



Confiabilidad

Sistemas Distribuidos
Dra. Cristina Abad



Confiabilidad (Reliability)

Capacidad de un sistema o componente de realizar sus funciones bajo condiciones dadas, por un periodo específico.



Confiabilidad de Sistemas Distribuidos

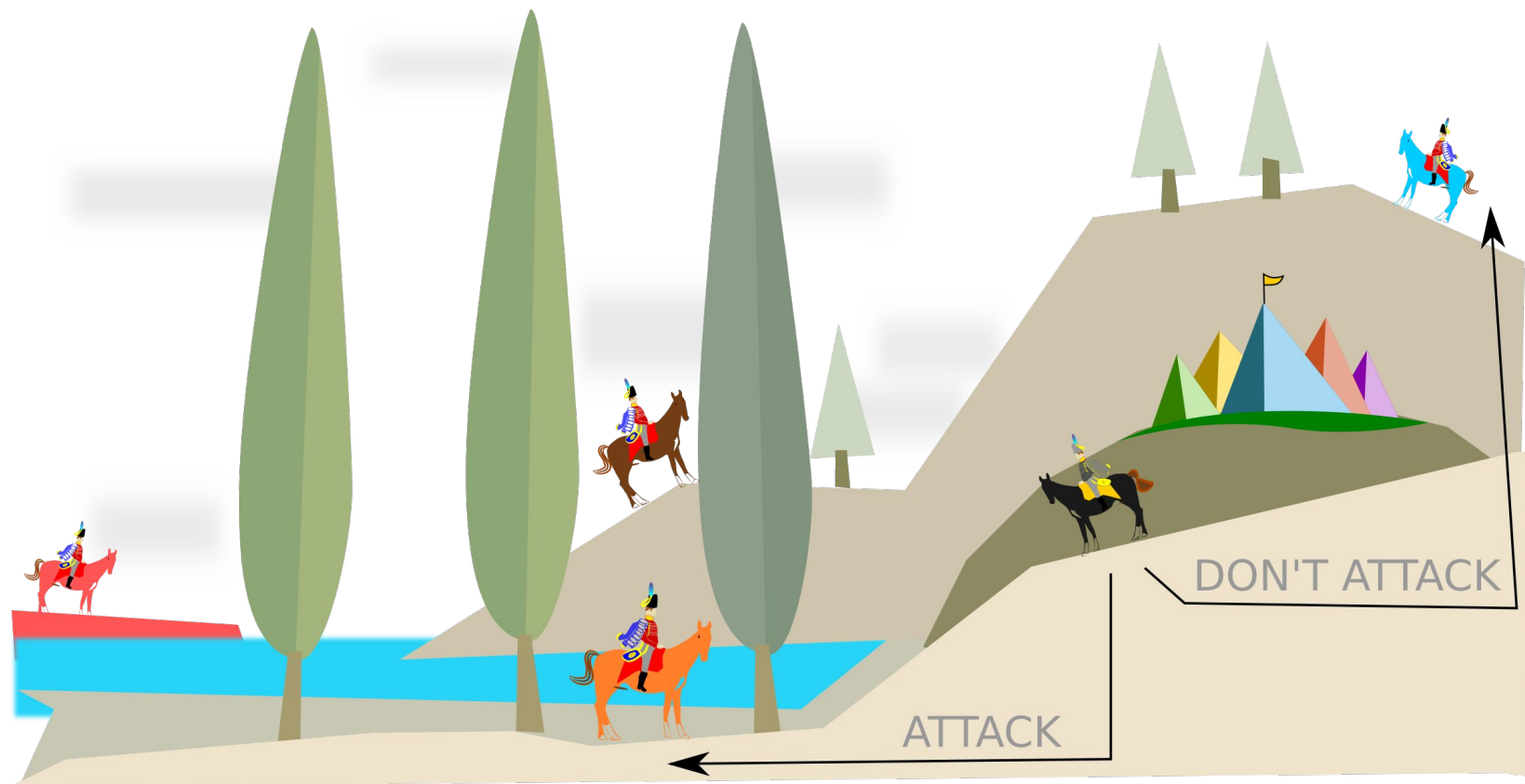
- Hacer que un SD sea confiable es muy importante
- Fallas en un SD pueden llevar de errores fácilmente reparables a problemas catastróficos
- SD confiable es diseñado para ser **tolerante a fallos**
 - Tolerancia a fallos: Hacer que un sistema funcione en presencia de fallas
 - Fallas pueden ocurrir en cualquier componente del sistema

Modelo de fallas (modelo fundamental)

- Fallas arbitrarias
- Fallas de omisión
- Fallas de tiempo



Fallas Arbitrarias o Bizantinas

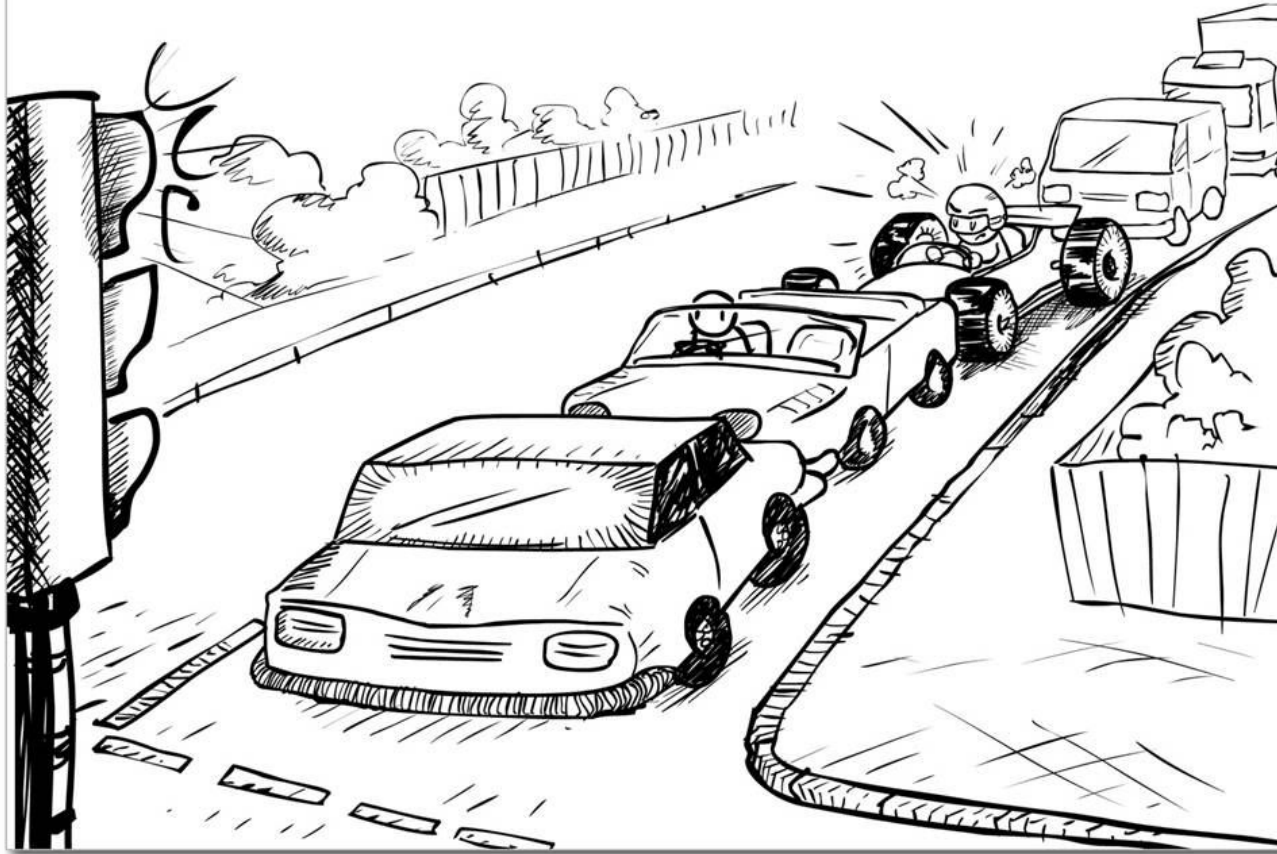


Fallas arbitrarias o de omisión

<i>Class of failure</i>	<i>Affects</i>	<i>Description</i>
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a <i>send</i> , but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

Fallas de tiempo

<i>Class of Failure</i>	<i>Affects</i>	<i>Description</i>
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.



Bob soon discovered that external factors affected his performance

¿Cómo construir sistemas distribuidos confiables?

- Solución: Enmascaramiento de fallas
- Mecanismos
 - Redundancia
 - Redundancia de información
 - Redundancia de tiempo
 - Redundancia física
 - Consenso y acuerdo

Enmascaramiento de fallas

- Para obtener sistemas confiables usando partes no confiables
- Maneras de enmascarar fallas
 - Escondiendo la falla
 - Ej.: Rollback en bases de datos
 - Convirtiendo la falla en otro tipo más manejable
 - Ej.: checksums en transmisión de paquetes, convierten una falla bizantina en falla de omisión o de tiempo (dependiendo de si hay retransmisiones)
- Elemento clave: **redundancia**

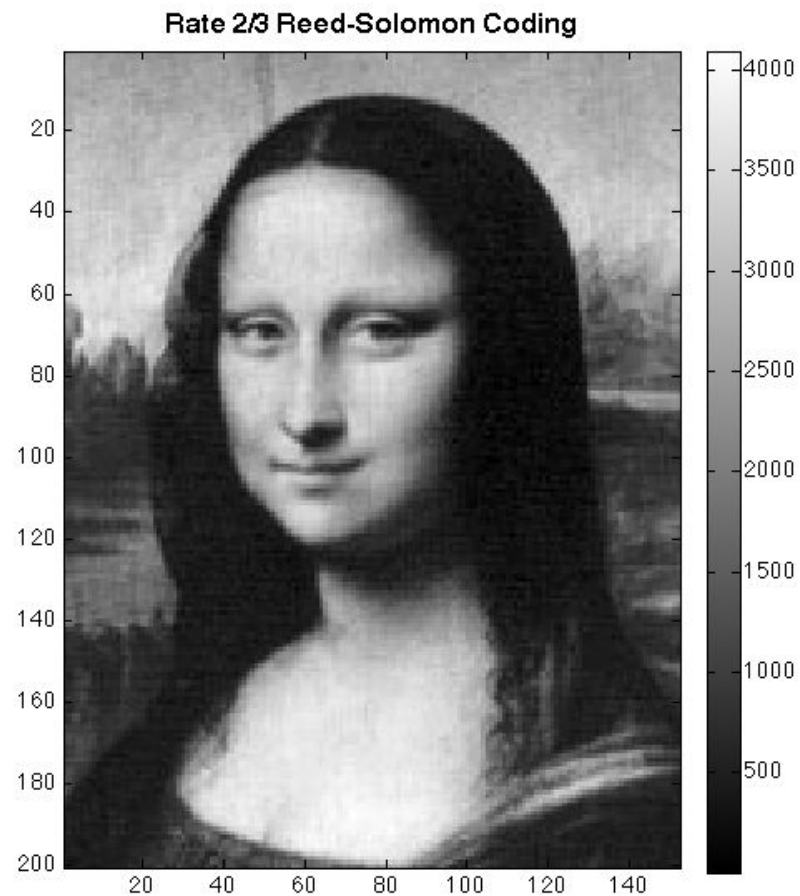
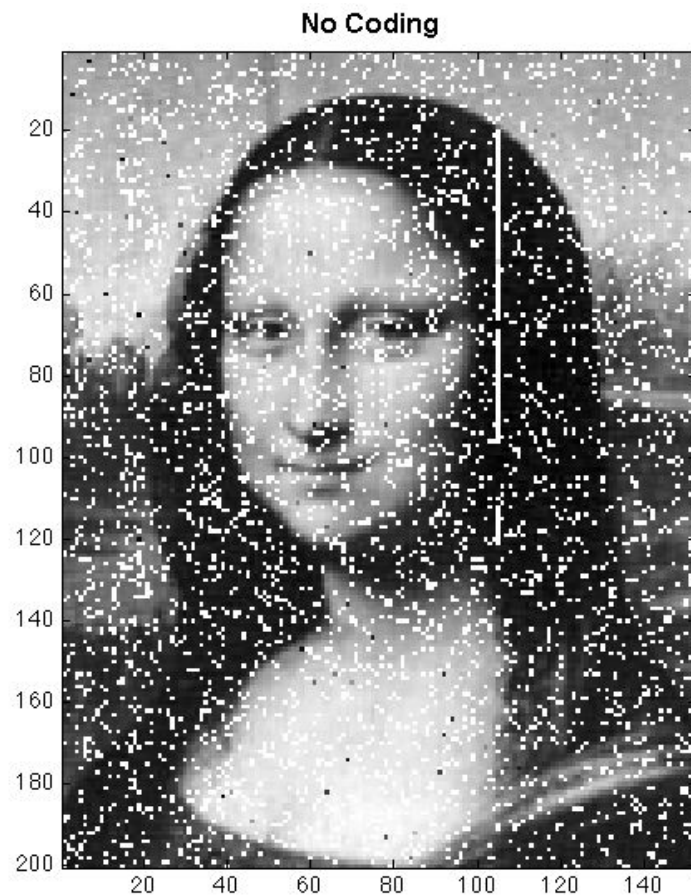
Redundancia de información

- Añadir bits para permitir **recuperarse** de bits dañados
 - Forward error correction (FEC)
- ¿Diferencia con códigos de paridad, CRCs, hashes o checksums?
 - Los códigos de paridad permiten **detectar** errores, no **corregirlos**
- ¿Ejemplos?

Ejemplos de Forward Error Correction (FEC)

- Hamming Codes
 - NAND flash memory
 - RAID 2
 - Algunos tipos de RAM
- Reed-Solomon codes (son un tipo de **Erase Codes**)
 - CDs
 - DVD
 - RAID 6
 - Códigos QR
- Usos: Streaming (Netflix), Almacenamiento (discos duros, Dropbox), Códigos de barras, ...

Ejemplo



Sesión 6

Redundancia de tiempo

- Una acción se realiza y, de ser necesario, se realiza nuevamente
- Solución más utilizada en fallas de intermitencia y transitorias
- Ejemplos:
 - En BD: Una transacción abortada y luego re-ejecutada
 - En redes: Retransmisión de paquetes perdidos (TCP)
- ¿Problemas de repetir acciones?

Terminología: Operaciones idempotentes

- No todas las operaciones pueden repetirse sin problemas
- Ok para operaciones **idempotentes**
- Idempotencia → operación puede repetirse múltiples veces y el resultado no cambia
 - Tampoco cambia el estado del sistema

IDEMPOTENT

operations that can be applied multiple times without changing the result beyond the initial application



Ejercicio (tomado del libro)

5.7 Discuss whether the following operations are *idempotent*:

- i) pressing a lift (elevator) request button;
- ii) writing data to a file;
- iii) appending data to a file.

Is it a necessary condition for idempotence that the operation should not be associated with any state?

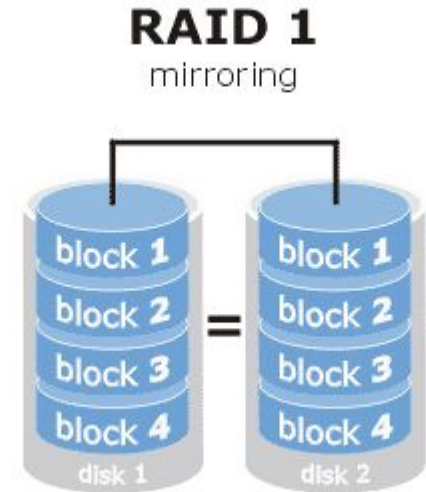
page 190

Terminología: Atomicidad

- “Todo o nada”
- Una operación atómica no puede ser interrumpida
- Da apariencia de ocurrir instantáneamente
- Atomicidad da garantía de aislamiento frente a procesos concurrentes
- Problema con operaciones NO atómicas → sistema puede quedar en estado inconsistente si proceso falla en medio de la operación
 - ¿Soluciones?

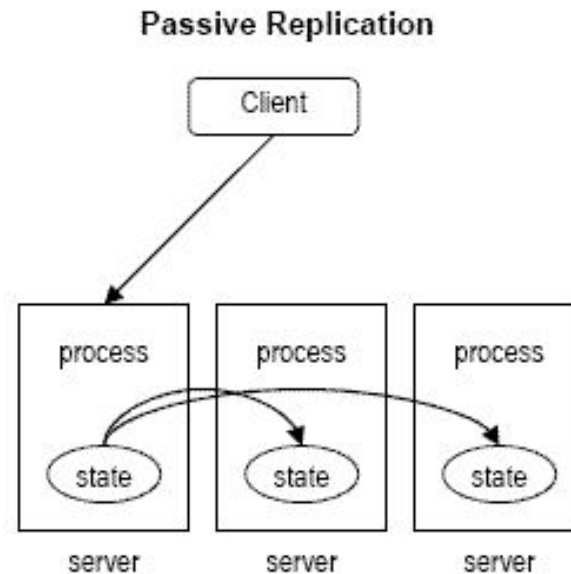
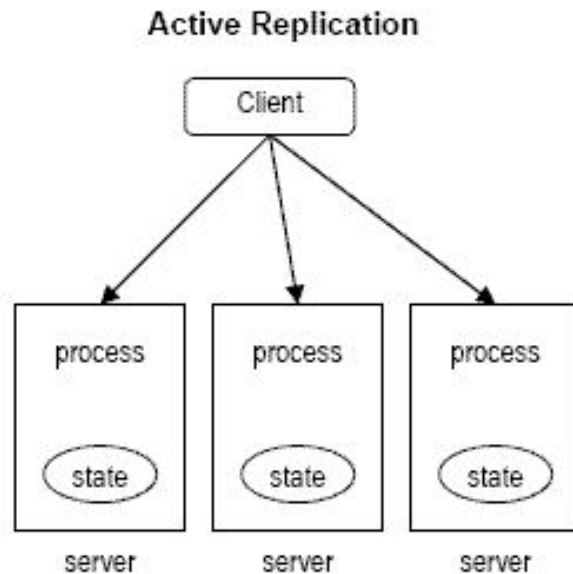
Redundancia física

- Añadir más componentes a un sistema por si uno falla
- Ejemplos:
 - Multihoming
 - Clusters con múltiples fuentes de poder
 - Múltiples tarjetas de red por servidor
 - Réplicas de bases de datos
 - Servidores múltiples (réplica de rol, SW)
 - RAID 1 (mirrored blocks)



Redundancia física: Replicación

- Replicación activa
- Replicación pasiva



Replicación activa

- Lamport la llamó **state machine replication**
- Puede requerir:
 - Operaciones determinísticas, idempotentes
 - Mecanismos de coordinación y acuerdo
- Compleja
 - Requiere más cuidado al implementar que la pasiva
- Mejor rendimiento que la pasiva
- ¿Cuánta réplicas (activas hacen falta)?
 - Depende del nivel de tolerancia a fallos deseado

k-fault tolerance

- Un sistema es **k-fault tolerant** si k nodos producen mismos resultados que sistema completo (de n nodos)
- Para que un sistema k-fault tolerant:
 - Si las fallas son **fail-stop** (o **fail-silent**) $\rightarrow k + 1$ nodos
 - Si las fallas son Byzantinas $\rightarrow 2k + 1$ nodos

Demostración en: Reaching Agreement in the Presence of Faults (Pease, Shostak y Lamport).
Journal of ACM. 1980. Ganador del 2005 Edsger W. Dijkstra Prize in Distributed Computing.
Disponible en: <http://research.microsoft.com/en-us/um/people/lamport/pubs/reaching.pdf>

- Lectura recomendada: The writings of Leslie Lamport
 - <http://research.microsoft.com/en-us/um/people/lamport/pubs/pubs.html>

Replicación pasiva

- Un servidor (**primario**) procesa los pedidos de clientes
- **Primario** actualiza estado en los otros servidores (**backups**)
- Si primario falla, uno de los servidores backup toma su lugar
- Puede ser usado en procesos determinísticos y no determinísticos
- Otra terminología:
 - Backup → pasivo
 - Primario → activo

Replicación pasiva (cont.)

- Ventaja:
 - Simplicidad
- Desventajas:
 - En caso de falla de primario, las respuestas demoran en llegar (hasta que backup asume rol)
 - No maneja muy bien fallas Byzantinas: no hay cómo verificar que primario esté funcionando correctamente
- Mínimo # de servidores: 2 (1 primario y 1 backup)

Failover: ¿Cuándo debe backup asumir rol primario?

1. Backup puede enviar mensajes a primario preguntando si todavía está funcionando
 - a. En sistemas asíncronos, puede que servidor esté lento
2. Un mecanismo de HW puede usarse para rebootear a primario
3. Usar un disco con dos puertos que se comparta entre primario y backup; si primario falla, backup puede leer status actual del disco compartido
4. Puede delegarse decisión a un humano

¿Qué pasa con operaciones incompletas cuando falta primario?

Tipos de failover

- Manual-frío (o, sin Alta Disponibilidad / no High Availability)
 - Operador debe manualmente apagar y reiniciar servicios
- Automático-frío
 - Guarda estado en una bitácora (journal) en un dispositivo de almacenamiento secundario; puede tomar minutos (o hasta más de una hora) re-iniciar el servicio
- Manual-caliente
 - El sistema se sincroniza con nodos backups; operador envía comando de failover manualmente a un backup cuando el primario falla
- Automático-caliente ← La verdadera Alta Disponibilidad
 - Proporciona failover rápido y automático al hacer que un backup caliente asuma rol primario
- Alta disponibilidad tibia (Warm HA)
 - Backup mantiene una copia sincronizada del estado del primario
 - Puede que software o parte del estado del servicio deban re-iniciarse o re-configurarse cuando falla el primario

Consenso y acuerdo

← Lo veremos más adelante

Byzantine Agreement (single source has an initial value)

Agreement: All non-faulty processes must agree on the same value.

Validity: If the source process is non-faulty, then the agreed upon value by all the non-faulty processes must be the same as the initial value of the source.

Termination: Each non-faulty process must eventually decide on a value.

Consensus Problem (all processes have an initial value)

Agreement: All non-faulty processes must agree on the same (single) value.

Validity: If all the non-faulty processes have the same initial value, then the agreed upon value by all the non-faulty processes must be that same value.

Termination: Each non-faulty process must eventually decide on a value.

Acuerdo y Consenso son problemas equivalentes
(pueden convertirse de uno al otro).



Ejercicio (tomado del libro)

18.1 Three computers together provide a replicated service. The manufacturers claim that each computer has a mean time between failure of five days; a failure typically takes four hours to fix. What is the availability of the replicated service? *page 766*

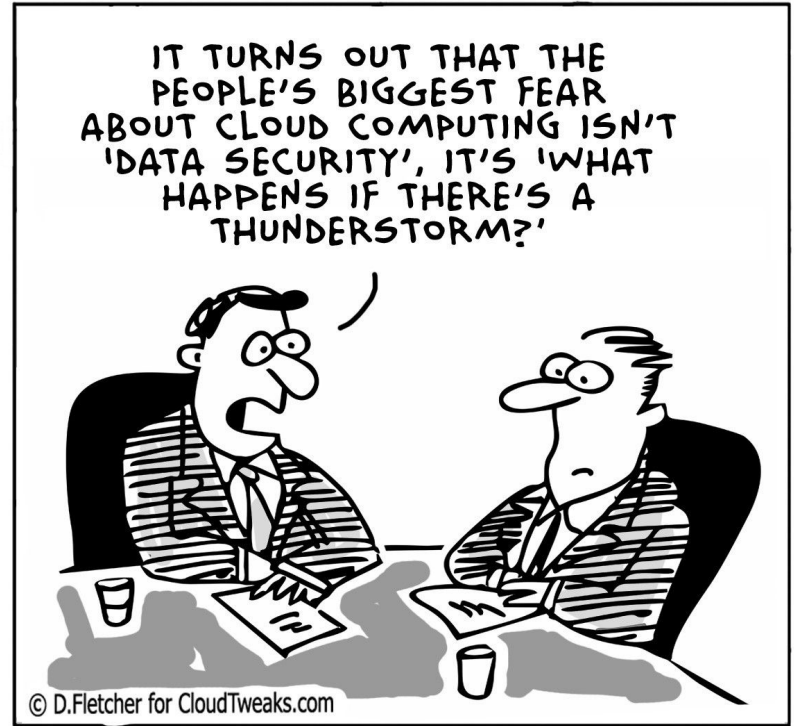
Pág 766: Si c/u de n servidores tiene una probabilidad p de fallar o estar no disponible, entonces la **disponibilidad (availability)** de un objeto que está almacenado (replicado) en cada uno de esos servidores es:

$$1 - \text{probabilidad}(\text{todos los servidores hayan fallado o no estén disponibles}) = \\ = 1 - p^n \quad \leftarrow \text{asume independencia de fallo de los componentes}$$

Modelo de seguridades (modelo fundamental)

■ Se debe proteger:

- Procesos
- Canales
- Objetos/Servicios/Microservicios
- Datos



Modelo de seguridades

■ Procesos

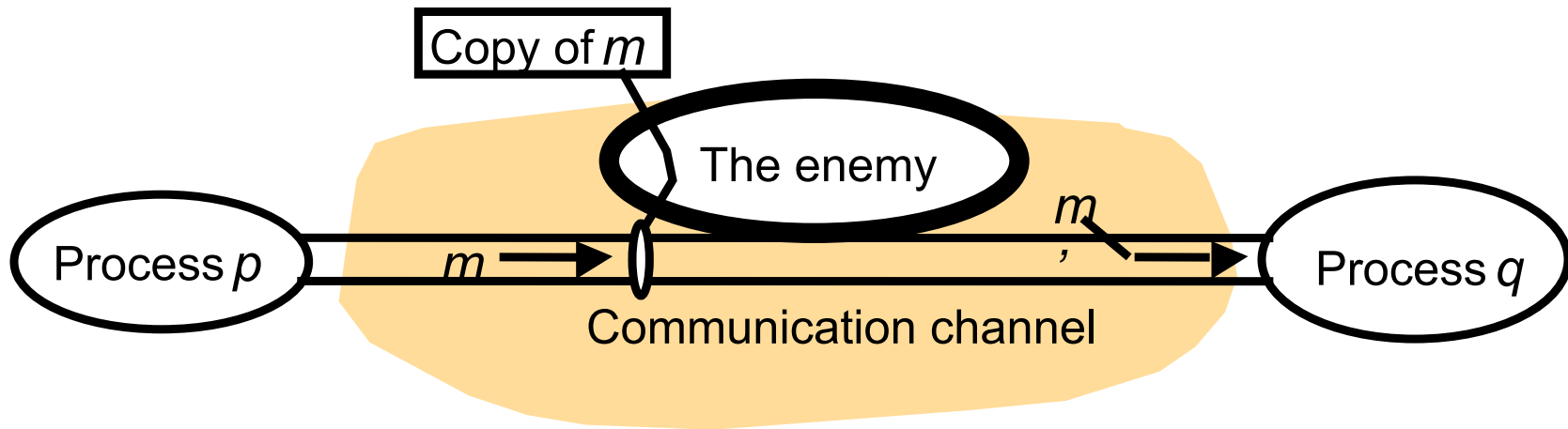
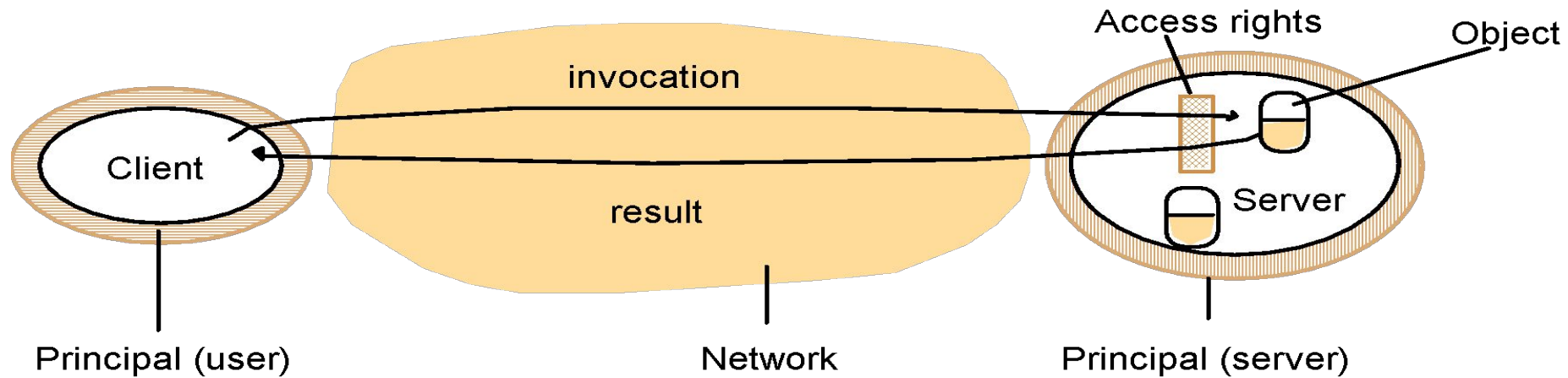
- Memoria: control de acceso
- Autenticación: firmas digitales

■ Canales → canales seguros

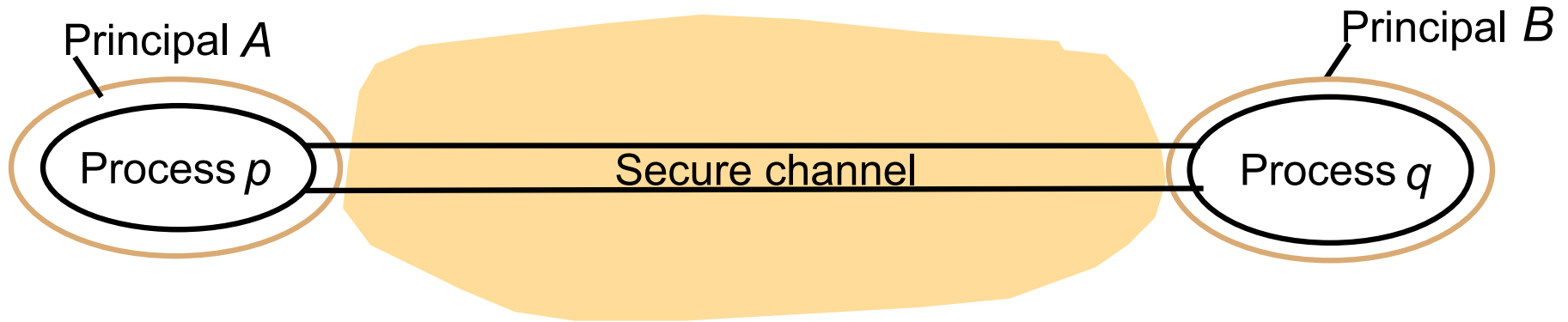
- Proteger los mensajes: encriptación
- Integridad: checksums y firmas digitales
- Disponibilidad: Ej.: DoS

■ Objetos/Datos

- Derechos de acceso

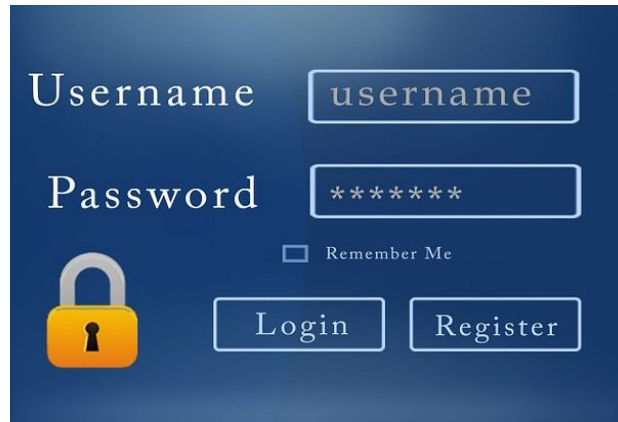


Canales seguros



Discusión

- Se sabe que almacenar contraseñas de usuarios en texto plano o encriptadas en una base de datos o archivo de texto es considerado una mala práctica
 - ¿Por qué?
 - BONUS: ¿Sabe qué es lo recomendado?



Username

Password

☐ Remember Me



Ejemplo: Caso de LinkedIn

What Happened?

On May 17, 2016, we became aware that data stolen from LinkedIn in 2012 was being made available online. This was not a new security breach or hack. We took immediate steps to invalidate the passwords of all LinkedIn accounts that we believed might be at risk. These were accounts created prior to the 2012 breach that had not reset their passwords since that breach.

What Information Was Involved?

Member email addresses, hashed passwords, and LinkedIn member IDs (an internal identifier LinkedIn assigns to each member profile) from 2012.

What We Are Doing

We invalidated passwords of all LinkedIn accounts created prior to the 2012 breach that had not reset their passwords since that breach. In addition, we are using automated tools to attempt to identify and block any suspicious activity that might occur on LinkedIn accounts. We are also actively engaging with law enforcement authorities.

LinkedIn has taken significant steps to strengthen account security since 2012. For example, we now use salted hashes to store passwords and enable additional account security by offering our members the option to use two-step verification.

What You Can Do

Ejemplos

Wireless Sensor Networks

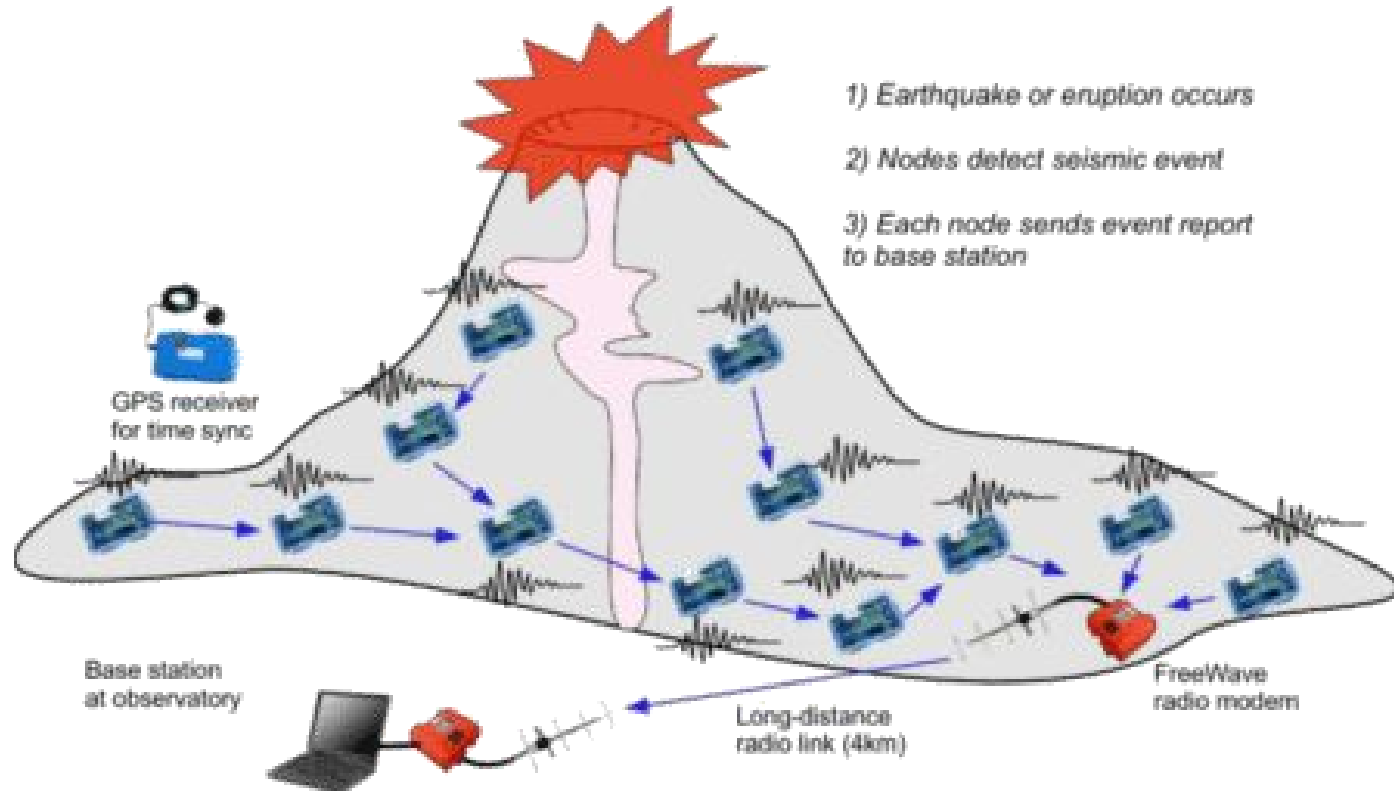
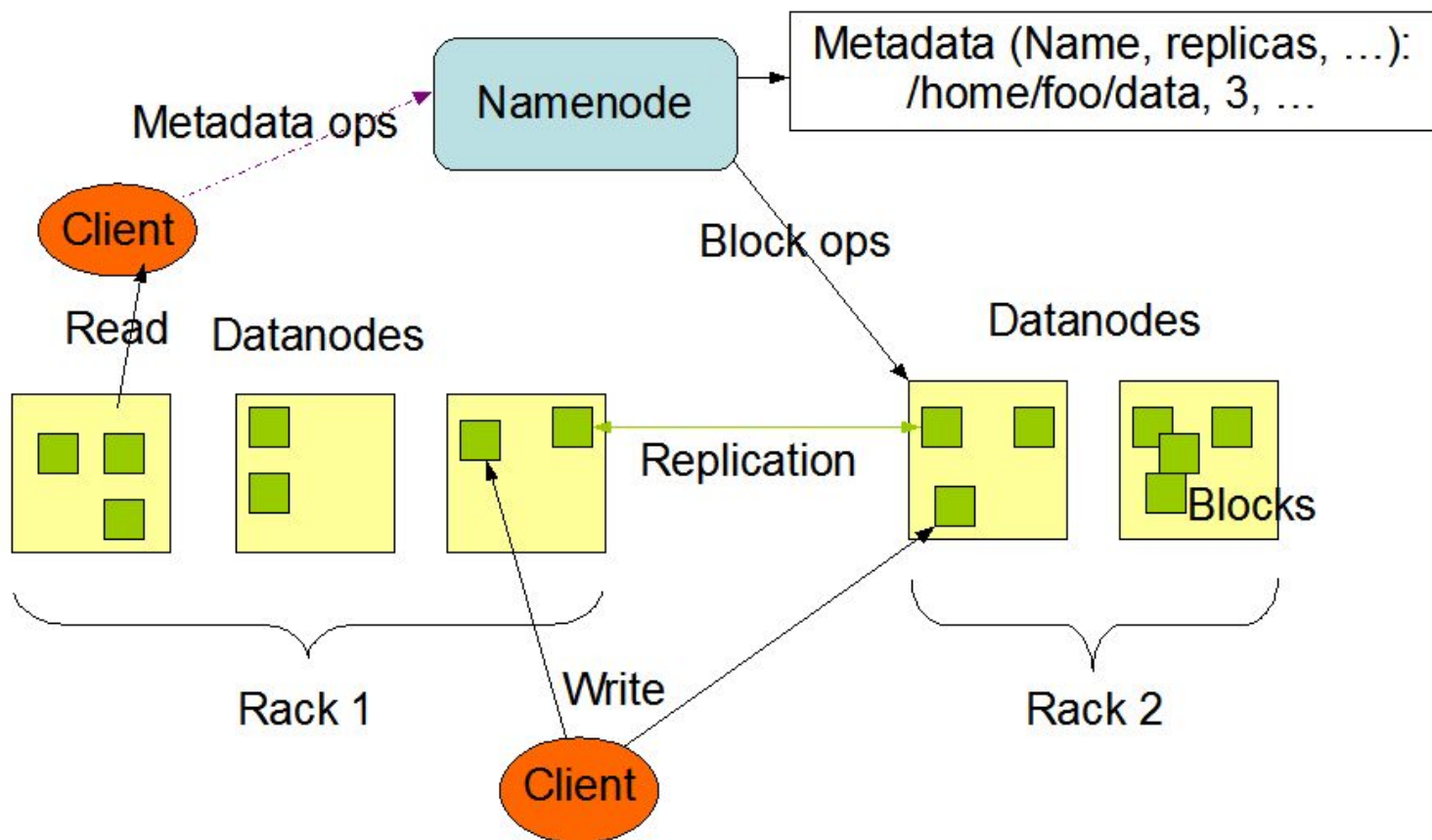


Figure 1: An application of wireless sensor network

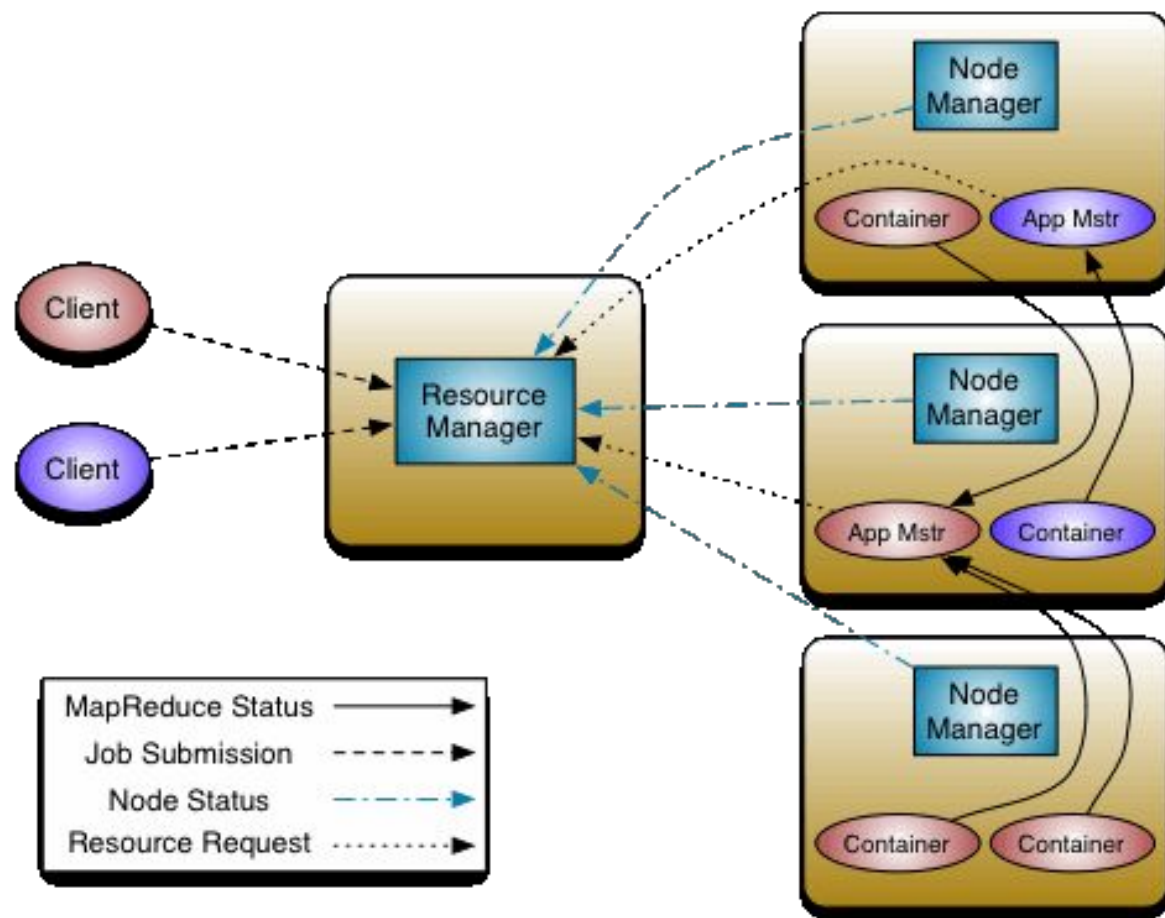
Hadoop: HDFS



¿Cómo mejorar disponibilidad del NameNode?

- Replicación pasiva → Soporta varios tipos de failover
 - Manual-frío → Estado almacenado en NFS compartido; operador se encarga de failover
 - Automático-frío → Con SW para mirroring de disco duro; failover con HA SW de Linux
 - Manual-caliente → AvatarNode (Facebook)
 - Automático-caliente → muchas alternativas usadas por varias empresas
 - Ej1: Usar SW de HA de Linux con StandbyNode y NodeReplicator
 - Ej2: convertir BackupNode en Hot Standby
 - Alta disponibilidad tibia (Warm HA)
 - BackupNode; Puede tomar de 20 minutos a 1 hora
 - BackupNode mantiene namespace totalmente sincronizado con NameNode activo
 - BackupNode re-descubre ubicaciones de bloques via “block reports” (DataNodes)
- Replicación activa → ConsensusNode:
 - <http://www.slideshare.net/KonstantinVShvachko/hs-2014shvachkocoordinating-metadata>

Hadoop: YARN



Más información

- Monitoreo de volcanes:
 - <http://www.eecs.harvard.edu/~mdw/proj/volcano/reventador-usenix.pdf>
- Hadoop:
 - HDFS: <http://www.slideshare.net/KonstantinVShvachko/hdfs-design-principles>
 - Replicación activa y pasiva para el HDFS namenode:
<http://www.slideshare.net/KonstantinVShvachko/hs-2014shvachkocoordinating-metadata>
 - YARN:
[https://www.sics.se/~amir/files/download/dic/2013%20-%20Apache%20Hadoop%20YARN:%20Yet%20Another%20Resource%20Negotiator%20\(SoCC\).pdf](https://www.sics.se/~amir/files/download/dic/2013%20-%20Apache%20Hadoop%20YARN:%20Yet%20Another%20Resource%20Negotiator%20(SoCC).pdf)
- Otros ejemplos:
 - <https://www.cs.rutgers.edu/~pxk/417/notes/content/16-dfs-slides.pdf>

Otras lecturas recomendadas

- Slides que usé para esta presentación:
Reliability of Distributed Systems. Erick Redwine, JoAnne Holliday.
Disponible: <http://www.cse.scu.edu/~jholliday/REL-EAR.htm>
- An argument for increasing TCP's initial congestion window:
<http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/36640.pdf>
- How speedy is SPDY?
<https://www.usenix.org/conference/nsdi14/technical-sessions/wang> ←
también hay un vídeo de la presentación en ese enlace



¿Preguntas?

