



## 1. Computación Grid (Grid Computing)

### Definición:

Conjunto de recursos computacionales **heterogéneos** y **distribuidos** (de distintas organizaciones) **interconectados por WAN o Internet**, funcionando como un **superordenador virtual descentralizado**.

**Objetivo:** **resolver tareas complejas** compartiendo recursos distribuidos.

### Servicios básicos del Grid:

- **SEGURIDAD**
  - Administración de **DATOS**
  - **EJECUCIÓN** de aplicaciones
  - **Notificación** y **monitoreo** del sistema
  - **Implementados** como **servicios web**

### NO es:

- Un clúster
- Una mejora de Internet
- Un proyecto único

### Diferencia con Cloud Computing:

- *Grid*: **recursos distribuidos** y **descentralizados**, los usuarios proveen los recursos.
- *Cloud*: **recursos centralizados**, administrados **por proveedores de nube**.

## 2. Tipos de Grid

1. **Grid de Cómputo**: comparte **capacidad de procesamiento**.
2. **Grid de Datos**: comparte **almacenamiento**; **acceso** transparente.
3. **Grid de Red**: **aprovecha** el **ancho de banda** subutilizado.
4. **Scavenging**: utiliza **computadoras inactivas** (como en la noche).
5. **Grid Multipropósito**: **combina los anteriores**; meta-Grid.

## 3. Arquitectura del Grid (en capas)



1. **Infraestructura:** recursos compartidos (clústeres, sistemas distribuidos).
2. **Conectividad:** protocolos para comunicación segura (TCP/IP, HTTP, GSI).
3. **Recursos:** administración individual (monitoreo, inicialización).
4. **Grupo de Recursos:** coordinación de múltiples recursos.
5. **Aplicación:** las aplicaciones que corren sobre el Grid; pueden interactuar con cualquier capa.



## 4. Globus Toolkit (GT)

Middleware para Grids desarrollado por Globus Alliance. Provee:

- Interfaces estandarizadas
- Componentes modulares (servicios, librerías, herramientas)
- Interoperabilidad

**Componentes clave:**

- **Seguridad (GSI):** usa SSL, certificados X.509 y proxy.
- **Datos:** acceso, transferencia y descubrimiento.
- **Ejecución:** planificación, monitoreo de trabajos.
- **Información (MDS):** monitorea y detecta recursos.
- **Contenedor:** hosting de servicios web (Java, Python, C).

## □ 5. Mini-Grid Implementado

**Características:**

- Tres subredes conectadas por ruteadores.
- Uso de direcciones IP estáticas.
- Seguridad con Simple CA para certificados.

**Red implementada con:**

- Enrutamiento estático
- GT como middleware
- Certificados en cada nodo (host y usuario)



## 6. Herramientas Matlab para Computación Distribuida

- **Parallel Computing Toolbox**: desarrollo en paralelo.
- **Distributed Computing Server (MDCS)**: procesamiento en múltiples nodos.

### Pasos para uso de MDCS:

1. Crear planificador (scheduler)
2. Crear trabajo (job)
3. Crear tareas (tasks)
4. Ejecutar trabajo
5. Obtener resultados

### Administradores disponibles:

- Local
- MathWorks Job Manager
- Terceros (*third-party*)

**Ejemplo:** uso de elementos finitos para electromagnetismo.



## 7. Integración de Cluster Matlab al Mini-Grid

### Proceso:

1. Definir interfaz en **WSDL**
2. Implementar servicio en **Java**
3. Definir parámetros de despliegue (**WSDD**)
4. Compilar y generar archivo **GAR**
5. Desplegar servicio con GT
6. Crear un **cliente** para consumir el servicio
7. Ejecutar la aplicación

**Resultado:** interoperabilidad entre aplicaciones Matlab (Windows) y nodos Linux usando servicios web.



## 8. Aplicaciones y Proyectos de Computación Grid

### Áreas de uso:



- Física de partículas
- Biomedicina y diagnóstico
- Bioinformática
- Observación terrestre
- Química computacional
- Astrofísica

**Beneficios:**

- Colaboración global
- Ejecución a gran escala (miles de nodos)
- Acceso desde cualquier lugar
- Uniformidad e independencia de localización



## 9. Computación Ubicua

**Definición:**

Integración de computadoras en el entorno cotidiano (omnipresente), casi invisibles.

**Factores clave:**

- **Identificación** (ID de usuario)
- **Localización** (geoposicionamiento)
- **Detección** (sensores, eventos)
- **Conexión** (inalámbrica)

**Áreas de investigación:**

- Internet de las cosas (IoT)
- Monitoreo remoto
- Mantenimiento predictivo
- Inventarios
- Seguridad
- Cadena de suministro



## 10. Computación Móvil



### Definición:

El usuario puede **computar en movimiento** usando dispositivos portátiles e inalámbricos.

### Aplicaciones:

- Finanzas
- Inventario
- Servicios de campo
- Localización de productos

### Características:

- **Movilidad:** uso en cualquier lugar.
- **Alcance amplio:** accesibilidad.
- **Ubicuidad**
- **Comodidad**
- **Conectividad instantánea**
- **Personalización**
- **Localización** de servicios



## TEMA 11: TRANSACCIONES EN BASES DE DATOS DISTRIBUIDAS



### ¿Qué es una transacción?

#### Definición general:

- Una transacción es una **secuencia de operaciones** agrupadas como **una unidad lógica**.
- **Se ejecuta** de manera **consistente y confiable** sobre una base de datos compartida.
- **Si alguna parte** de la transacción **falla**, **nada se ejecuta**: es un proceso de **todo o nada**.

**Ejemplo clásico:** Transferencia bancaria: retiro de una cuenta A y depósito en cuenta B → ambos pasos deben ejecutarse juntos.



### Propiedades ACID



1. **Atomicidad:**

- La transacción se ejecuta completamente o no se ejecuta en absoluto.
- Si falla, se revierten todos los cambios (rollback).

2. **Consistencia:**

- La base de datos pasa de un estado válido a otro estado válido.

3. **Aislamiento:**

- La ejecución concurrente no debe interferir con otras transacciones.
- Simula ejecución secuencial.

4. **Durabilidad:**

- Una vez hecha una transacción, sus efectos permanecen incluso ante fallos del sistema.



## Condiciones de terminación

- **Commit:** La transacción termina con éxito y sus efectos se guardan de forma permanente.
- **Abort:** La transacción no finaliza correctamente y todos los efectos se deshacen.
- **Toda transacción debe terminar**, incluso en caso de fallas.



## Tipos de transacciones

1. **Transacciones planas:**

- Inician y terminan con una única estructura (Begin / End).
- No contienen subtransacciones.

2. **Transacciones anidadas:**

- Contienen otras transacciones dentro de sí mismas.
- Conservan las propiedades de sus "padres".



### Reglas de transacciones anidadas

- La transacción **hija debe empezar después** y **terminar antes** que la **padre**.
- El **commit del padre depende** del commit de todas sus hijas.
- Si una **hija aborta**, la transacción **padre también debe abortar**.

### Estados de una transacción

Estado	Descripción
Activa	Se <b>inicia la ejecución</b> .
Parcialmente confirmada	Todas las <b>operaciones se ejecutaron</b> , pero <b>no son permanentes</b> .
Confirmada (Committed)	<b>Finaliza correctamente</b> , los cambios son permanentes.
Fallida	Ocurre <b>un error</b> , se <b>cancela</b> la ejecución.
Terminada	<b>Sale</b> del sistema.

### Bitácora (Log de transacciones)

- Archivo que **registra todas las operaciones para poder rehacer o deshacer** acciones en caso de fallos.
- Garantiza **atomicidad** e **integridad**.

### Primitivas de manejo

Transacciones están delimitadas por:

```
BEGIN TRANSACTION
```

```
-- operaciones
```

```
END TRANSACTION
```






## Estructura del sistema de manejo de transacciones

1. **Manejador de Transacciones:**
  - Valida las peticiones y las pasa al planificador.
2. **Planificador (Scheduler):**
  - Organiza ejecución concurrente de forma secuencialmente equivalente.
3. **Manejador de Datos:**
  - Accede a disco, transfiere datos, ejecuta actualizaciones, gestiona recuperación.

## TEMA 12: REPLICACIÓN Y TOLERANCIA A FALLAS EN SISTEMAS DISTRIBUIDOS

### ¿Qué es la replicación?

- **Replicación:** mantener copias de datos en múltiples computadoras o servidores.
- Usada en servicios web, DNS, bases de datos, proxies, caches, relojes, etc.
- Mejora:
  -  Rendimiento
  -  Disponibilidad
  -  Tolerancia a fallos

### Tipos de replicación

1. **Total:** se replica todo el contenido (objetos/archivos).
2. **Parcial:** solo se replican fragmentos o partes seleccionadas.
3. **No replicada:** los datos se almacenan en un solo sitio.





## ✓ Ventajas

- Mayor disponibilidad.
- Confiabilidad: tolerancia ante fallos de nodos.
- Paralelismo: múltiples nodos pueden atender lecturas simultáneamente.

## ✗ Desventajas

- Costos en actualización y almacenamiento.
- Sobrecarga por mantener consistencia entre réplicas.
- Complejidad del software.

## ↻ Tipos de replicación según enfoque

Tipo	Descripción
Activa	Todos los nodos ejecutan las peticiones en el mismo orden. Alta consistencia.
Pasiva	Solo el nodo primario ejecuta, los demás actualizan después. Menos mensajes, pero más vulnerable.

## 📈 Mejora de rendimiento

- Caching: almacenar resultados anteriores en memoria.
- Reduce latencia, especialmente útil en datos inmutables.

## 🛡️ Alta disponibilidad

- Replicar servidores mejora disponibilidad:
  - Fórmula:  $\text{Disponibilidad} = 1 - p^n$  (p: prob. de fallo; n: cantidad de réplicas).
- Puede haber pérdida de consistencia aunque el sistema esté disponible.

## ⚠️ Tolerancia a fallos



- **Fallos bizantinos:** comportamiento arbitrario o malicioso. Se necesita  $3f + 1$  réplicas para tolerar  $f$  fallos.
- **Objetivo:** servicio continúa funcionando sin que el cliente perciba fallos.



## Requisitos de replicación

- **Transparencia:** el cliente no debe notar que hay réplicas.
- **Consistencia:** todas las copias deben comportarse como una sola (one-copy serializability).



## Tipos de ordenación de peticiones

1. **FIFO:** orden de llegada por cliente.
2. **Causal:** respeta relaciones de dependencia entre peticiones.
3. **Total:** todos ven las operaciones en el mismo orden.



## Niveles de replicación

1. **Caching (cliente):** reduce acceso remoto. No requiere consistencia estricta.
2. **Replicación (servidores):** datos redundantes persistentes. Requiere consistencia.



## Modelos de replicación



### Modelo síncrono

- Actualización atómica y ordenada en todas las réplicas.
- Alta consistencia, pero baja eficiencia y disponibilidad en actualizaciones.



### Modelo asíncrono

- Cada nodo procesa localmente y propaga actualizaciones después (gossip).
- Alta disponibilidad, pero menor consistencia (eventual).



### Copia primaria



- Solo **un nodo acepta escrituras**. Resto son réplicas **esclavas** (solo lectura).



### Protocolo de quórum

- Lectura y escritura sobre subconjuntos ( $r, w$ ) de réplicas.
- Condiciones:
  - $r + w > N \rightarrow$  asegura al menos una copia actualizada.
  - $w > N/2 \rightarrow$  evita conflictos entre escrituras.



## Caso de estudio: DNS

- **Caching:** cada servidor guarda en caché respuestas anteriores.
- **Replicación:** cada zona debe estar replicada en al menos 2 servidores.
- **Tipos:**
  - **Primario:** fuente original.
  - **Secundario:** se sincroniza con el primario.
- **Problemas:** pueden surgir inconsistencias si no se actualizan correctamente.



## Sistemas destacados



### Arquitectura “cotilla” (Gossip-based)

- Se responde rápido al cliente sin esperar replicación completa.
- Replicación "perezosa" a los demás nodos.
- Ideal para disponibilidad, no para transacciones críticas.



### Sistema Bayou

- Propagación eventual de actualizaciones.
- Técnicas de resolución de conflictos por aplicación.
- Aplicaciones: sistemas desconectados o colaborativos.



## Conceptos clave a recordar

- **Linealizabilidad:** orden de operaciones coherente con tiempo real.



- **One-copy serializability:** todas las réplicas actúan como una sola base de datos.
- **Transparencia + Consistencia + Replicación = sistema fiable.**



## TEMA 13: SEGURIDAD EN SISTEMAS DISTRIBUIDOS



### Introducción

- Un sistema distribuido = múltiples computadoras trabajando como si fueran una sola.
- La seguridad es **crítica** debido a que:
  - Se transmite información por **redes inseguras**.
  - Los recursos están distribuidos y expuestos.
- **Objetivos clave de la seguridad:**
  - **Confidencialidad**
  - **Integridad**
  - **Disponibilidad**
  - **Autenticación**
- **Amenazas comunes:**
  - Interceptación, modificación, suplantación, denegación de servicio (DoS).



### Criptografía

**Propósito:** proteger la confidencialidad e integridad de los datos.

#### Tipos:

- **Simétrica (AES):**
  - Misma clave para cifrar y descifrar.
  - Rápida, pero difícil de compartir claves de forma segura.
- **Asimétrica (RSA, ECC):**



- Clave pública para cifrar, privada para descifrar.
- Usada en HTTPS, VPN, firmas digitales.
- **Funciones hash:**
  - Verifican integridad (ej. SHA-256).
  - Aplicaciones: blockchain, firmas digitales.

**Herramientas comunes:** OpenSSL, GnuPG, NaCl.

---

## Autenticación

**Verifica identidad de usuarios o sistemas.**

**Métodos:**

- Contraseñas
- Certificados digitales (X.509)
- Tokens (JWT, OAuth 2.0)
- Biometría (huellas, rostro)

**En sistemas distribuidos:**

- Kerberos, LDAP, OAuth 2.0, SSO (Single Sign-On)

**Ejemplo práctico:** OAuth permite que una app acceda a tu cuenta de Google sin ver tu contraseña.

---

## Confianza en sistemas distribuidos



- La confianza **no se puede asumir por defecto**.
- **Modelos:**
  - Red confiable (limitado)
  - **Zero Trust Architecture (ZTA):** "Nunca confíes, siempre verifica".
    - Uso de certificados, políticas estrictas y comportamiento previo.

---

## ✓ Autorización y control de acceso

Define qué puede hacer un usuario una vez autenticado.

**Técnicas:**

- **ACL:** Listas de control de acceso.
- **RBAC:** Control basado en roles.
- **ABAC:** Control basado en atributos (más dinámico).

**Herramientas:** Keycloak, Open Policy Agent (OPA)



## Monitoreo y detección de intrusos

**Objetivo:** detectar comportamientos maliciosos o anómalos.

**Tecnologías:**

- **IDS:** Sistemas de Detección de Intrusos
- **SIEM:** Gestión de eventos e información de seguridad

**Herramientas conocidas:**



- Snort, OSSEC, Wazuh, Splunk



## Técnicas actuales de seguridad

- Zero Trust
- Blockchain (para confianza distribuida)
- MFA (Autenticación multifactor)
- Cifrado homomórfico (permite operaciones sobre datos cifrados)
- IA en seguridad (detección de amenazas)



## Herramientas de seguridad destacadas

Herramienta	Función Principal
Wireshark	Análisis de tráfico
Metasploit	Pentesting (pruebas de penetración)
Keycloak	Gestión de identidad y acceso (IAM)
Vault	Gestión de secretos (tokens, claves)
Wazuh	IDS y monitoreo de seguridad



## Casos reales de seguridad en SD

1. SolarWinds (2020):
  - Ataque a cadena de suministro.



- Comprometió agencias gubernamentales y empresas privadas.
- Falló la verificación del código fuente y actualizaciones.

## 2. Microsoft Exchange Server (2021):

- Vulnerabilidad que permitió ejecución remota de código.
- Impactó a miles de organizaciones.

## 3. Cloudflare (2022):

- Migración a **Zero Trust** para proteger su infraestructura global.



## Actividad y análisis sugerido

### Preguntas a desarrollar:

1. ¿Qué desafíos de seguridad tienen los sistemas distribuidos frente a los centralizados?
2. Diferencia entre autenticación y autorización.
3. ¿Qué es el modelo Zero Trust? ¿Por qué es relevante hoy?
4. ¿Cómo ayuda Keycloak a mejorar la seguridad?
5. Resumen del ataque SolarWinds: qué falló y qué aprendimos.
6. ¿Qué técnica sería prioritaria en una universidad? ¿Por qué?



## Conclusión





- La seguridad en sistemas distribuidos **requiere un enfoque integral y proactivo**.
- Se deben considerar múltiples capas: cifrado, control de acceso, monitoreo, confianza.
- Modelos como **Zero Trust** y herramientas como **Keycloak, Snort o Wazuh** son clave.
- Casos reales como SolarWinds muestran que **la seguridad no es opcional**.



## TEMA 13 – SEGURIDAD EN SISTEMAS DISTRIBUIDOS (2024A)

---



### 1. ¿Qué es la Seguridad Informática?

Es la capacidad de un sistema para **resistir ataques** y proteger información y recursos, tanto de accesos maliciosos como accidentales.



#### Principios fundamentales:

- **Confidencialidad:** Solo acceden personas autorizadas.
  - **Integridad:** Solo personas autorizadas pueden modificar información.
  - **Disponibilidad:** La información debe estar accesible cuando se necesite.
  - **Vinculación / No repudio:** Nadie puede negar haber realizado una acción.
- 



### 2. Tipos de ataques

#### Por tipo de efecto:

- **Interrupción:** Destrucción o indisponibilidad de recursos.
- **Intercepción:** Acceso no autorizado (ej. sniffing).



- **Modificación:** Alteración de contenido por terceros.
- **Fabricación:** Envío de mensajes falsos.

#### Por naturaleza:

- **Pasivos:** No alteran, solo espían (difíciles de detectar).
- **Activos:** Alteran la comunicación (suplantación, reenvío, modificación, degradación).

Diferencia **DoS vs DDoS**: el primero es un atacante, el segundo involucra muchos (ataque distribuido).



### 3. Servicios de Seguridad

- **Confidencialidad**
- **Autenticación**
- **Integridad**
- **Vinculación (No repudio)**
- **Control de acceso**



### 4. Mecanismos de seguridad

Mecanismo	Función
Intercambio de autenticación	Verifica identidad
Encriptado	Oculto el contenido
Firma digital	Prueba de autoría y autenticidad



---

Control de acceso	Restringe acceso no autorizado
Tráfico de relleno	Oculto patrones
Control de enrutamiento	Define rutas seguras

---

## 5. Firma digital

- Garantiza:
    - **Autenticidad** del autor
    - **Infalsificabilidad**
    - **No repudio**
  - Puede implementarse con:
    - **Criptografía asimétrica** (más común)
    - **Criptografía simétrica** (menos recomendable)
- 

## 6. Criptografía

### a) Simétrica (clave única compartida)

- Ejemplo: AES
- Rápida pero problemáticas en distribución de clave.

### b) Asimétrica (clave pública/privada)

- Ejemplo: RSA



- Lo cifrado con una clave solo puede ser descifrado con la otra.
- Aplicable también para autenticación y firmas.

#### Modelo de uso:

- Juan cifra con **clave pública de Ana**, y ella descifra con su **clave privada**.
- Juan firma con su **clave privada**, y Ana verifica con la **clave pública de Juan**.

---

## 7. Kerberos (protocolo de autenticación)

- Usa **criptografía simétrica**.
- Evita el envío de contraseñas por la red.
- Componentes:
  1. **AS** (Authentication Server)
  2. **TGS** (Ticket Granting Server)
  3. **SS** (Service Server)
- Flujo:
  1. Cliente se autentica con el AS.
  2. Recibe un **TGT** (ticket de concesión).
  3. Solicita al TGS acceso a un servicio.
  4. Recibe **ticket de servicio** que luego presenta al servidor.

#### Características:



- Usa **claves de sesión temporales**.
- Evita el uso repetido de credenciales.
- Usa **principals** (identidades Kerberos con formato similar a DNS).



## Conclusiones clave

- La seguridad es **esencial en sistemas distribuidos**, donde los riesgos aumentan por la distribución de componentes.
- La protección debe aplicarse a **nivel de comunicación, datos e identidad**.
- Protocolos como Kerberos son fundamentales para autenticación robusta.
- Criptografía, firma digital y control de acceso deben aplicarse de forma coordinada.



## TEMA 14 – SISTEMAS P2P (Peer-to-Peer)



### 1. Introducción a P2P

- **Modelo tradicional cliente/servidor:**
  - Cliente solicita, servidor responde.
  - Costos y procesamiento concentrados en el servidor.
- **Modelo P2P:**
  - Todos los nodos pueden ser clientes y servidores a la vez ("sirvientes").
  - Intercambio **directo** entre nodos.
  - Devuelve poder y control a los usuarios.



## 2. Redes Superpuestas

- Red lógica que se construye sobre la red física.
- Puede ser:
  - **Estructurada**
  - **No estructurada**
- Los datos pueden no estar relacionados con la topología.

## 3. Objetivos de los Sistemas P2P

- Distribución de costos
- Aprovechamiento de recursos no utilizados
- Escalabilidad y confiabilidad
- Mayor autonomía
- Resistencia a censura, fallos y restricciones

## 4. Clasificación de P2P

### a) Según descentralización

Tipo	Descripción	Ejemplos
Híbrido descentralizado	Un servidor coordina	Napster



<b>Puro descentralizado</b>	Todos iguales, sin coordinación central	Gnutella, Freenet
<b>Parcialmente centralizado</b>	Algunos nodos son supernodos	Kazaa, nueva Gnutella

#### b) Según estructura

Tipo	Características	Ejemplos
No estructurado	Distribución aleatoria	Napster, KaZaA
Estructurado	Asignación precisa de datos	Chord, CAN, Tapestry, Pastry



## 5. Aplicaciones P2P

- Compartición de archivos (Napster, Gnutella, BitTorrent)
- Comunicación (VoIP, mensajería instantánea)
- Computación distribuida (SETI@home)
- Bases de datos (AmbientDB, XPeer)



## 6. Compartición de Archivos

#### Ventajas:

- Alta disponibilidad
- Escalabilidad
- Anonimato

#### Desventajas:

- Seguridad



- Uso intensivo del ancho de banda
- Búsqueda deficiente en algunos modelos



## 7. Ejemplos clave

- **Napster:** híbrido, usa servidor de directorio central.
- **Gnutella:** descentralizado puro, nodos iguales.
- **BitTorrent:** descarga por bloques, política “ojo por ojo”.
- **SETI@home:** usa recursos ociosos de millones de PC para buscar señales del espacio.



## 8. Búsqueda en P2P – Taxonomía

Método	Descripción
Búsqueda ciega	Inundaciones (BFS), caminatas aleatorias
Índice local	Cada nodo mantiene índices locales o filtros bloom
Profundización iterativa	Se incrementa progresivamente el alcance de búsqueda



## 9. DHT – Distributed Hash Tables

¿Qué es?

Una **tabla hash distribuida**, donde claves y valores están distribuidos entre nodos.

**Operaciones:**

- `put(clave, valor)`





- `get(clave) → valor`

#### **Ventajas:**

- Búsqueda eficiente
- Buena escalabilidad
- Alta disponibilidad

#### **Ejemplo de funcionamiento:**

1. Hashear clave
2. Determinar nodo responsable
3. Enviar petición `PUT` o `GET` al nodo

---

## 10. Seguridad en P2P

#### **Amenazas comunes:**

- Ataques de enrutamiento (redireccionamiento malicioso)
- Comportamiento inconsistente
- Suplantación de identidad
- Ataques DoS
- Nodos que entran/salen frecuentemente

#### **Soluciones:**



- Gestión de confianza
- Reputación distribuida
- Protocolos robustos como Kademlia

Aquí tienes un **resumen detallado para estudiar el tema de Sistemas Multiagente (SMA)** según el contenido del PDF que compartiste:



## SISTEMAS MULTIAGENTE (SMA)



### 1. Propiedades clave de los SMA

- **Autonomía:**  
Cada agente toma decisiones sin intervención directa.
- **Distribución:**  
Agentes operan en diferentes entornos o nodos, de forma descentralizada.
- **Interacción:**  
Se comunican y coordinan entre ellos para lograr sus metas.
- **Reactividad y Proactividad:**
  - *Reactividad:* responden a estímulos del entorno.
  - *Proactividad:* actúan para alcanzar objetivos definidos.
- **Flexibilidad y Adaptabilidad:**  
Pueden modificar su comportamiento según las condiciones cambiantes.



### 2. Tipos de SMA



Tipo	Descripción
cooperativos	Agentes colaboran hacia un <b>objetivo común</b> .
competitivos	Cada agente tiene <b>objetivos propios</b> , potencialmente en conflicto.
híbridos	Combinan cooperación y competencia entre agentes.

### 3. Niveles de Organización en SMA

Nivel	Descripción
Individual	Comportamiento interno y reglas de cada agente.
Social	Relaciones y dinámicas como cooperación o negociación.
Organizacional	Estructuras con roles, normas y jerarquías.
Ambiental	Entorno físico o digital donde actúan los agentes.

**Ejemplo:** Drones logísticos operan individualmente, cooperan entre ellos, siguen normas y se adaptan al clima.

### 4. Comunicación entre Agentes

- **ACL (Agent Communication Language):**  
Lenguaje estándar definido por **FIPA**, basado en actos de habla como **informar**, **solicitar**, etc.
- **Conversaciones:**  
Diálogos estructurados mediante protocolos.
- **Protocolos comunes:**
  - **Contract Net Protocol**



- Subasta (Auction-based)
- Negociación bilateral



## 5. Red de Contratos y Ontología



### Red de Contratos

- Modelo para **delegación de tareas**:
  1. Un *manager* publica un contrato.
  2. Agentes ofrecen propuestas.
  3. Se elige al mejor postor.



### Ontología

- Define un **vocabulario compartido** entre agentes.
- Especifica **conceptos y relaciones**.
- Asegura **interoperabilidad semántica**.

Ejemplo: Robots de diferentes marcas colaboran usando la misma ontología logística.



## 6. Plataformas para SMA

Plataforma	Características
JADE	Compatible con FIPA. Ideal para simulaciones.
Jason	Basado en AgentSpeak. Orientado a agentes <b>BDI</b> .



---

<b>GAMA</b>	Especializada en <b>simulaciones geoespaciales</b> .
<b>SPADE</b>	Agentes sobre protocolo <b>XMPP</b> .

---

## 7. Criterios para elegir plataforma SMA

- Compatibilidad con **estándares** (ej. FIPA).
- Soporte para:
  - Comunicación
  - Movilidad
  - Percepción del entorno
- Facilidad de **integración con sistemas distribuidos**.