) دمون زمان به هر ماشین میگوید که ساعتش را چگونه تنظیم کند.

- لازم نیست که این زمان با زمان واقعی که هر ساعت توسط رادیو اعلان می شود یکسان باشد.
- الگوریتم برکلی مستقل از کامپایلر و زبان است و در سیستم های ناهمگن نسبت به NTP مناسب تر است.

## ساعت های منطقی

- لامپورت نشان داد که گر چه همگام سازی ساعت امکان پذیر است، لازم نیست مطلق باشد. اگر دو فرآیند تعامل نداشته باشند، لازم نیست ساعت‌های آنها همگام شوند، زیرا مشکلی به وجود نمی آید.
- او اشاره کرد که مهم نیست تمام فرآیندها بر زمان خاصی توافق داشته باشند، بلکه بر روی ترتیب وقوع رویدادها توافق دارند.

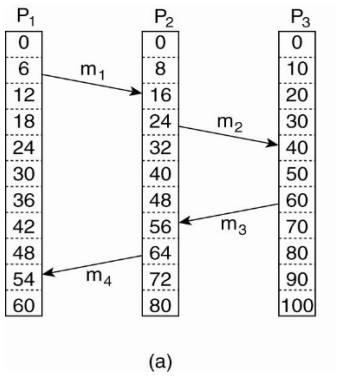
## ساعت منطقی لامپورت

- برای همگام سازی ساعت های منطقی، لامپورت رابطه ای به نام تقدم رویداد (→) happens-before را با شرایط زیر تعریف کرد:

  ۱- در یک ماشین یا فرایند اگر a قبل از b رخ دهد آنگاه $a \rightarrow b$ درست است.

  ۲- در دو ماشین یا فرایند پیام نمی تواند قبل از ارسال، دریافت شود، یا حتی همزمان با ارسال، دریافت شود، زیرا مدتی زمان نیاز دارد تا به گیرنده برسد.
- عبارت $a \rightarrow b$ به این صورت خوانده می شود: a قبل از b رخ می دهد.
- تقدم رویداد، رابطه تراگذری است: لذا اگر $a \rightarrow b$ و $b \rightarrow c$ آنگاه $a \rightarrow c$.
- ساعت در این روش هیچ گاه به عقب بر نمیگردد و تنها به جلو افزایش می یابد، یعنی ساعت هیچ گاه منفی نمیشود.
- سیستمی که ساعتش از همه بیشتر است اگر برای دیگران پیغام بفرستد ساعت همه را بالا می برد.

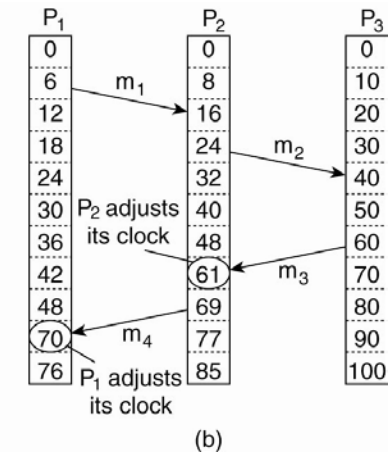
## ساعت منطقی لامپورت - مثال (۱)



(a)

سه فرآیند که هر کدام ساعت خاص خودش را دارد. ساعت ها با تیکهای مختلفی اجرا می شوند.

## ساعت منطقی لامپورت - مثال (۲)




- الگوریتم لامپورت ساعت ها را اصلاح می کند.
- هر فرآیند P(i) یک شمارنده محلی c(i) را نگهداری می کند. این شمارنده ها به صورت زیر به هنگام می شوند:

۱- قبل از اجرای رویداد P(i) رابطه c(i)←c(i) +۱ را اجرا می کند.

۲- وقتی فرآیند P(i) پیام m را به P(j) میفرستد، مهر زمانی m، یعنی ts(m) را پس از اجرای مرحله اول برابر با c(i) قرار می دهد.

۳- به محض دریافت پیام m، فرآیند P(j) شمارنده محلی خود را به صورت c(i) ← max{c(j),ts(m)} تنظیم می کند که پس از آن مرحله اول را اجرا می کند و پیام را به کاربرد تحویل می دهد.

### همگام سازی فرآیند ها(Process)

- برای همگام سازی فرآیندها بحث دو بدو ناسازگاری یا انحصار متقابل Mutual Exclusion مطرح می شود.
- بخش بحرانی نه همزمان نه همروند در اختیار ۲ تا فرآیند نباشد.
- باید اجازه دهیم فقط یک فرآیند از بخش بحرانی استفاده کند در صورتی فرآیند بعد می تواند از بخش بحرانی استفاده کند که فرآیند اول کارش تمام شده باشد.

### الگوریتم های انحصار متقابل توزیع شده:

۱- روش مبتنی بر نشانه (Token based)

۲- روش مبتنی بر اجازه (Permission Based)

### روش مبتنی بر نشانه(Token based)

- از طریق ارسال پیام خاصی بین فرآیند ها انجام می شود که نشانه نام دارد.
- فقط یک نشانه وجود دارد و هر کسی که آن نشانه را دارد می تواند به منبع مشترک دستیابی داشته باشد. پس از اتمام دستیابی، نشانه به فرآیند بعدی فرستاده می شود.
- این روش جلوگیری از بن بست را تضمین می کند.
- عیب عمده این روش این است که، وقتی نشانه مفقود می شود، باید رویه توزیع شده ی دقیقی اجرا شود تا تضمین گردد که نشانه جدیدی ایجاد شده است و این نشانه یکتا است.

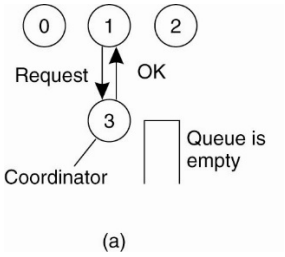
**روش مبتنی بر اجازه(Permission Based)**

- در این روش فرآیندی که منتظر دستیابی به منبع است، ابتدا نیاز به اجازه یک فرآیند هماهنگ کننده یا سایر فرآیندها دارد.
- انواع روشهایی که برای اعطای چنین اجازه ای وجود دارد:
  ۱- الگوریتم متمرکز
  ۲- الگوریتم نامتمرکز
  ۳- الگوریتم توزیع شده

**الگوریتم متمرکز**

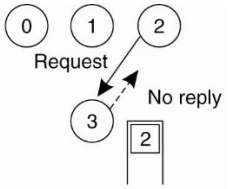
- راحت ترین راه برای رسیدن به انحصار متقابل در سیستم توزیع شده، شبیه سازی چگونگی انجام آن در سیستم تک پردازنده ای است.
- یک فرآیند به عنوان هماهنگ کننده انتخاب می شود. هر وقت فرآیندی میخواهد به منبع مشترکی دست یابد، پیام درخواست را به هماهنگ کننده می فرستد و می گوید به کدام منبع میخواهد دستیابی داشته باشد و کسب اجازه می کند.

**مثال – الگوریتم متمرکز**



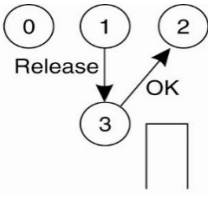
(a)

فرآیند ۱ از هماهنگ کننده اجازه می خواهد تا به منبع مشترک دست یابد. سپس هماهنگ کننده اجازه می دهد.



(b)

فرآیند ۲ اجازه می خواهد تا به همان منبع دست یابد. هماهنگ کننده پاسخ نمیدهد.



(c)

وقتی فرآیند ۱ منبع را آزاد میکند، به هماهنگ کننده اعلان میکند، که آن نیز به فرآیند ۲ پاسخ دهد.

**ویژگی های الگوریتم متمرکز**

- پیاده سازی و مدیریت آن خیلی ساده است. انحصار متقابل را تضمین می کند.
- گرسنگی وجود ندارد.
- هماهنگ کننده، یک نقطه شکست دارد واگر خراب شود، کل سیستم از کار می افتد.
- تنها یک هماهنگ کننده می تواند گلوگاه برای کارایی محسوب شود.

**الگوریتم نامتمرکز**

- یک روش رأی گیری مبتنی بر سیستم DHT است و روش آن این است که هماهنگ کننده مرکزی را بسط می دهد.
- فرض می شود هر منبع n بار تکثیر شده است. هر کپی دارای هماهنگ کننده خاص خودش برای کنترل دستیابی توسط فرآیندهای همزمان است.
- هر وقت فرآیندی میخواهد به منبعی دستیابی داشته باشد،لازم است از $m>n/2$ هماهنگ کننده رأی جمع کند.
- اگر فرایندی کمتر از m رای جمع نماید. دوباره بعد از زمان تصادفی همان کار را تکرار میکند.
- در صورت خرابی هماهنگ کننده و ترمیم آن کلیه رایهای داده شده را فراموش میکند که احتمال وقوع مشکل حتی در صورت رای دادن به فرایند جدید در صورتی که قبلا به دیگری هم رای داده باشد بسیار پایین است.

**ویژگی های الگوریتم نامتمرکز**

- نسبت به الگوریتم متمرکز کمتر دچار آسیب پذیری می شود.
- یک نقطه شکست وجود ندارد.
- گلوگاه وجود ندارد (صف وجود ندارد).

**الگوریتم توزیع شده**

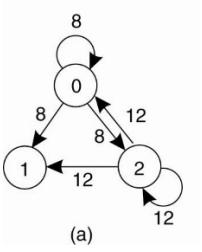
- وقتی فرآیندی میخواهد به منبع مشترکی دست یابد، پیامی میسازد که شامل نام منبع، شماره فرآیند آن و زمان (منطقی) فعلی است. سپس پیام را به تمام فرآیندها گروه خودش ارسال میکند.
- وقتی فرآیندی پیام درخواست را از فرآیند دیگری دریافت میکند، یکی از سه حالت مختلف آینده را انجام می دهد:

۱- اگر گیرنده در حال دستیابی به منبع نباشد و نخواهد به آن دست یابد، پیام OK را به فرستنده ارسال می کند.

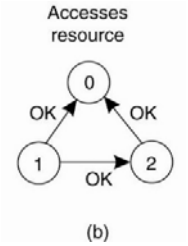
۲- اگر گیرنده به منبع دستیابی داشته باشد، در عوض، درخواست را در صف قرار میدهد.

۳- اگر گیرنده بخواهد به منبع دستیابی داشته باشد ولی هنوز موفق نشده است، مهر زمان پیام ورودی را با مهر زمانی موجود در پیامی که به هر کسی فرستاده است، مقایسه می کند. کمترین مهر زمانی برنده است.

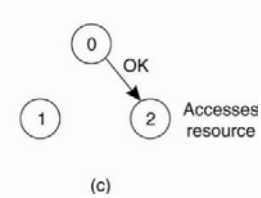
**مثال – الگوریتم توزیع شده**




دو فرآیند می خواهند همزمان به منبع مشترک دستیابی داشته باشند.




فرآیند صفر مهر زمان کمتری دارد، لذا برنده است.



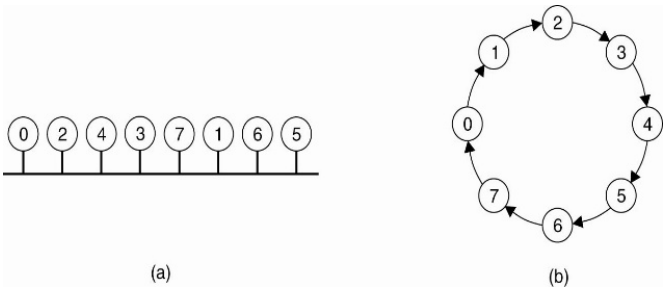

وقتی فرآیند صفر کارش را انجام داد، پیام OK را می فرستد و فرآیند ۲ می تواند به کارش ادامه دهد.

## ویژگی های الگوریتم توزیع شده

سوال: آیا همواره الگوریتم توزیع شده از الگوریتم متمرکز بهتر است؟

- همانند الگوریتم متمرکز، انحصار متقابل بدون بن بست و گرسنگی تضمین می شود.
- n نقطه شکست وجود دارد! و اگر یکی از کار بیفتد همه از کار می افتند.
- حجم پیام ها خیلی بالا است.
- نسبت به الگوریتم متمرکز کندتر، پیچیده تر، گران تر و با توانمندی کمتر است!

## الگوریتم حلقه نشانه




a)گروه نامرتبی از فرآیند ها در شبکه.  b)حلقه منطقی که در نرم افزار ساخته میشود.

- در شکل a یک شبکه خطی داریم (مثل اترنت) که فاقد ترتیب فرآیندها است.
- در شکل b به هر فرآیند مکانی در حلقه داده میشود. موقعیت های حلقه ممکن است به ترتیب عددی آدرس های شبکه یا ابزارهای دیگری تخصیص یابد. مهم نیست که ترتیب چه باشد. مهم این است که هر فرآیند میداند بعد از آن چه فرآیندی قرار دارد.
- وقتی حلقه آماده شد، به فرآیند 0 ، یک نشانه داده میشود. این نشانه در حلقه دور می زند. از فرآیند k به فرآیند k+1 میرود، و بررسی میکند که آیا این فرآیند نیاز به دستیابی به منبع مشترک دارد یا خیر. اگر نیاز داشته باشد، فرآیند اجرا می شود، پس از اتمام کارهایش، منابع را آزاد میکند. پس از پایان کار، نشانه را در ادامه حلقه آزاد می کند.

### ویژگی‌های الگوریتم حلقه نشانه

- در هر زمان فقط یک فرآیند نشانه را در اختیار دارد.
- گرسنگی رخ نمیدهد.
- اگر نشانه مفقود شود، باید دوباره تولید کرد. (تشخیص مفقود شدن نشانه دشوار است)
- اگر فرآیندی خراب شود، الگوریتم دچار درد سر می شود، اما ترمیم آن نسبت به موارد دیگر آسان تر است.

### مقایسه چهار الگوریتم انحصار متقابل

| Algorithm | Messages per entry/exit | Delay before entry (in message times) | Problems |
|---|---|---|---|
| Centralized | 3 | 2 | Coordinator crash |
| Decentralized | 3mk, k = 1,2,... | 2 m | Starvation, low efficiency |
| Distributed | 2 (n – 1) | 2 (n – 1) | Crash of any process |
| Token ring | 1 to ∞ | 0 to n – 1 | Lost token, process crash |

### الگوریتم انتخاب (Election)

- در اغلب الگوریتم های توزیع شده لازم است یک فرآیند به عنوان هماهنگ کننده یا آغاز کننده عمل کند.
- اگر تمام فرآیندها دقیقاً یکسان باشند، هیچ راهی برای انتخاب آنها وجود ندارد.
- بطور کلی، الگوریتم انتخاب سعی میکند فرآیندی با شماره بالاتر را به عنوان هماهنگ کننده انتخاب نماید.

### الگوریتم های انتخاب سنتی

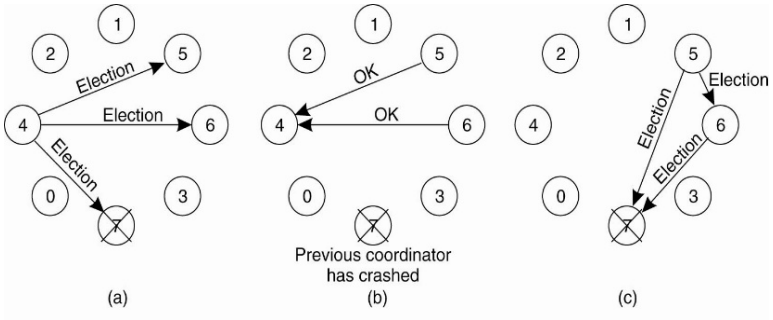
۱-الگوریتم توانمند (Bully Algorithm)  ۲- الگوریتم حلقه (Ring)

### الگوریتم توانمند (Bully)

- همیشه توانمند باقی می ماند.
- وقتی فرآیندی می بیند که هماهنگ کننده به درخواست ها پاسخ نمیدهد، انتخاب را شروع میکند.
- در هر لحظه، فرآیند می تواند پیام ELECTION را از یکی از همکاران با شماره پایین بگیرد. وقتی چنین پیامی میرسد، گیرنده، پیام OK را به فرستنده ارسال میکند تا نشان دهد که زنده است و آن را تحویل خواهد گرفت.

### مثال – الگوریتم توانمند (Bully)

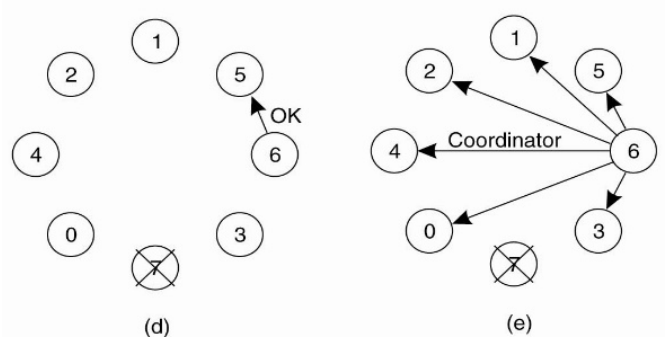
- فرآیند P ، انتخاب را به صورت زیر انجام می دهد:
  - P پیام ELECTION را به تمام فرآیندهایی با شماره بالاتر ارسال می کند.
  - اگر هیچ کدام پاسخ ندهند، P انتخاب را برنده میشود و به عنوان هماهنگ کننده عمل میکند.
  - اگر یکی از آنها پاسخ دهد به عنوان هماهنگ کننده انتخاب میشود و کار P خاتمه می یابد.



a) فرایند ۴ انتخابات را انجام می دهد.

b) فرآیند های ۵ و ۶ پاسخ می دهند و به ۴ می گویند که متوقف شود.

c) اکنون ۵ و ۶ انتخابات را انجام می دهند.

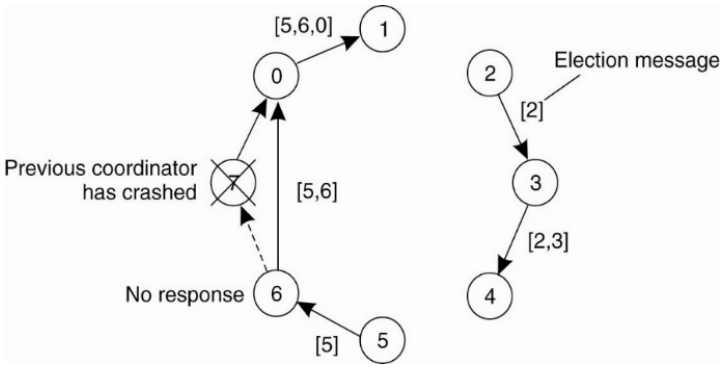



d) فرآیندهای ۶ به ۵ میگوید که متوقف شود.

e) فرآیند ۶ برنده میشود و به همه خبر میدهد.

### الگوریتم حلقه(Ring)

- این الگوریتم از نشانه استفاده نمیکند.
- فرض میکنیم فرآیندها به طور منطقی یا فیزیکی مرتب اند، به طوری که هر فرآیند می داند بعدی او کدام است.
- وقتی فرآیندی متوجه میشود که هماهنگ کننده عمل نمیکند، خودش را کاندیدای انتخاب هماهنگ کننده معرفی میکند یک پیام ELECTION می سازد که شامل شماره فرآیند خودش است و آن را به بعدی خودش می فرستد.
- اگر بعدی او از بین برود، فرستنده از آن عبور میکند و به عدد بعدی در حلقه یا عدد بعد از خودش میرود تا یک فرآیند در حال اجرا پیدا شود.
- سرانجام پیام به فرآیندی بر میگردد که آن را شروع کرده است و فرایند یک پیام COORDINATOR با بیشترین شماره برای ارسال به اعضای حلقه تولید میکند.




الگوریتم انتخاب با استفاده از حلقه. گره‌های ۲ و ۵ متوجه خرابی هماهنگ کننده شده اند و هر دو شروع به ارسال پیام کرده‌اند