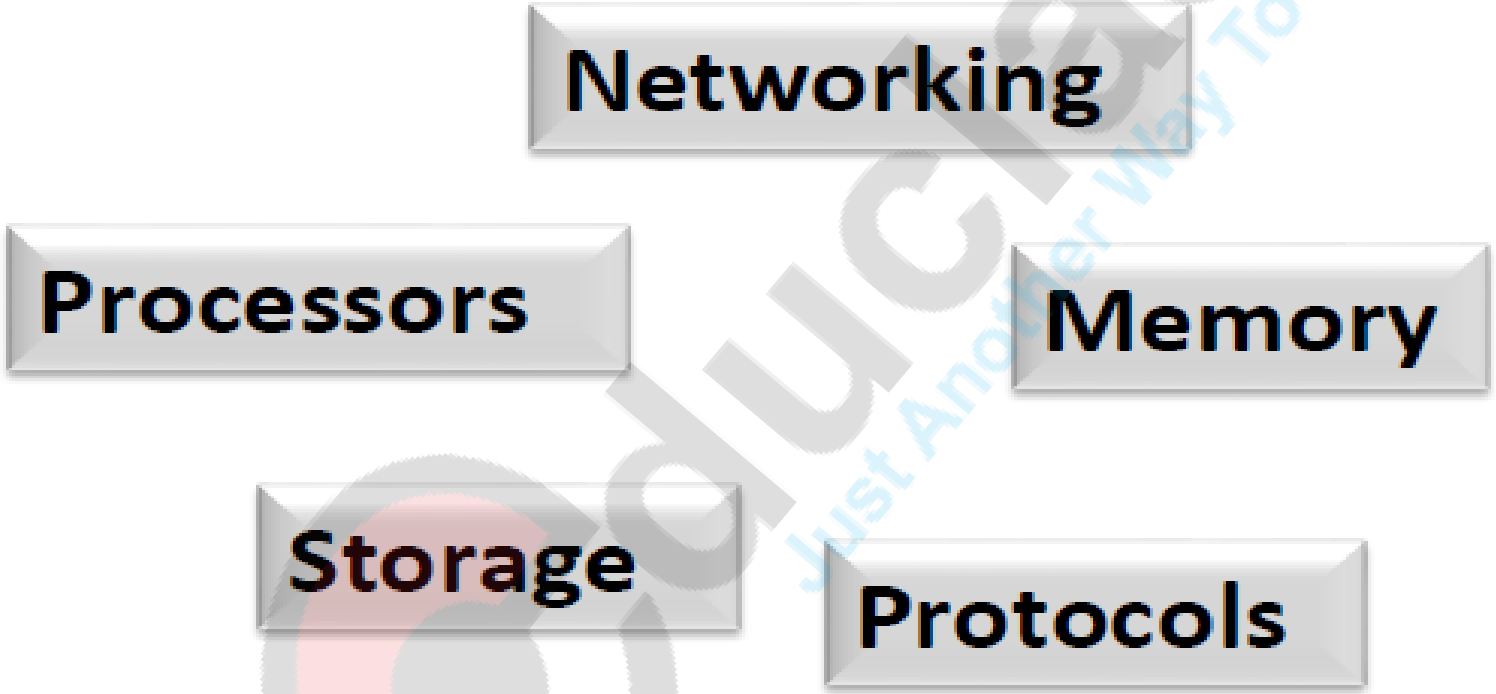
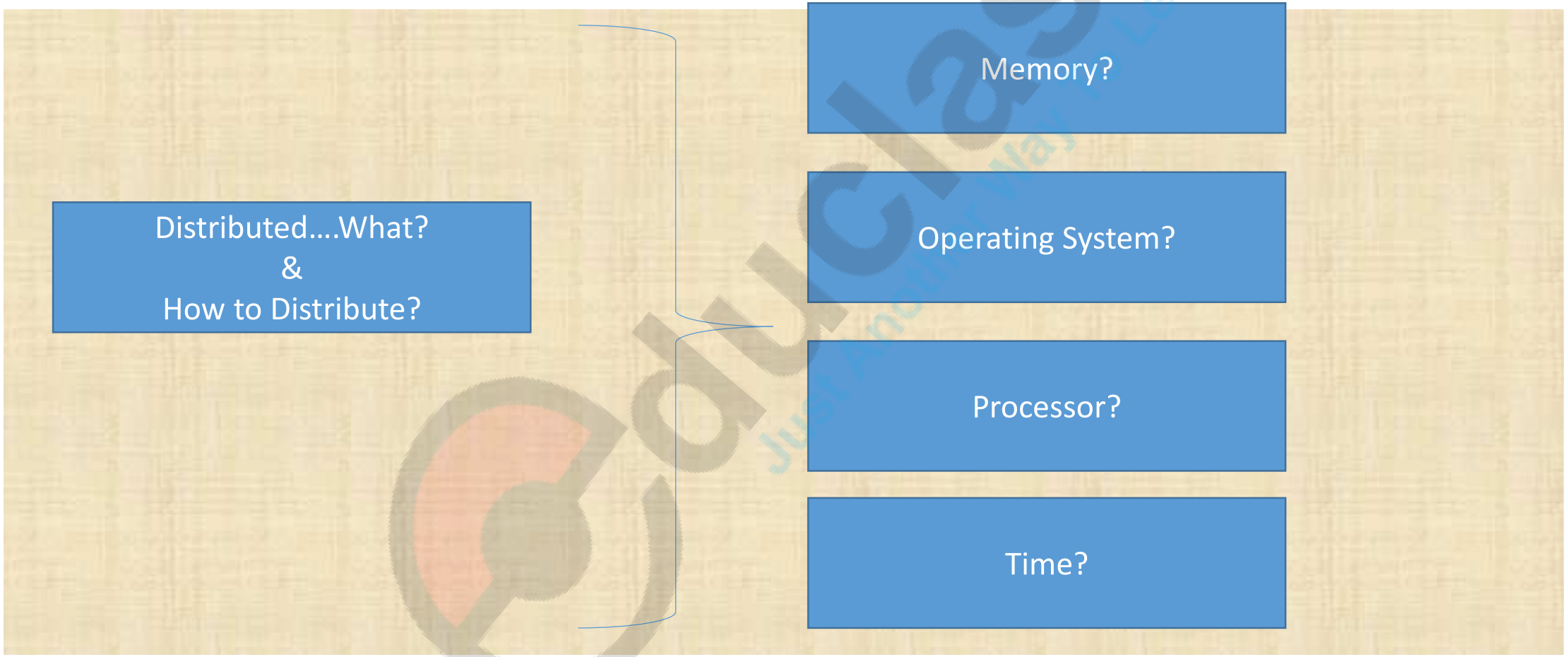
Distributed Computing Concepts

Computers improved……



educlash
Just Another Way To Learn

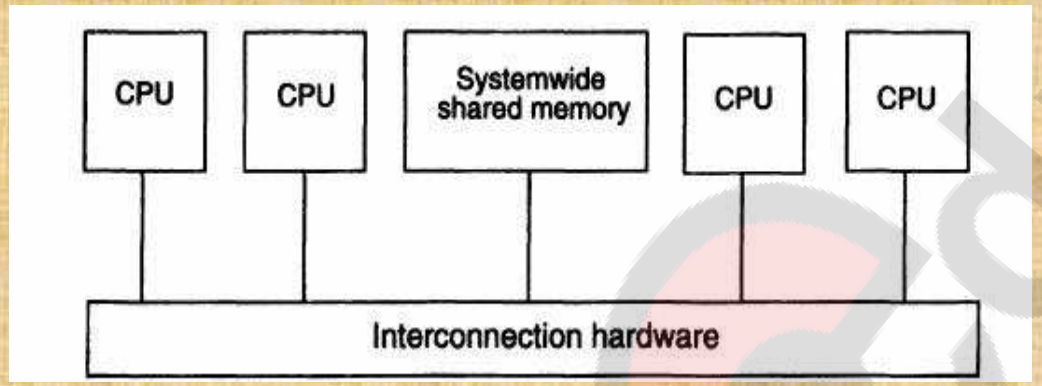
What is Distributed?



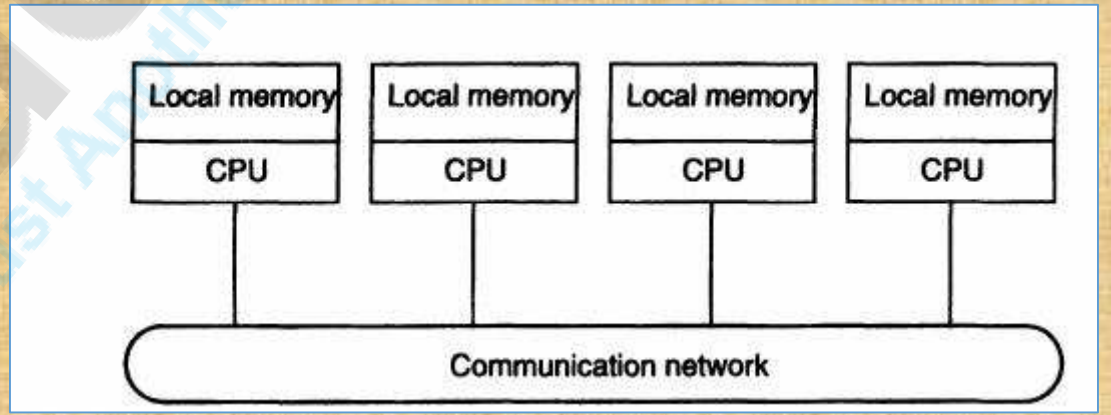
Basic Concept of Distributed System

Computer Architecture

Tightly Coupled → **Parallel Processing System**



Loosely Coupled → Distributed Computing System



Parallel v.s. Distributed Systems

	Parallel Systems	**Distributed Systems**
Memory	**Tightly coupled shared** memory UMA, NUMA	**Distributed** memory Message passing, RPC, and/or used of distributed shared memory
Control	**Global clock** control SIMD, MIMD	**No global clock** control Synchronization algorithms needed
Processor interconnection	Order of **Tbps** Bus, mesh, tree, mesh of tree, and hypercube (-related) network	Order of **Gbps** Ethernet(bus), token ring and SCI (ring), myrinet(switching network)
Main focus	**Performance** Scientific computing	Performance(**cost and scalability**) **Reliability/availability** **Information/resource sharing**

Distributed System..some definitions

"A distributed system is a collection of independent computers that appear to the users of the system as a single coherent system"[Definition by Tanenbaum & Van Renesse]

- "A **distributed system** consists of multiple autonomous computers that communicate through a computer network"
- "Distributed computing utilizes a network of many computers, each accomplishing a portion of an overall task, to achieve a computational result much more quickly than with a single computer"
- "Distributed computing is any computing that involves multiple computers remote from each other that each have a role in a computation problem or information processing"

Architecture

Various hardware and software architectures are used for distributed computing.

At a lower level, it is necessary to inter connect multiple CPUs with some sort of network, regardless of whether that net work is printed on to a circuit board or made up of loosely-coupled devices and cables.

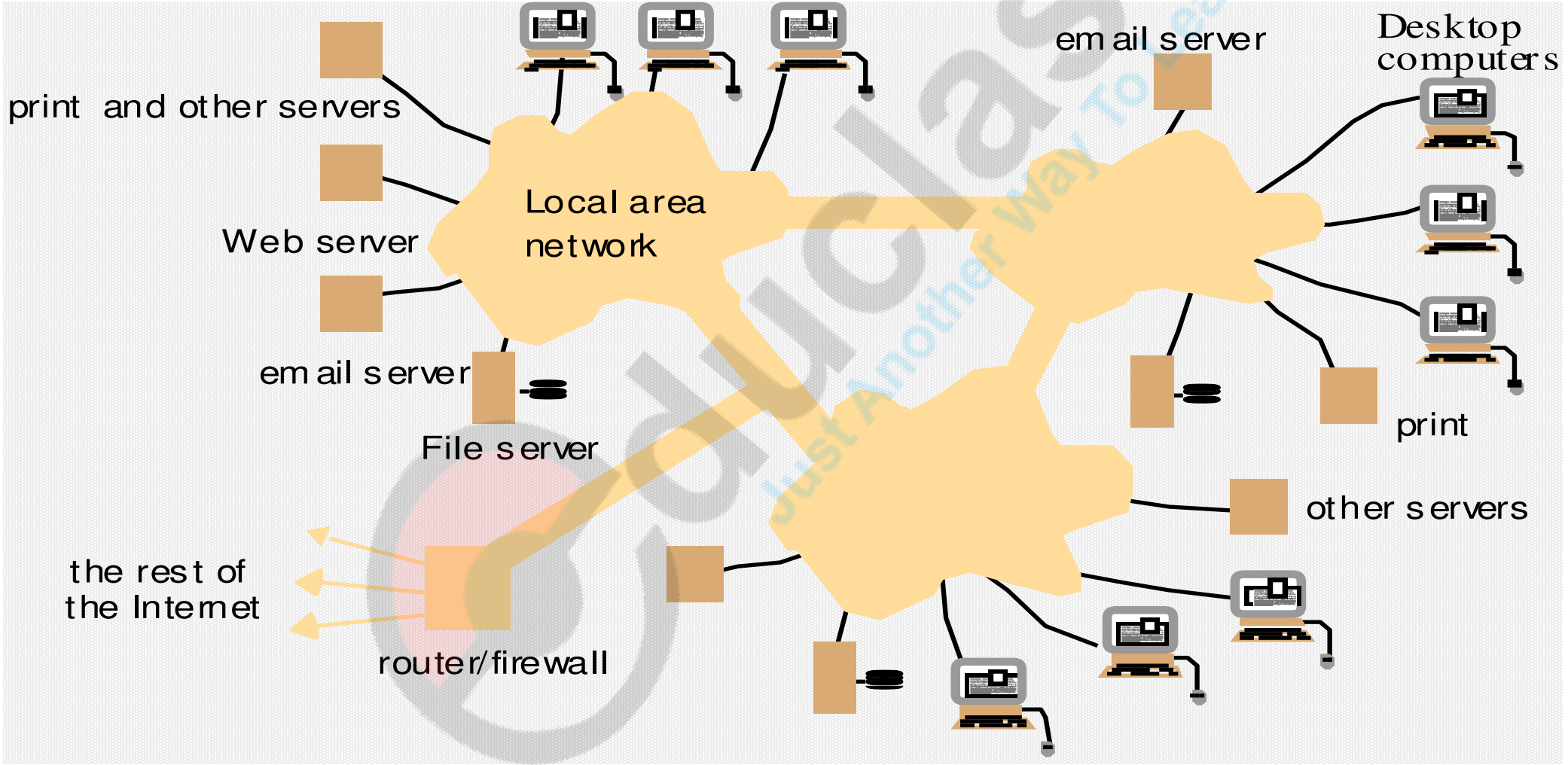
At a higher level, it is necessary to interconnect processes running on those CPUs with some sort of communication system

Examples of Distributed Systems

- Local Area Network and Intranet
- Database Management System
- Automatic Teller Machine Network
- Internet/World-Wide Web
- Mobile and Ubiquitous Computing

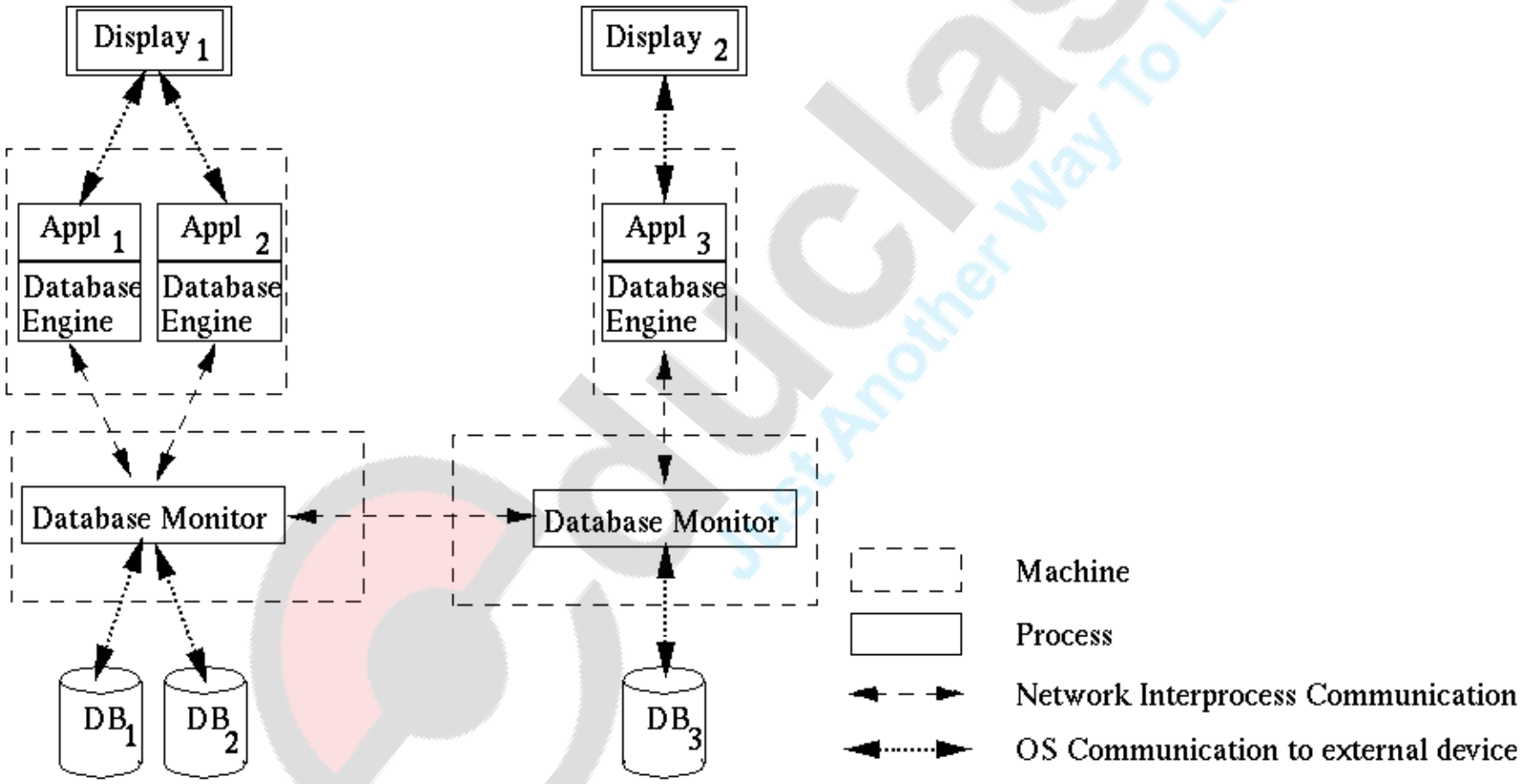
Educlash
Just Another Way To Learn

2.1 Local Area Network



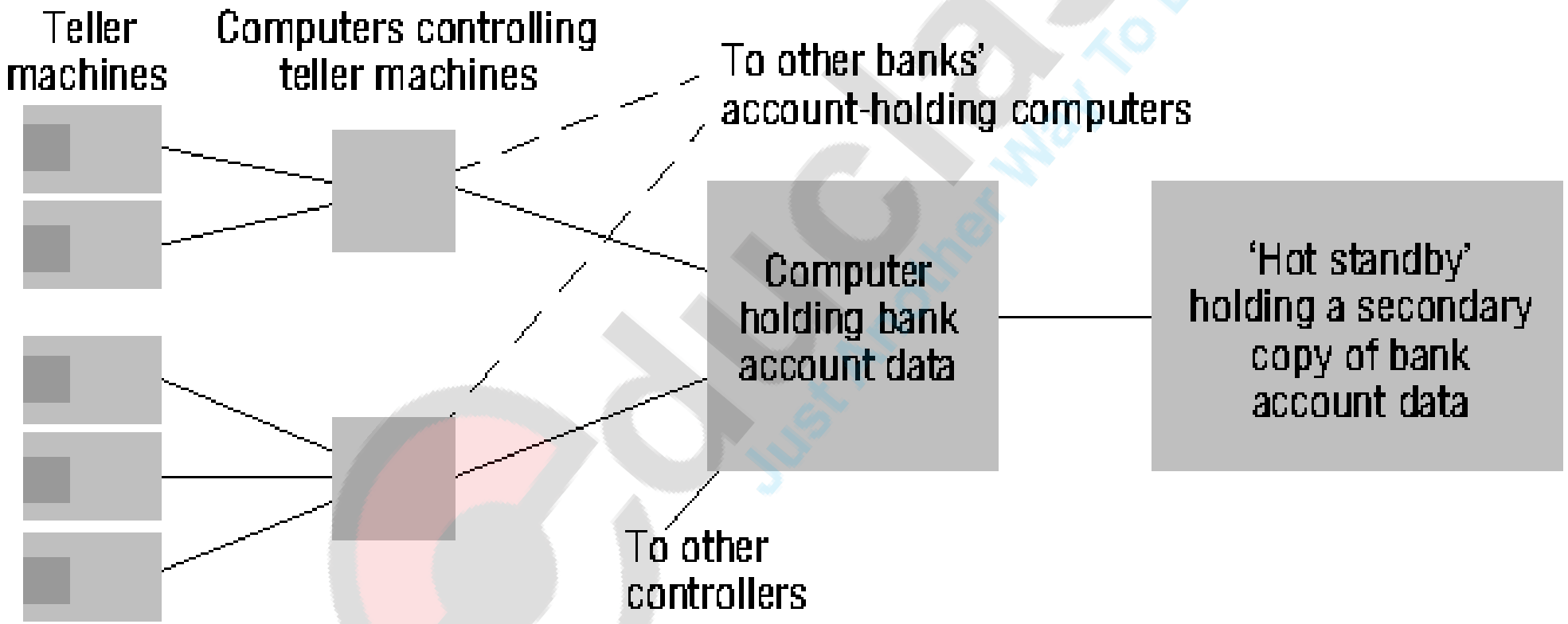
FB/IG/TW: @educlashco

2.2 Database Management System



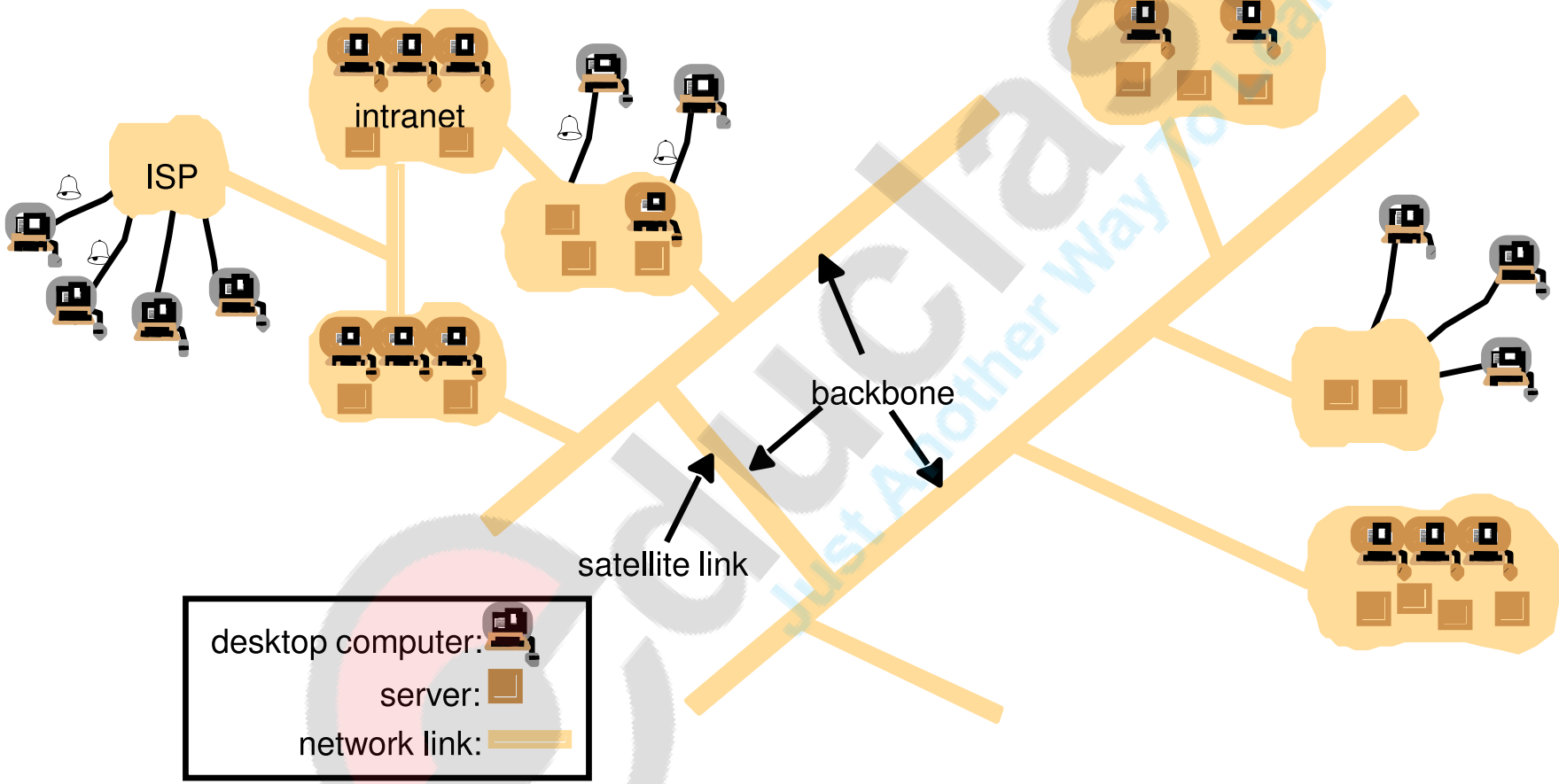
Educlash
Just Another Way To Learn

2.3 Automatic Teller Machine Network



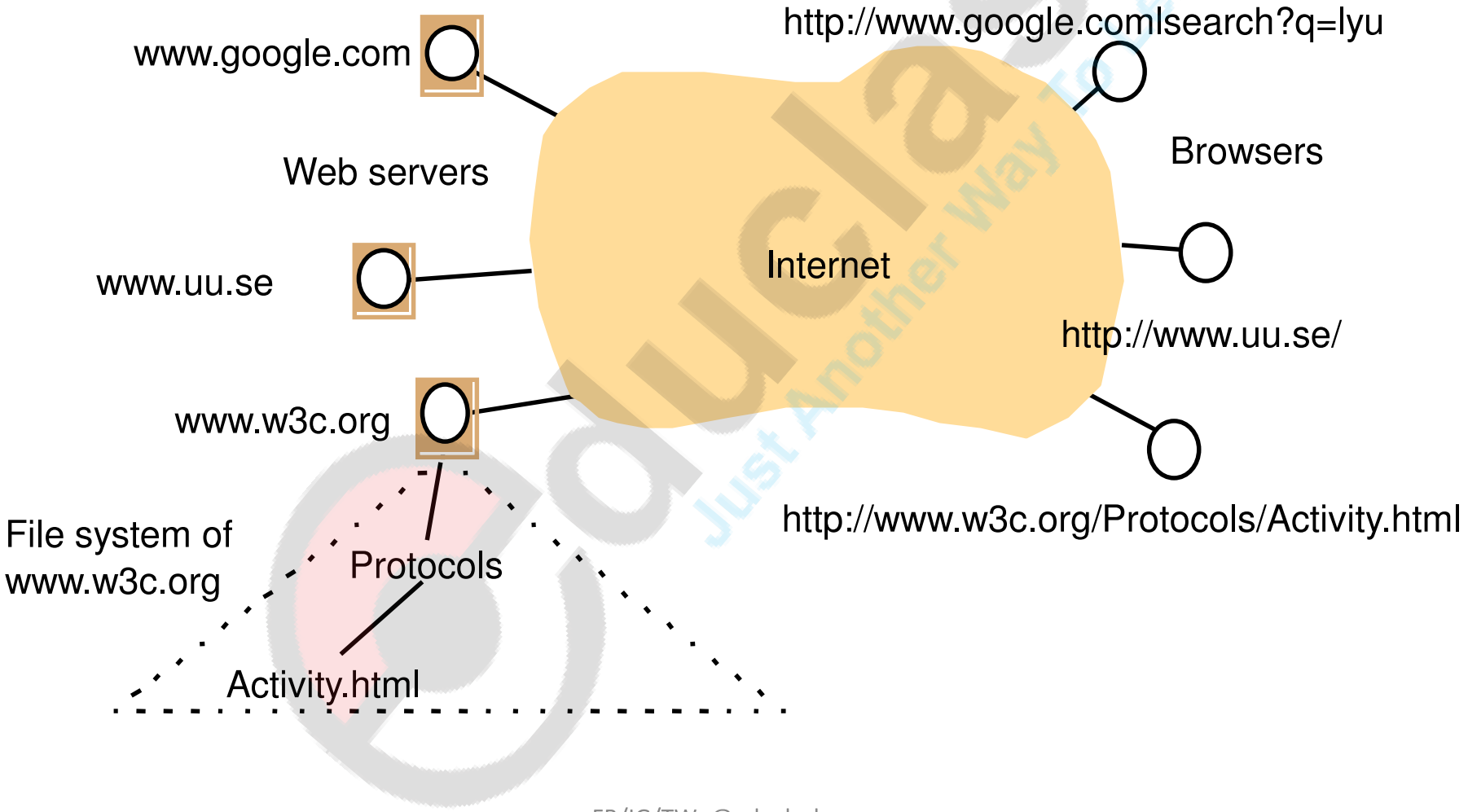
FB/IG/TW: @educlashco

Internet



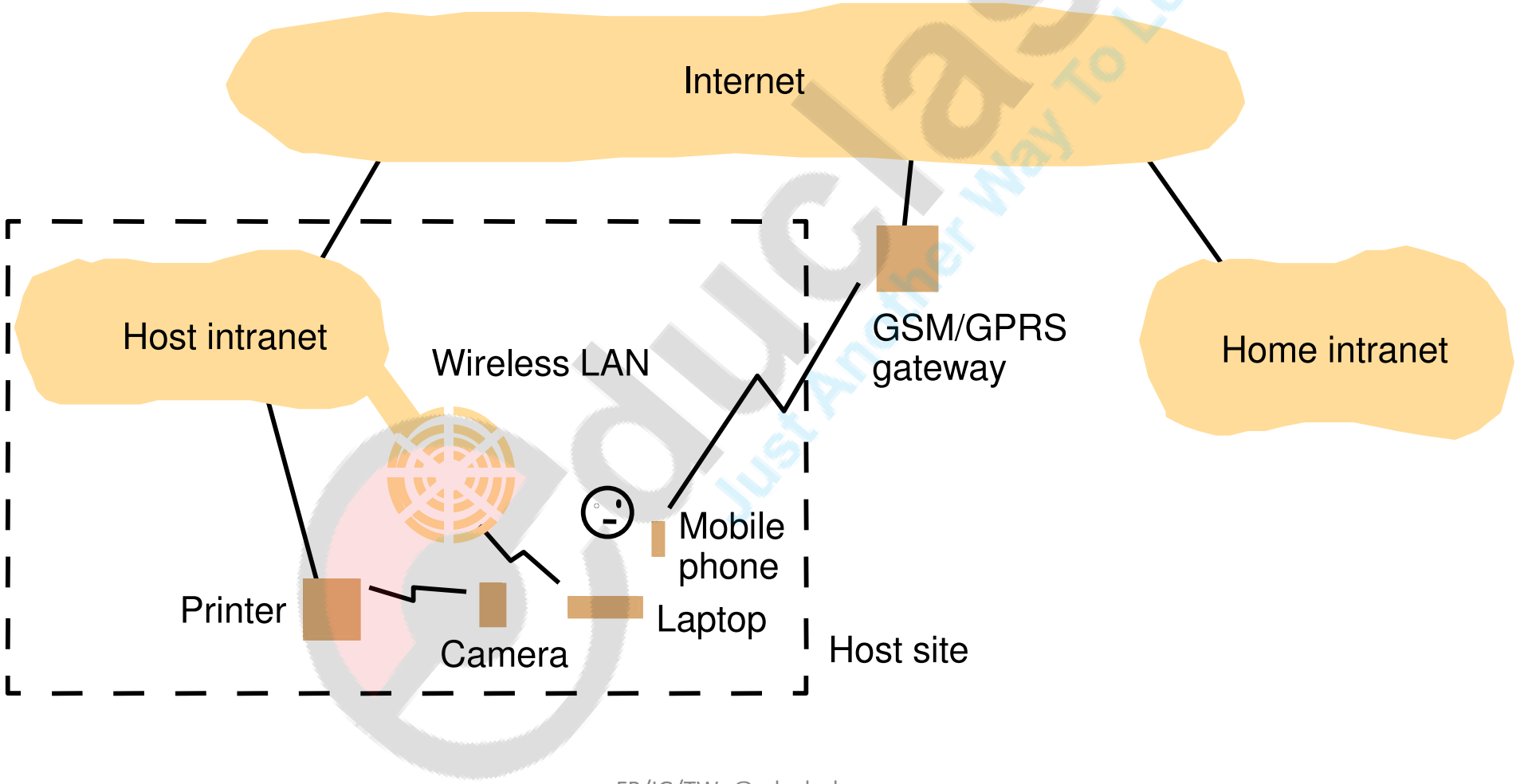
FB/IG/TW: @educlashco

Web Servers and Web Browsers



Eduglash
Just Another Way To Learn

Mobile and Ubiquitous Computing



FB/IG/TW: @educlashco

Distributed Computing Models

Any distributed computing system follows any of the five models

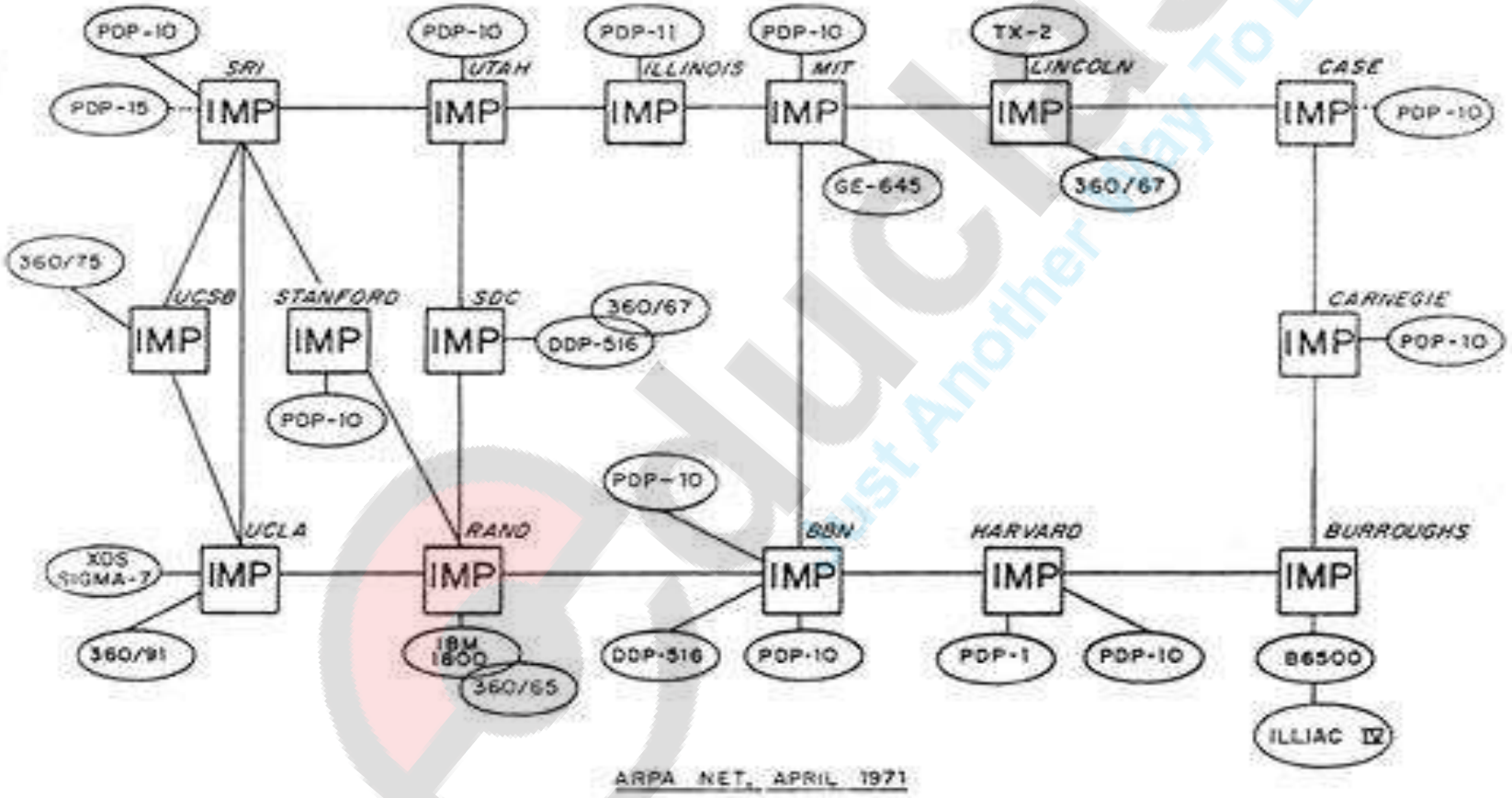
Educlash Just Another Way to Learn

Distributed Computing System Models

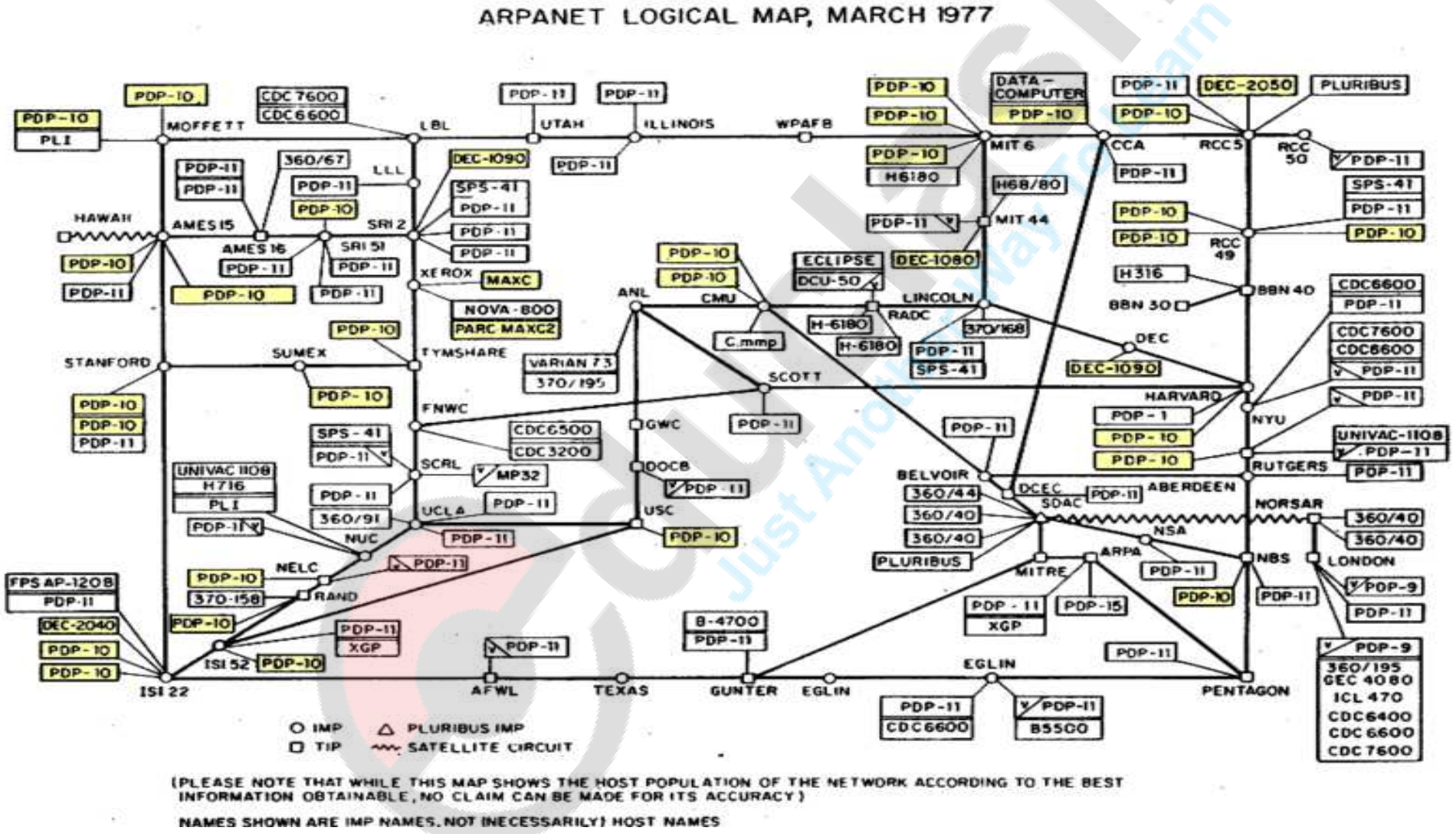
- Minicomputer model
- Workstation model
- Workstation-server model
- Processor-pool model
- Hybrid Model

EDUCLASH
Just Another Way To Learn

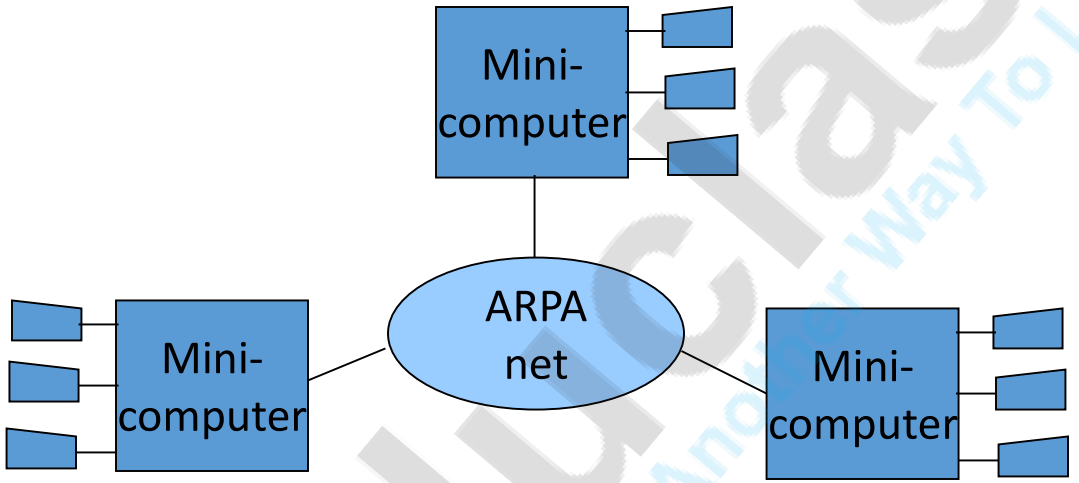
Early ARPA NET(1971)- Case Study
(Mini Computer Model)



ARPANET -1977



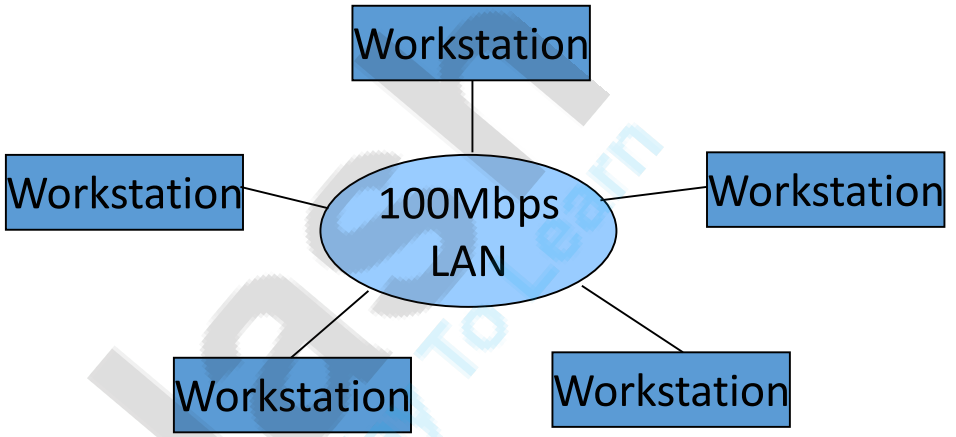
Minicomputer Model



- Extension of Time sharing system
  - User must log on his/her home minicomputer.
  - Thereafter, he/she can log on a remote machine by telnet.
- Resource sharing
  - Database
  - High-performance devices

Educlash
Just Another Way To Learn

Workstation Model



- Workstation with own hard disk and local file system , interconnected over network → diskful workstation
- Process migration
  - Users first log on his/her personal workstation.
  - If there are idle remote workstations, a heavy job may migrate to one of them.
- Problems:
  - How to find am idle workstation [solution: Load Balancing Techniques]
  - How to migrate a job [solution : Load Balancing Techniques]
  - What if a user log on machine ;already remote process running but user initiates a local process(what will happen to remote process)
    [Sol 1: remote and local share resources… But reduces system response]
    [Sol 2:kill remote let local run….all work done till time is waste,leads to file system inconsistencies]
    [sol 3:ask remote to go back to home ..but migration during execution is very costly]

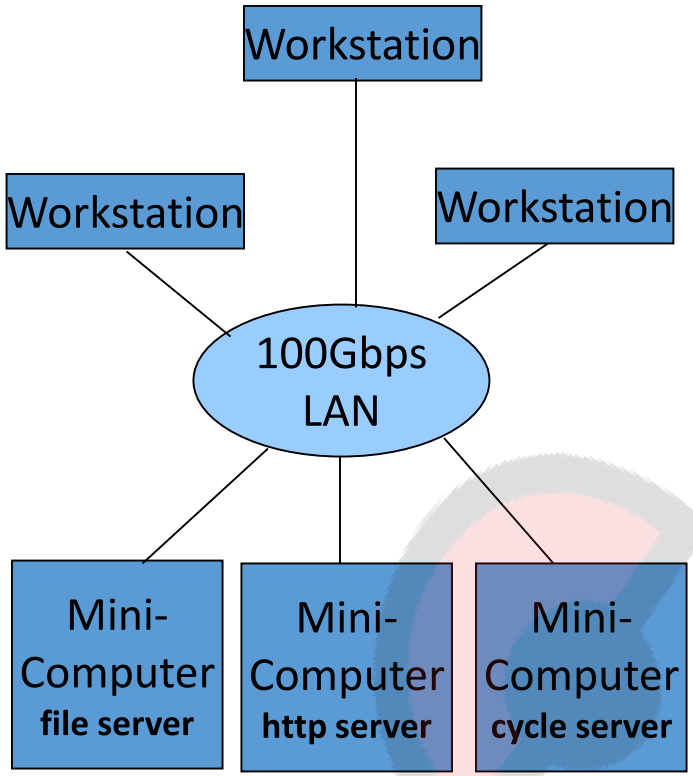
Example:The Sprite System

Early work on Sprite was based on the idea of making the operating system more "network aware", and thereby at the same time make it invisible to the user. The primary area of work was the building of a new network file system which made heavy use of local client-side caching in order to improve performance. After opening the file and some initial reads, the network is only used on-demand, and most user actions occur against the cache. Similar utilities allow remote devices to be mapped into the local computer's space, allowing for network printing and similar duties.

Many of the key Unix files are based on the network, including things like the password file. All machines in a network share the root directory as well. Other common Unix utilities such as finger were re-written to make them network aware as well, listing all of the people logged on across the network. This makes a Sprite network appear as if it were a single large time-sharing system, or a single-system image.

Another key addition to Sprite is process migration, which allows programs to be moved between machines at any time. The system maintains a list of machines and their state, and automatically moves processes to idle machines to improve local performance. Processes can also be "evicted" from machines to improve their performance, causing the original starter to move it to another machine on the network, or take control of it locally again. Long tasks (like compiling the Sprite system) can appear very fast.

Workstation-Server Model



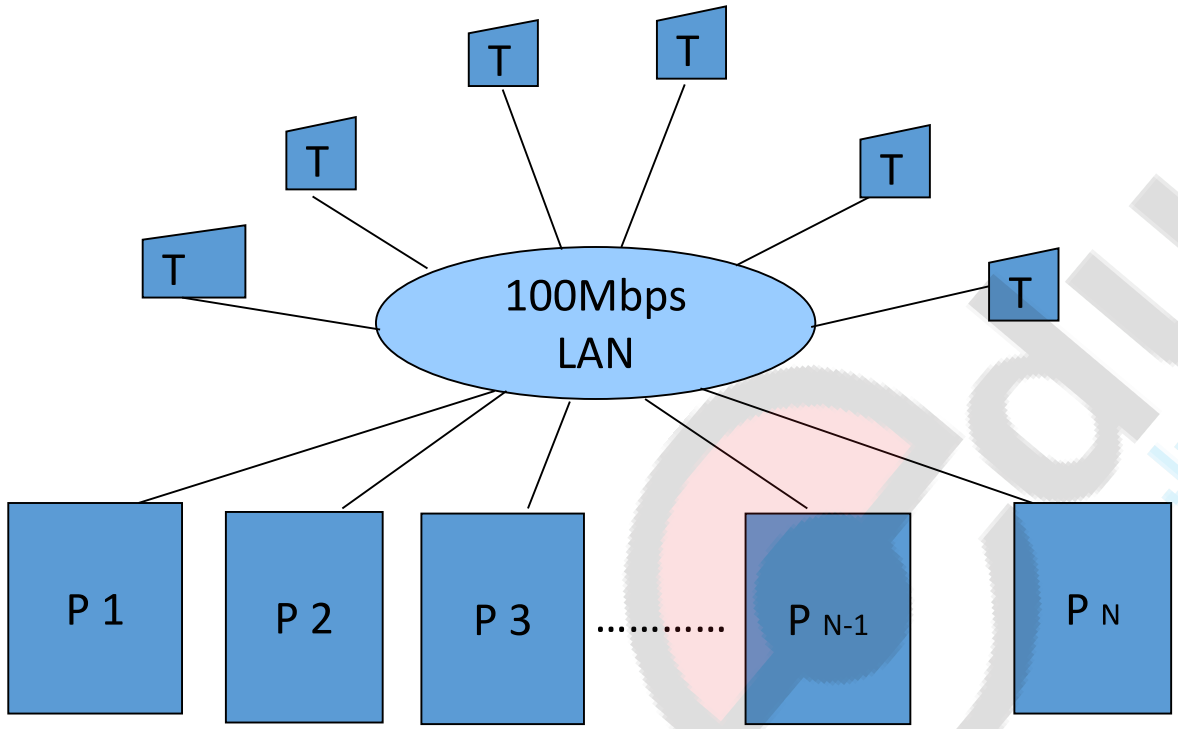
- Work station without own disk , keeps file on server→diskless workstation(Clients)
- More number of less expensive clients and a few costly and powerful servers
  - Graphic/interactive applications (more sophisticated systems possible)
  - Scalabile,reliable,fault tolerant

Just Another Way to Learn

The V-System(Work station server model)

- The key concepts in V are *multithreading* and *synchronous message passing*. The original V terminology uses "process" for what is now commonly called a "thread", and "team" for what is now commonly called a "process" consisting of multiple threads sharing an address space. Communication between threads in V uses synchronous message passing, with short, fixed-length messages that can include access rights for the receiver to read or write part of the sender's address space before replying. The same message-passing interface is used both between threads within one process, between threads of different processes within one machine, and between threads on different machines connected by a local Ethernet. A thread receiving a message is not required to reply to it before receiving other messages; this distinguished the model from Ada rendezvous.
- One common pattern for using the messaging facility is for clients to send messages to a server requesting some form of service. From the client side, this looks much like RPC (remote procedure call) The convenience of an automatic stub generator was lacking, but on the other hand, the client can pass one parameter by reference, which is not possible with other RPC implementations. From the server side the model differs more from RPC, since by default all client requests are multiplexed onto one server thread. The server is free to explicitly fork threads to handle client requests in parallel, however; if this is done, the server-side model is much like RPC too.

Processor-Pool Model



- Pool of processors(powerful micro computers and mini computers)
- Processor with own memory/running system programs/application programs
- No terminals connected directly to processors in the pool
- Users access the processors through terminal(T) via high speed network
- T can be diskless or specialized
- Run server (RS) handles processor scheduling
- If user need ‘x’ processors RS allocates x processors for the job
- No concept of home machine. Process running decision made by RS
- If one machine in the pool fails RS allocates another processor

Plan 9

Plan 9 is a distributed operating system, designed to make a network of heterogeneous and geographically separated computers function as a single system. In a typical Plan 9 installation, users work at terminals running the window system rio, and they access CPU servers which handle computation-intensive processes. Permanent data storage is provided by additional network hosts acting as file servers and archival storage.

Its designers state that:

the foundations of the system are built on two ideas: a per-process name space and a simple message-oriented file system protocol.

Hybrid Model

- Workstation-server model + Processor pool model
  - Workstation-server model best fit → users perform simple interactive task(email,editing , executing small programs)
  - Processor pool Model best fit → Massive Computation
  - Hybrid Model → Workstations + server+ pool of processors
    - Simple process done by workstations
    - Heavy task by processor pool

Issues in Designing Distributed System

EduClash
Just Another Way To Learn

Design Issues

- Transparency
- Flexibility
- Reliability
- Performance
- Scalability
- Fault tolerance

Educlash
Just Another Way To Learn

Issues in designing Distributed Computing System

Educlash
Just Another Way To Learn

Transparency

Transparency: Single system user image

Transparency is defined as the concealment from the user and the application programmer of the separation of components in a distributed system, so that the system is perceived as a whole (virtual uniprocessor) rather than as a collection of independent components

Issues in designing Distributed Computing System

- Eight forms of transparency:
  - Access transparency (user need not know whether the resourse is local or remote, well designed set of system calls)
  - Location transparency (resourse names should be unique systemwide ,name transparency,users should be able to logon from any m/c, user mobility)
  - Concurrency transparency(each user has a feeling that he is the sole user)

    Event ordering property,Mutual exclusion property,

    No starvation property, No deadlock property

Design Issues--Tranparency

- Replication transparency (mapping of the names and replication control are the issues, should be automated)
- Failure transparency(commn. link failure ,m/c failure,storage device crash, cooperation among multiple servers)
- Migration transparency (migration should automatically take place,IPC, no change in name)
- Performance transparency (for load balancing the system should be automatically be reconfigured, intelligent resource allocation & process migration facilities)
- Scaling transparency (allow the sys. to expand without disrupting the acitivities of the user .i;e open system arch.)

Issues in designing Distributed Computing System :Transparency contd..

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource may be shared by several competitive users
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource
Scaling	Hides the expanding property of the system
Performance	Hide the dynamically varying loads by reconfiguring
Persistence	Hide whether a (software) resource is in memory or on disk

Eduklash Just Another Way To Learn

Issues in designing Distributed Computing System:
Reliability

- Reliability:

  For higher reliability , fault –handling mechanisms of a DOS must be designed properly to avoid faults , to tolerate faults, to detect and recover from faults

Educlash
Just Another Way To Learn

Issues in designing Distributed Computing System: **Reliability**

- A fault is a mech.or algo. defect that may generate an error

  When faults occur in hardware or software, programs may produce incorrect results (**Byzantine** difficult to deal with)or they may stop (**fail stop**)before they have completed the intended computation
  - Fail stop
  - Byzantine failure

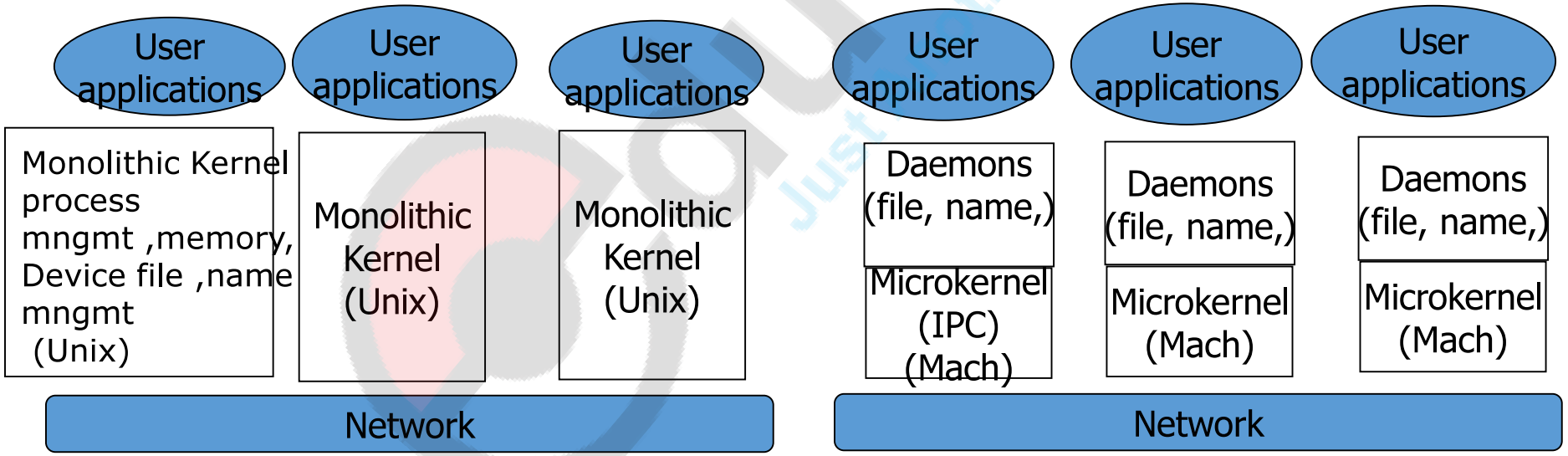
Issues in designing Distributed Computing System: Reliability contd..

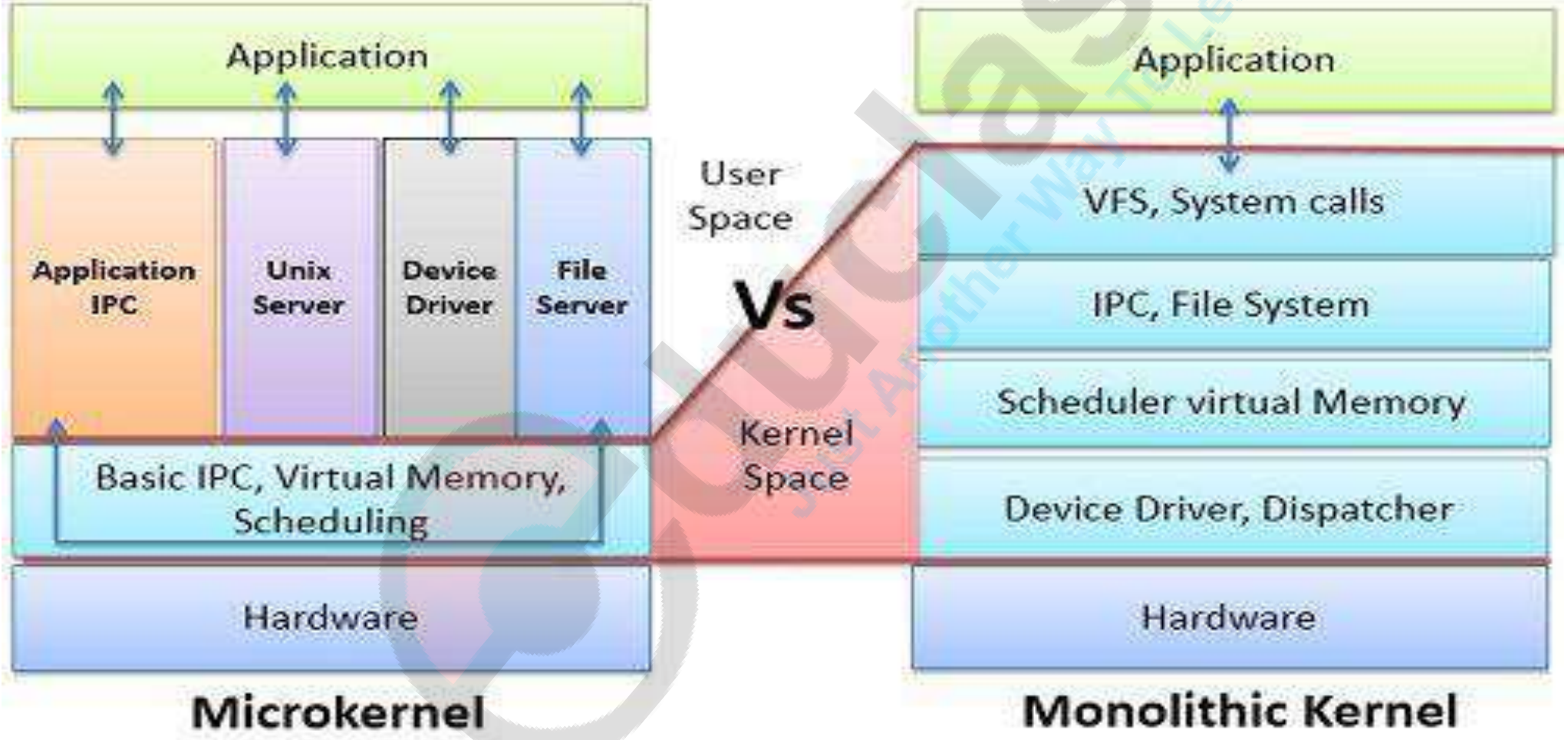
- Fault avoidance
  - The more machines involved, the less avoidance capability, **highly reliable components**
- Fault tolerance
    - Techniques for dealing with failures:
      - Redundancy techniques
        - K-fault tolerance needs K + 1 replicas
        - K-Byzantine failures needs 2K + 1 replicas.
      - Distributed control
        - Avoiding a complete fail stop
- Fault detection and recovery
  - Atomic transaction
  - Stateless servers
  - Acknowledgements and timeout based retransmissions of messages (**IPC,** lost ,duplicate messages)

Just Another Way To Learn

Issues in designing Distributed Computing system :Flexibility

- Ease of modification (bug, replaced , modified)
- Ease of enhancement (added functionalities)
- Easy to port,maintain,install,debug and implement.




Application
Application IPC
Unix Server
Device Driver
File Server
Basic IPC, Virtual Memory, Scheduling
Hardware
Microkernel
User Space
Vs
Kernel Space
Application
VFS, System calls
IPC, File System
Scheduler virtual Memory
Device Driver, Dispatcher
Hardware
Monolithic Kernel

Monolithic and micro kernel comparison

BASIS FOR COMPARISON	MICROKERNEL	MONOLITHIC KERNEL
Basic	In microkernel user services and kernel, services are kept in separate address space.	In monolithic kernel, both user services and kernel services are kept in the same address space.
Size	Microkernel are smaller in size.	Monolithic kernel is larger than microkernel.
Execution	Slow execution.	Fast execution.
Extendible	The microkernel is easily extendible.	The monolithic kernel is hard to extend.
Security	If a service crashes, it does effect on working of microkernel.	If a service crashes, the whole system crashes in monolithic kernel.
Code	To write a microkernel, more code is required.	To write a monolithic kernel, less code is required.
Example	QNX, Symbian, L4Linux, Singularity, K42, Mac OS X, Integrity, PikeOS, HURD, Minix, and Coyotos.	Linux, BSDs (FreeBSD, OpenBSD, NetBSD), Microsoft Windows (95,98,Me), Solaris, OS-9, AIX, HP-

Educlash Just Another Way To Learn
FB/IG/TW: @educlashco

Issues in designing Distributed Computing: Performance

Unlike parallel systems, distributed systems involves OS intervention and slow network medium for data transfer

- Send messages in a batch: chunks of data,piggybacking of ack.of previous msgs with the next msgs)
  - Avoid OS intervention for every message transfer.
- Cache data (at clients site)
  - Avoid repeating the same data transfer
- Minimizing copying of data (moving data in and out of buffers can be reduced :kernel ,I/O devices ,clients ,servers)
- Minimize n/w traffic (migrating a process closer to the recourses)

Issues in designing Distributed Computing: Scalability

Capability of a system to adapt to increased service load

- Avoid centralized entities and algorithms
  - Avoid network saturation.
- Perform most operations on client sides
  - Avoid heavy traffic between clients and servers
  - Hide communication latencies (accessing db using forms)
  - Hide distribution
  - Hide replication

Educlash
Just Another Way To Learn

Issues in designing Distributed Computing system: Heterogeneity

- Different networks, hardware, operating systems, programming languages, developers, platforms,e;g some hosts use 32 bit wordlength while others may use 16 or 64 bit length,flexiblities with users,therefore some form of **data translation is necesary**
- We set up protocols to solve these heterogeneities.
- Middleware: a software layer that provides a programming abstraction as well as masking the heterogeneity.
- Mobile code: code that can be sent from one computer to another and run at the destination.
- Data and instruction formats depend on each machine architecture
- If a system consists of K different machine types, we need K–1 translation software.
- If we have an architecture-independent standard data/instruction formats, each different machine prepares only such a standard translation software.

EduClash Just Another Way To Learn

Issues in designing Distributed Computing system: Security

- Security for information resources has three components:
  - Confidentiality: protection against disclosure to unauthorized individuals.
  - Integrity: protection against alteration or corruption.
  - Availability: protection against interference with the means to access the resources.
- Two new security challenges:
  - Denial of service attacks (DoS).
  - Security of mobile code.

  These issues of security can , most of the time be addressed by cryptography

Educlash Just Another Way To Learn

Software Concept

Educlash
Just Another Way To Learn

Operating System Milestones in Distributed Systems

1950s-1960s	Batch operating system	
1960s	Multiprogramming	
1960s-1970s	Time sharing systems	Multics, IBM360
1969-1973	WAN and LAN	ARPAnet, Ethernet
1960s-early1980s	Minicomputers	PDP, VAX
Early 1980s	Workstations	Alto
1980s – present	Workstation/Server models	Sprite, V-system
1990s	Clusters	Beowulf
Late 1990s	Grid computing	Globus, Legion

Educlash Just Another Way To Learn

Batch operating system

- The users of a batch operating system do not interact with the computer directly. Each user prepares his job on an off-line device like punch cards and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group. The programmers leave their programs with the operator and the operator then sorts the programs with similar requirements into batches.
- The problems with Batch Systems are as follows –
  - Lack of interaction between the user and the job.
  - CPU is often idle, because the speed of the mechanical I/O devices is slower than the CPU.
  - Difficult to provide the desired priority.

Time-sharing operating systems

- Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time.
- Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing.
- The main difference between Multiprogrammed Batch Systems and Time-Sharing Systems is that in case of Multiprogrammed batch systems, the objective is to maximize processor use, whereas in Time-Sharing Systems, the objective is to minimize response time.
- Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, the processor executes each user program in a short burst or quantum of computation. That is, if **n**users are present, then each user can get a time quantum. When the user submits the command, the response time is in few seconds at most.
- The operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems have been modified to time-sharing systems..

Distributed operating System(used in Distributed Computing)

- Distributed systems use multiple central processors to serve multiple real-time applications and multiple users. Data processing jobs are distributed among the processors accordingly.
- The processors communicate with one another through various communication lines (such as high-speed buses or telephone lines). These are referred as **loosely coupled systems** or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, computers, and so on.

Educlash
Just Another Way To Learn

Network operating System(Used in Distributed Computing)

- A Network Operating System runs on a server and provides the server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks.
- Examples : Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

eduClash
Just Another Way To Learn

Distributed OS vs Network OS

Sr. No.	Network Operating System	Distributed Operating System
1	A network operating system is made up of software and associated protocols that allow a set of computer network to be used together.	A distributed operating system is an ordinary centralized operating system but runs on multiple independent CPUs.
2	Environment users are aware of multiplicity of machines.	Environment users are not aware of multiplicity of machines.
3	Control over file placement is done manually by the user.	It can be done automatically by the system itself.
4	Performance is badly affected if certain part of the hardware starts malfunctioning.	It is more reliable or fault tolerant i.e distributed operating system performs even if certain part of the hardware starts malfunctioning.
5	Remote resources are accessed by either logging into the desired remote machine or transferring data from the remote machine to user's own machines.	Users access remote resources in the same manner as they access local resources.

Distributed OS features

- IPC-process communicates globally(local and remote treated equal)
- Uniform Process Management –Process devided and executed.

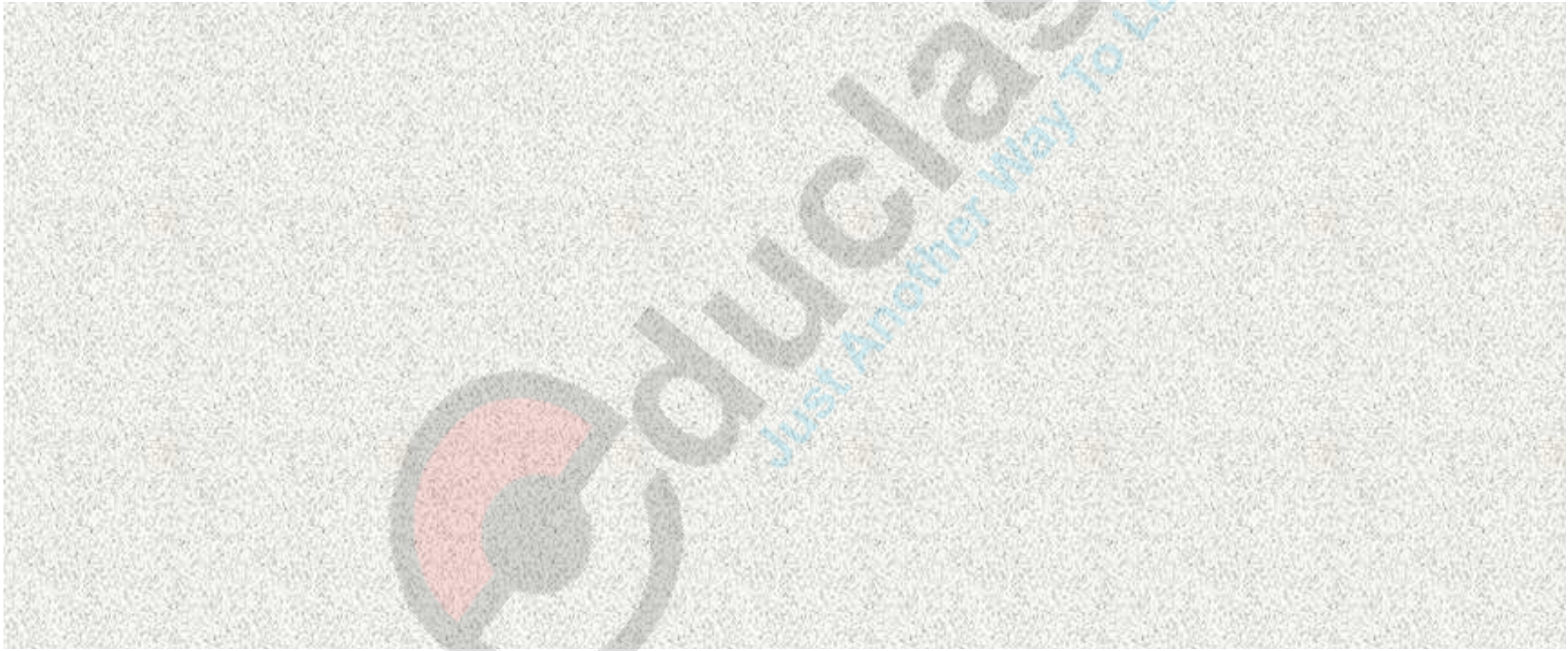
  All computers doing process related tasks→ create,destroy,run,terminate…
- Uniform and visible file system-file system must allow same naming convention
- Identical Kernal Implimentation:Kernels running on all cpu must be identical
- Local Control of machine-Control on its own loal resourse
- Scheduling Issues- CPU must manage its own scheduling+cooperate with global scheduling

Educlash Just Another Way To Learn

Issues in Designing Distributed System

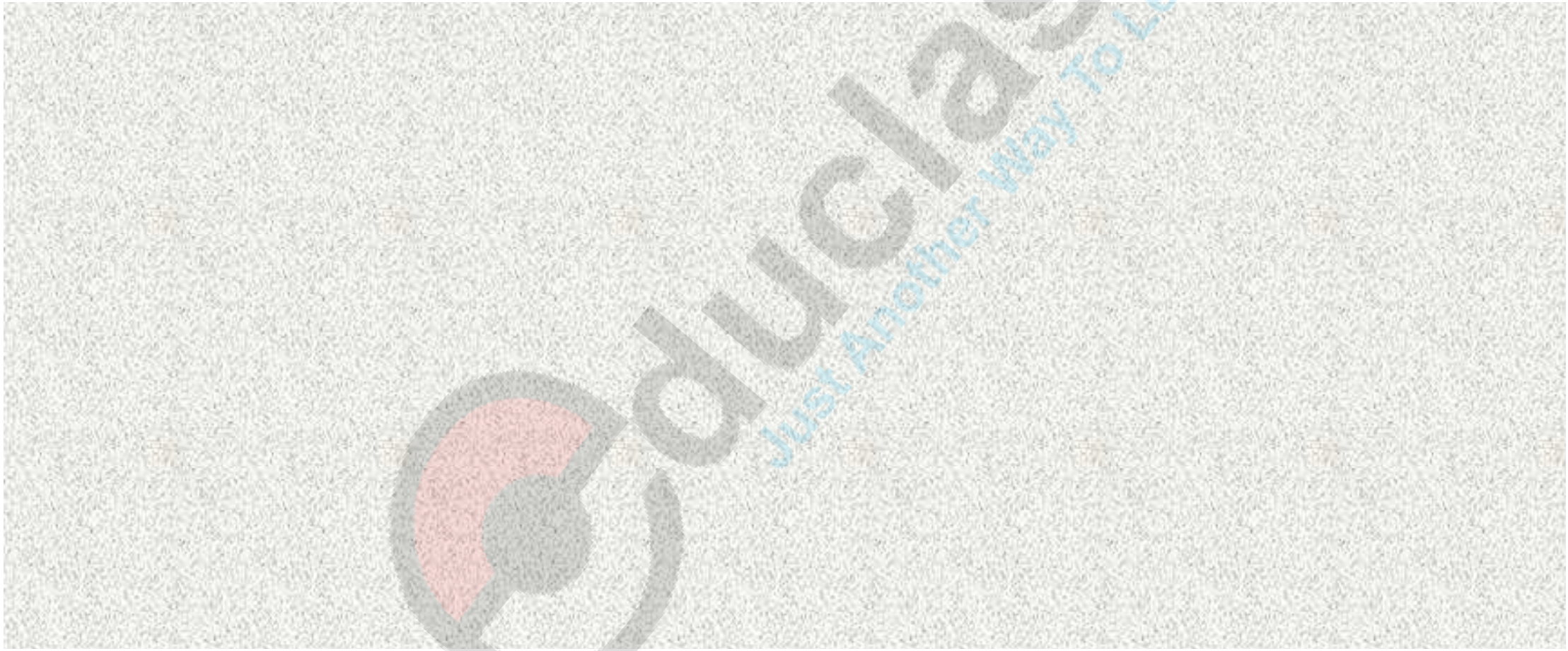
eduClash
Just Another Way to Learn

Transparency

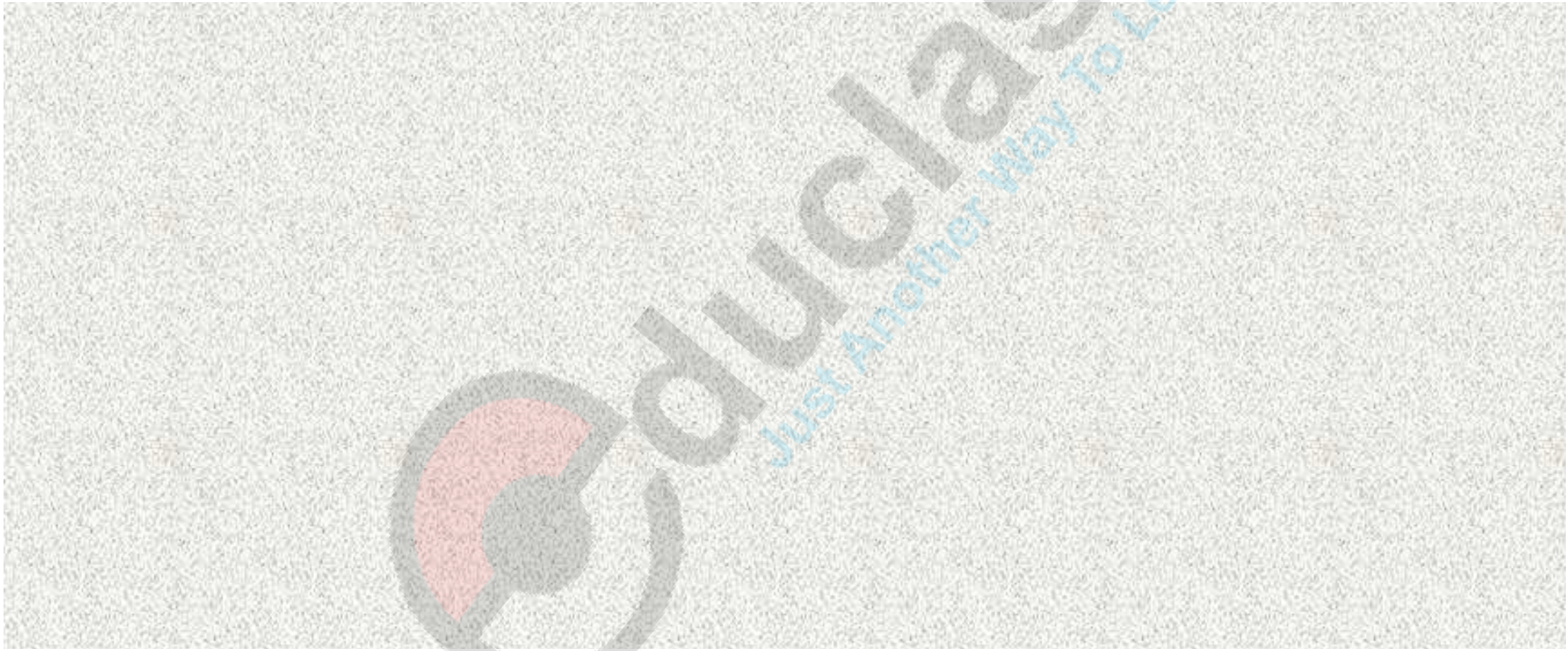


educlash
Just Another Way To Learn

FLexibility

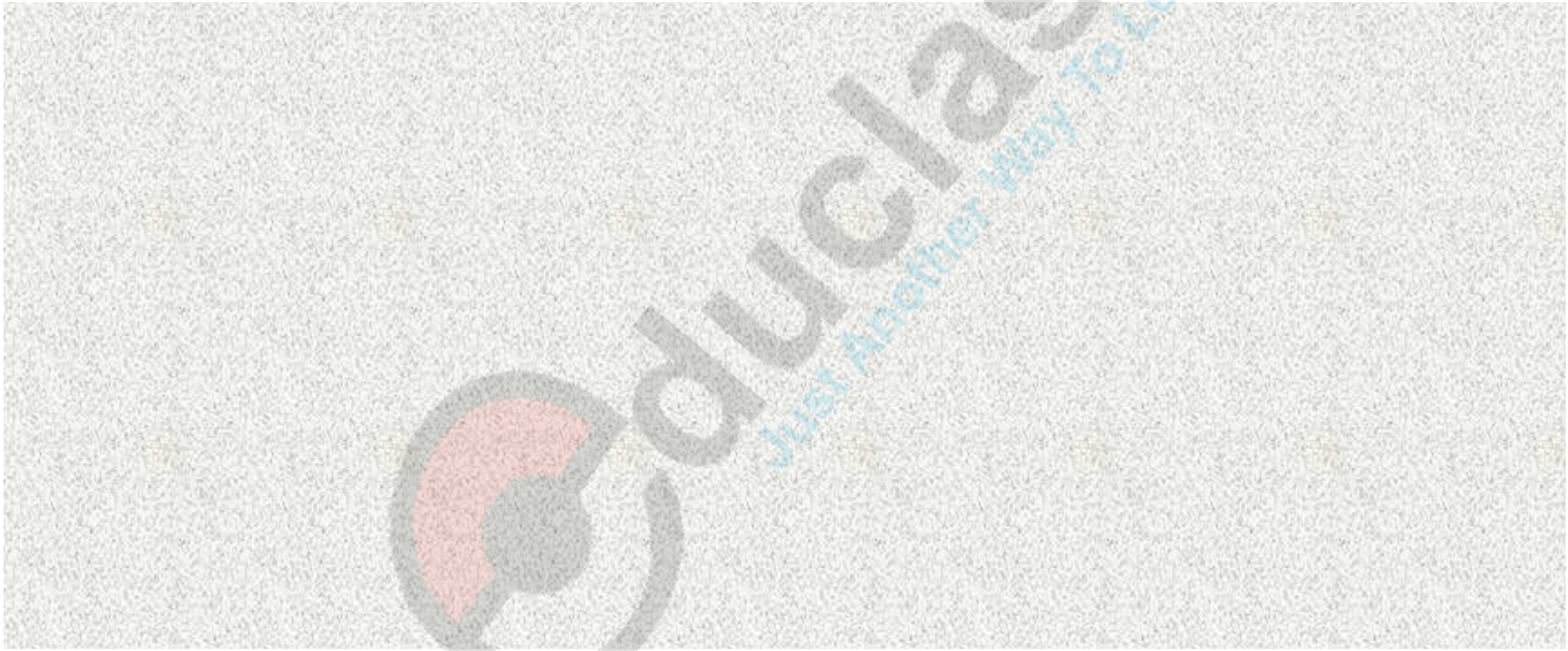


Reliability



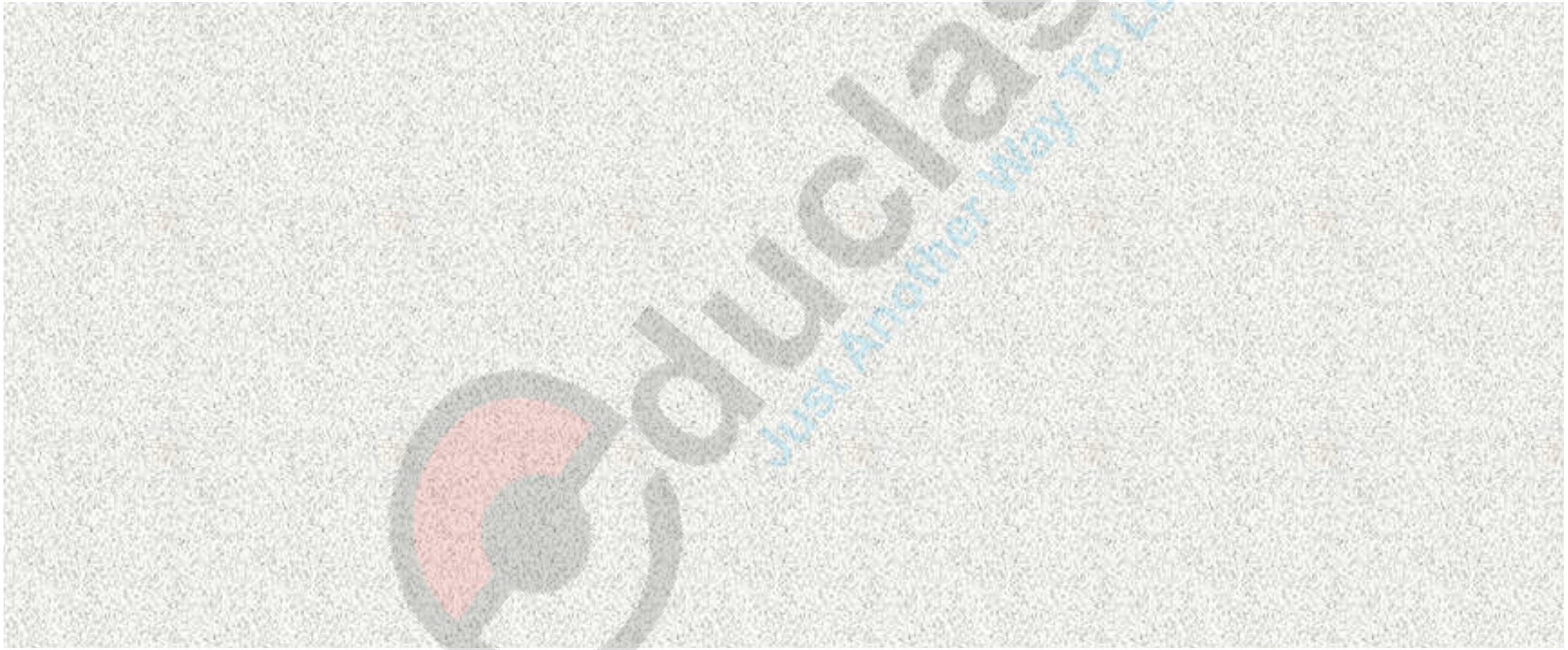
educlash
Just Another Way To Learn

Scalability



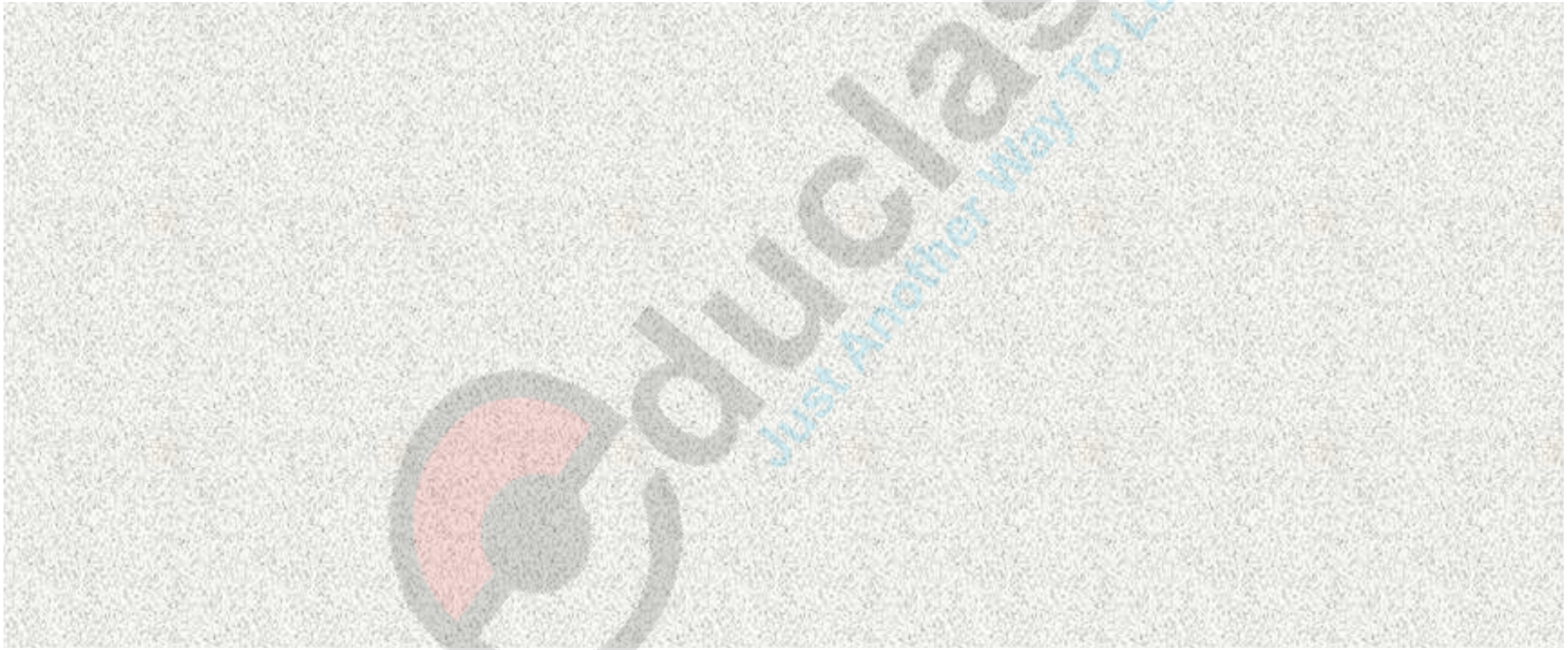
educlash
Just Another Way To Learn

Security



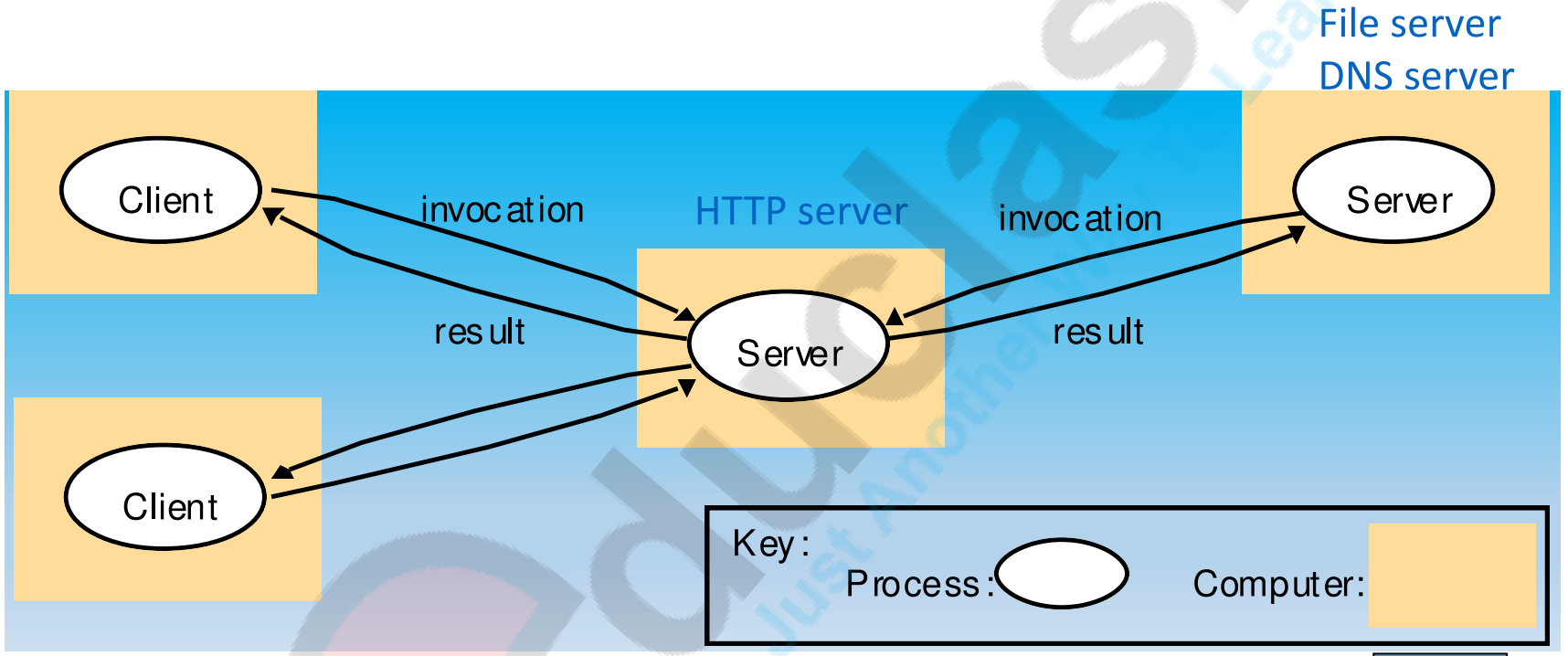
educlash
Just Another Way To Learn

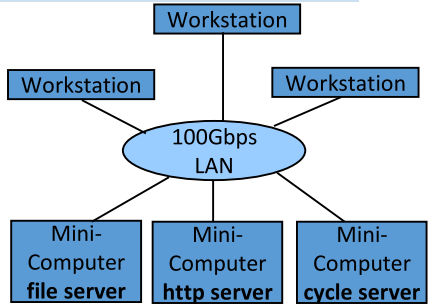
Fault Tolerance



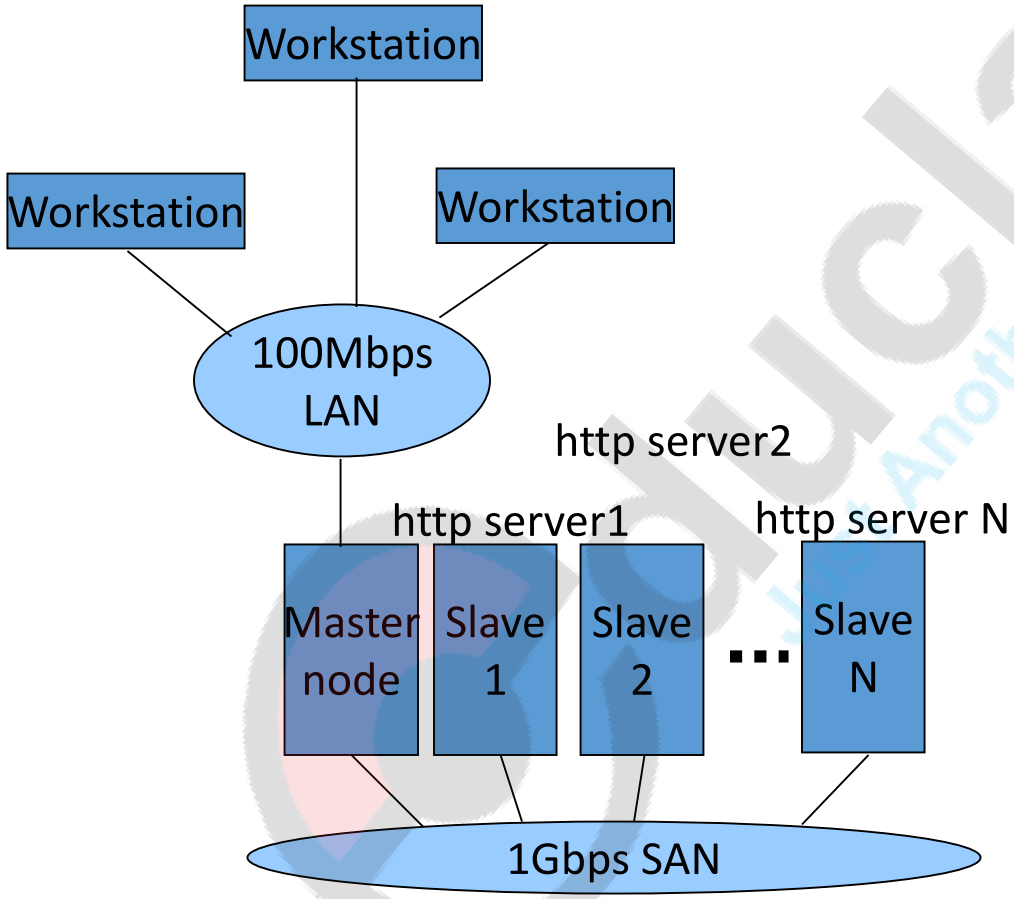
educlash
Just Another Way To Learn

Client-Server Model





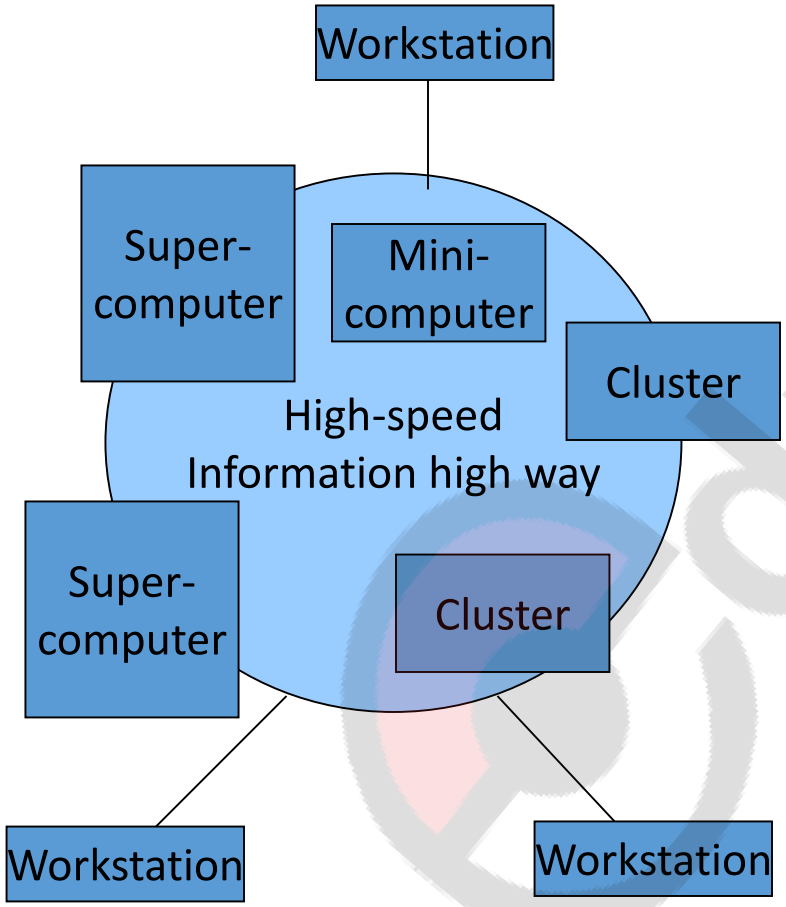
Cluster Model



- Client
  - Takes a client-server model
- Server
  - Consists of many PC/workstations connected to a high-speed network.
  - Puts more focus on performance: serves for requests in parallel.

Educlash
Live Another Way To Learn

Grid Computing



- Goal
  - Collect computing power of supercomputers and clusters sparsely located over the nation and make it available as if it were the electric grid
- Distributed Supercomputing
  - Very large problems needing lots of CPU, memory, etc.
- High-Throughput Computing
  - Harnessing many idle resources
- On-Demand Computing
  - Remote resources integrated with local computation
- Data-intensive Computing
  - Using distributed data
- Collaborative Computing
  - Support communication among multiple parties

Middleware Models

Middleware Models	Platforms
Client-server model	Workstation-server model
Services provided by multiple servers	Cluster model
Proxy servers and caches	ISP server Cluster model
Peer processes	Workstation model
Mobile code and agents	Workstation model Workstation-server model
Thin clients	Processor-pool model Cluster model