IIC2523
Sistemas Distribuidos

Hernán F. Valdivieso López
(2025 - 2 / Clase 23)

Redes *Peer to Peer* (P2P)
¿Qué hacer cuando todos son iguales?

Temas de la clase

1. Introducción a las *Peer to Peer*
    a. Definiciones
    b. Desafíos
    c. Búsqueda de información
    d. Ataques y Seguridad
2. *Peer to Peer* Estructurado
    a. *Distributed Hash Table* (DHT)
    b. Seguridad en DHT
    c. Caso aplicado - *Chord*

Introducción a las *Peer to Peer* (P2P)

Definiciones

Desafíos

Búsqueda de información

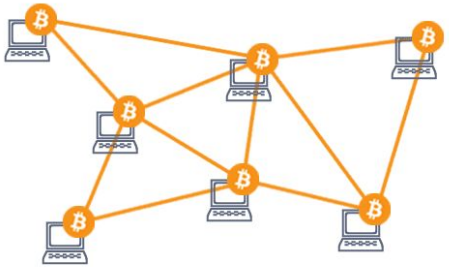
Ataques y Seguridad

Peer to Peer (P2P) - Definición

Arquitectura de red distribuida donde cada nodo actúa tanto como cliente como servidor, compartiendo recursos directamente con otros nodos sin necesidad de un servidor centralizado.

Peer to Peer (P2P) - Características Clásicas

- No hay organización global y cada usuario contribuye con recursos para alivianar la carga y no depender de solo un nodo.
- Mientras más crece la red, los nodos tienen conexión con un subconjunto de otros nodos.
- Vecinos de la red definidos por proximidad o azar.
- Alta tolerancia ingreso y salida de clientes.
- Más simple en términos de costos.
- Ejemplos: BitTorrent o GTA Online.



Peer to Peer (P2P) - Desafíos

- **Complejidad de distribución:** Garantizar un equilibro de carga de trabajo y disponibilidad sin añadir sobrecargas indebidas.
- **Volatilidad de los recursos**: Los usuarios y dueños de los datos no garantizan que estén encendidas o conectadas continuamente.
- **Consistencia de datos mutables**: Ante objetos con valores cambiantes, se requiere servidores de confianza adicionales para gestionar versiones de tales archivos.

Peer to Peer (P2P) - Desafíos

- **Complejidad de distribución:** Garantizar un equilibro de carga de trabajo y disponibilidad sin añadir sobrecargas indebidas.
- **Volatilidad de los recursos**: Los usuarios y dueños de los datos no garantizan que estén encendidas o conectadas continuamente.
- **Consistencia de datos mutables**: Ante objetos con valores cambiantes, se requiere servidores de confianza adicionales para gestionar versiones de tales archivos.
- **Búsqueda de información:** en ausencia de un servidor central, no existe una entidad que indique qué nodo tiene la información deseada... hay que **buscar**.

Peer to Peer (P2P) - Búsqueda *Flooding*

Gritar con desesperación... y hacer que todos griten también.

- *Breadth-First Search* (BFS)
- Reenvío masivo a todos los vecinos y cada vecino vuelve a reenviar la búsqueda a todos.
- Se define ***Time-To-Live*** **(TTL)** que es el tiempo que una consulta puede estar existiendo antes de ser declarada como fallida.
- Explosión exponencial de mensajes implica un alto consumo de banda ancha.

Peer to Peer (P2P) - Búsqueda *Diffusion*

Susurrar el mensaje... con la esperanza de que se propague.

- *Breadth-First Search* (BFS) con aleatoriedad.
- Reenvío masivo a todos los vecinos con un *delay* aleatorio entre cada uno y cada vecino vuelve a reenviar la consulta con un *delay*.
- Más escalable que *Flooding* porque no congestiona la red en un tiempo tan acotado, pero la respuesta puede demorar más en llegar.
- Algunas consultas llegan más lejos bajo el mismo TTL.

Peer to Peer (P2P) - Búsqueda *Random Walks*

Buscar a ciegas... y rezar que alguien escuche.

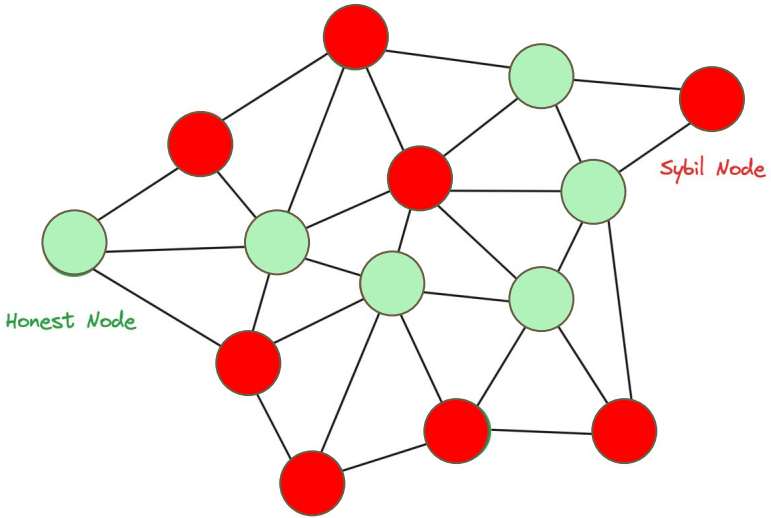
- *Depth-First Search* (DFS) con aleatoriedad.
- Consultar a un vecino aleatoriamente y este propaga a otro vecino aleatoriamente.
- Alta latencia en la respuesta e incluso está la probabilidad de no encontrar el dato incluso si estaba cerca.
- Más escalable por su bajo consumo de red simultáneo.

Peer to Peer (P2P) - Seguridad

- Ante una red sin servidor, el atacante tiene ciertas dificultades en planificar su ataque... pero inventa nuevas formas 🔪

Ataque Sybil

Multiplicando identidades



Peer to Peer (P2P) - Seguridad: Ataque Sybil

Contramedidas - Ninguna es perfecta por si sola, pero combinadas son muy robustas.

Pruebas de trabajo (*Proof of Work*)

Hacer que un nodo tenga que realizar un cálculo costoso para obtener su identificador que permite entrar a la red.

Peer to Peer (P2P) - Seguridad: Ataque Sybil

Contramedidas - Ninguna es perfecta por si sola, pero combinadas son muy robustas.

Pruebas de trabajo (*Proof of Work*)

Hacer que un nodo tenga que realizar un cálculo costoso para obtener su identificador que permite entrar a la red.

Verificación de identidad

Usar claves públicas verificadas por una autoridad descentralizada (*web-of-trust* o *blockchain*) o por quorum.

Peer to Peer (P2P) - Seguridad: Ataque Sybil

Contramedidas - Ninguna es perfecta por si sola, pero combinadas son muy robustas.

Pruebas de trabajo (*Proof of Work*)

Hacer que un nodo tenga que realizar un cálculo costoso para obtener su identificador que permite entrar a la red.

Topologías reforzadas

Algunos diseños restringen cómo se forman conexiones, evitando que un nodo elija arbitrariamente a quién se conecta.

Verificación de identidad

Usar claves públicas verificadas por una autoridad descentralizada (*web-of-trust* o *blockchain*) o por quorum.

Peer to Peer (P2P) - Seguridad: Ataque Sybil

Contramedidas - Ninguna es perfecta por si sola, pero combinadas son muy robustas.

Pruebas de trabajo (*Proof of Work*)

Hacer que un nodo tenga que realizar un cálculo costoso para obtener su identificador que permite entrar a la red.

Verificación de identidad

Usar claves públicas verificadas por una autoridad descentralizada (*web-of-trust* o *blockchain*) o por quorum.

Topologías reforzadas

Algunos diseños restringen cómo se forman conexiones, evitando que un nodo elija arbitrariamente a quién se conecta.

Contención geográfica o de latencia

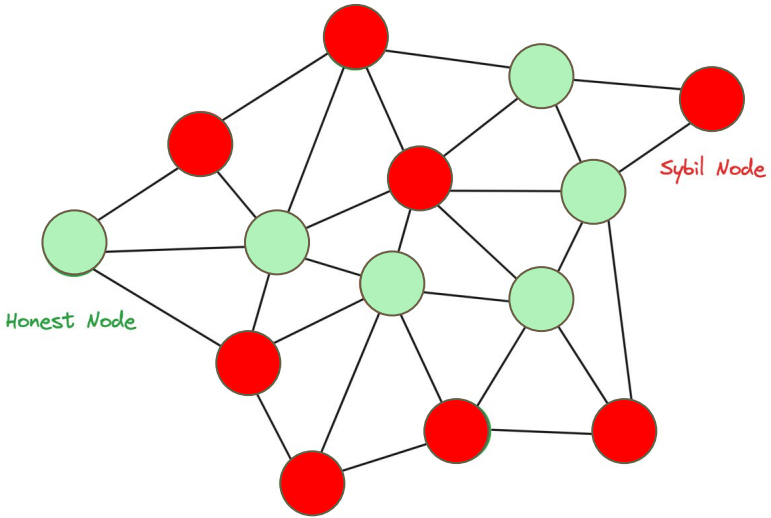
Obligar a que los nodos que interactúan tengan ciertas características (IP geográficamente consistentes o una latencia mínima)

Peer to Peer (P2P) - Seguridad

- Ante una red sin servidor, el atacante tiene ciertas dificultades en planificar su ataque... pero inventa nuevas formas 🔪

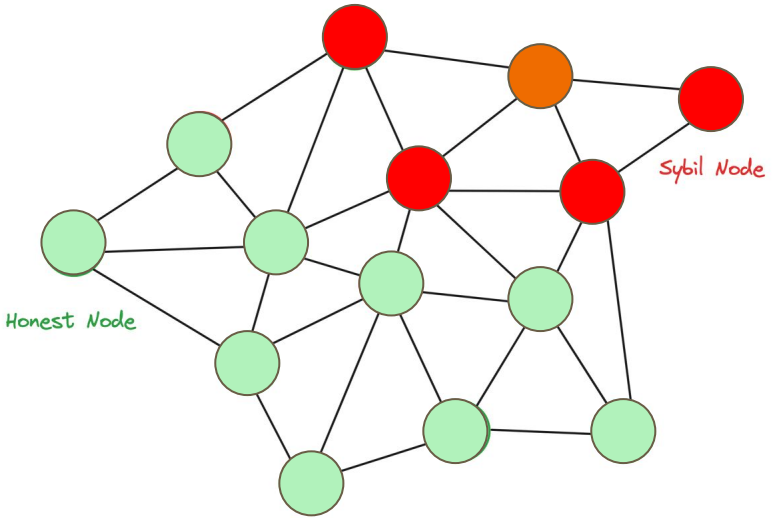
Ataque Sybil

Multiplicando identidades



Ataque Eclipse

Aislar para controlar



Peer to Peer (P2P) - Seguridad: Ataque Eclipse

Contramedidas - Ninguna es perfecta por si sola, pero combinadas son muy robustas.

Chequeo cruzado entre múltiples caminos

Consultar nodos posiblemente más lejanos en la conexión para verificar consistencia.

Peer to Peer (P2P) - Seguridad: Ataque Eclipse

Contramedidas - Ninguna es perfecta por si sola, pero combinadas son muy robustas.

Chequeo cruzado entre múltiples caminos

Consultar nodos posiblemente más lejanos en la conexión para verificar consistencia.

Uso de múltiples fuentes independientes

Definir algunos vecinos cuyos nodos sean geográficamente dispares.

Peer to Peer (P2P) - Seguridad: Ataque Eclipse

Contramedidas - Ninguna es perfecta por si sola, pero combinadas son muy robustas.

Chequeo cruzado entre múltiples caminos

Consultar nodos posiblemente más lejanos en la conexión para verificar consistencia.

Limitar el número de conexiones de entrada por IP/subred

Evita que un solo atacante con múltiples nodos falsos se infiltre completamente como vecino de otros.

Uso de múltiples fuentes independientes

Definir algunos vecinos cuyos nodos sean geográficamente dispares.

Peer to Peer (P2P) - Seguridad: Ataque Eclipse

Contramedidas - Ninguna es perfecta por si sola, pero combinadas son muy robustas.

Chequeo cruzado entre múltiples caminos

Consultar nodos posiblemente más lejanos en la conexión para verificar consistencia.

Uso de múltiples fuentes independientes

Definir algunos vecinos cuyos nodos sean geográficamente dispares.

Limitar el número de conexiones de entrada por IP/subred

Evita que un solo atacante con múltiples nodos falsos se infiltre completamente como vecino de otros.

Verificación temporal y rotación periódica de vecinos

Actualizar lista de vecinos con fuentes aleatorias y comprobar consistencia en la información entregada.

Peer to Peer (P2P) - Seguridad inherente

En una red P2P no estructurada, donde cada cliente también actúa como servidor y la información está distribuida, un atacante se enfrenta a varias dificultades:

- No sabe de antemano qué nodo posee la información que desea interceptar o atacar.
- Bloquear o comprometer un nodo no afecta al funcionamiento global del sistema.
- Cada nodo puede usar mecanismos de defensa independientes (cifrado, *firewall*, IDS/IPS...), lo que impide una estrategia de ataque uniforme.
- Si la información está fragmentada o replicada, atacar requiere coordinar acciones contra múltiples nodos, aumentando el costo y reduciendo la probabilidad de éxito.

Peer to Peer (P2P) - Seguridad inherente

En una red P2P no estructurada, donde cada cliente también actúa como servidor y la información está distribuida, un atacante se enfrenta a varias dificultades:

- No sabe de antemano qué nodo posee la información que desea interceptar o

Sin embargo... la libertad total en las conexiones y búsquedas tiene un costo:

- **La comunicación y la localización de información pueden ser lentas, ruidosas y poco eficientes.** 😓

- Si la información está fragmentada o replicada, atacar requiere coordinar acciones contra múltiples nodos, aumentando el costo y reduciendo la probabilidad de éxito.

Peer to Peer (P2P) - Seguridad inherente

En una red P2P no estructurada, donde cada cliente también actúa como servidor y la información está distribuida, un atacante se enfrenta a varias dificultades:

- No sabe de antemano qué nodo posee la información que desea interceptar o
- exito.

Sin embargo... la libertad total en las conexiones y búsquedas tiene un costo:

- **La comunicación y la localización de información pueden ser lentas, ruidosas y poco eficientes.** 😓

¿Existe una forma de organizar la red, sin servidores centrales, que mantenga la descentralización pero optimice la búsqueda de datos? 🤔

Peer to Peer Estructurado

***Distributed Hash Table* (DHT)**

Seguridad en DHT

Caso particular: Chord

Distributed Hash Table (DHT)

- Estructura de datos distribuida que permite almacenar y buscar eficientemente sin servidor central.
- Se utiliza una función de ***hash* consistente** para definir una *key* para cada dato y nodo de la red.
  - La key del nodo (o ID) suele ser el *hash* de una pieza de información de identificación sobre el nodo, como su dirección IP, correo, etc.
  - Se utiliza la *key* para determinar la posición del dato en la red y quién es el nodo responsable de dicho dato.

Distributed Hash Table (DHT) - *Hash* consistente

- Primero veamos *hash* "normal" (no consistente)
  - Asumiendo 5 nodos, se tendría que hacer `hash(X)%5` para determinar qué nodo guarda la información de X.
  - Si un nodo se sale, habría que hacer `hash(X)%4` y todos los datos cambiarían al nodo que es responsable de él.

Distributed Hash Table (DHT) - *Hash* consistente

- Primero veamos *hash* "normal" (no consistente)
  - Asumiendo 5 nodos, se tendría que hacer `hash(X)%5` para determinar qué nodo guarda la información de X.
  - Si un nodo se sale, habría que hacer `hash(X)%4` y todos los datos cambiarían al nodo que es responsable de él.
- Ahora, con *hash* consistente.
  - Este entrega un valor que ya aplica un %N internamente que no depende del tamaño de la red → El *hash* está condicionado a un rango predefinido.
  - El *hash_consistente(id_nodo)* dará la posición del nodo en el rango predefinido.
  - Si se va un nodo, a lo más el nodo "sucesor" y "predecesor" serán afectados.

Distributed Hash Table (DHT) - *Hash* consistente

Ejemplo

- Hash da valores entre 0 y 9.
- Tengo 3 nodos cuyo Hash es 0, 4 y 8 respectivamente.
  - Nodo 0 → Responsable de todos los datos cuya *key* es 9 y 0.
  - Nodo 4 → Responsable de todos los datos cuya *key* es 1, 2, 3 y 4.
  - Nodo 8 → Responsable de todos los datos cuya *key* es 5, 6, 7 y 8.

Distributed Hash Table (DHT) - *Hash* consistente

Ejemplo

- Hash da valores entre 0 y 9.
- Tengo 3 nodos cuyo Hash es 0, 4 y 8 respectivamente.
  - Nodo 0 → Responsable de todos los datos cuya *key* es 9 y 0.
  - Nodo 4 → Responsable de todos los datos cuya *key* es 1, 2, 3 y 4.
  - Nodo 8 → Responsable de todos los datos cuya *key* es 5, 6, 7 y 8.
- Se va el nodo 8.
  - Nodo 0 adjunta la información del nodo 8. Es decir, será responsable de 5, 6, 7, 8, 9 y 0.
  - Nodo 4 sigue sin cambios.

Distributed Hash Table (DHT) - Funcionamiento

- El *Hash* produce una salida numérica de longitud fija.
  - Por ejemplo, `SHA-256(uwu.mp4)` = 192371391873192381444491831911
- Asignación a un nodo cuyo identificador esté más próximo al *hash* (según alguna métrica, como XOR, prefijo, o distancia circular).

Distributed Hash Table (DHT) - Funcionamiento

- El *Hash* produce una salida numérica de longitud fija.
  - Por ejemplo, `SHA-256(uwu.mp4)` = 192371391873192381444491831911
- Asignación a un nodo cuyo identificador esté más próximo al *hash* (según alguna métrica, como XOR, prefijo, o distancia circular).
- Cada nodo tiene una tabla de *routing* que corresponde a un conjunto de punteros a otros nodos, cuidadosamente elegidos, para permitir que un nodo pueda encaminar solicitudes hacia el nodo responsable de una clave, en pocos saltos.
- **Si un atacante logra que todas las entradas de esa tabla apunten a nodos que él controla, ahí el nodo víctima queda atrapado.**

Distributed Hash Table (DHT) - Seguridad

1. Difícil secuestrar rutas sin alterar muchos nodos

- Cada nodo tiene una posición y vecinos determinados por su ID y el *hash*.
- Las rutas de búsqueda siguen ciertas regla determinadas por el algoritmo.

🔒 Implicancia 🔒

- Para un atacante, no basta con insertar uno o dos nodos maliciosos en la red; necesitaría estar posicionado en ubicaciones específicas a lo largo de múltiples rutas para tener una probabilidad razonable de interceptar búsquedas.
- Permite mitigar ataques Eclipse.

Distributed Hash Table (DHT) - Seguridad

2. Nodos no pueden elegir qué datos almacenar

- Cada dato se almacena en los nodos cercanos al *hash* de su clave.
- Los nodos responsables no eligen los datos; es una consecuencia del algoritmo de ubicación.

🔒 Implicancia 🔒

- Un nodo malicioso no puede simplemente decidir almacenar claves críticas o espiar datos sensibles a no ser que logre conseguir un ID específico y para eso debe "vencer" al *hash*.

Distributed Hash Table (DHT) - Seguridad

3. La red se adapta a nodos que entran/salen

- Están diseñadas para ser dinámicas: si un nodo desaparece, los datos se replican o redistribuye automáticamente.
- Las tablas con la posición de los nodos se actualizan con el tiempo.

🔒 Implicancia 🔒

- La red es resiliente a ataques de denegación parcial, donde se intenta eliminar nodos clave.
- También permite expulsar nodos erráticos o sospechosos mediante rotación natural.

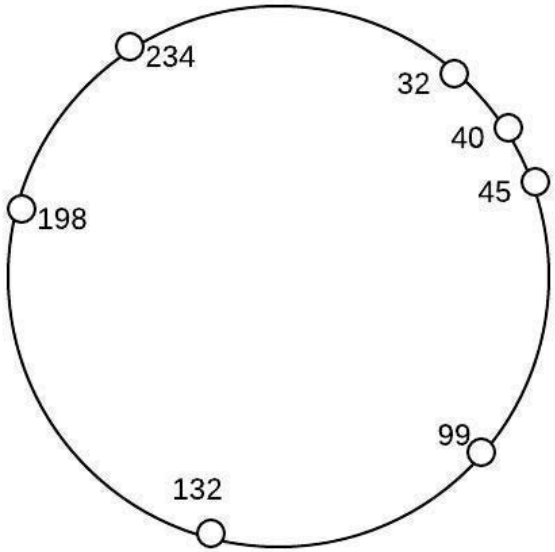
Distributed Hash Table (DHT) - Seguridad

Resumen: Aumenta el costo del comportamiento malicioso

- El esfuerzo para atacar de forma eficaz una DHT es mucho mayor que en una red no estructurada.
  - No puedes elegir tus vecinos libremente ni qué datos guardar en la tabla de punteros.
  - No puedes posicionarte arbitrariamente.
- Obliga a los atacantes a crear ataques masivos, coordinados y sofisticados, elevando el umbral técnico y económico.

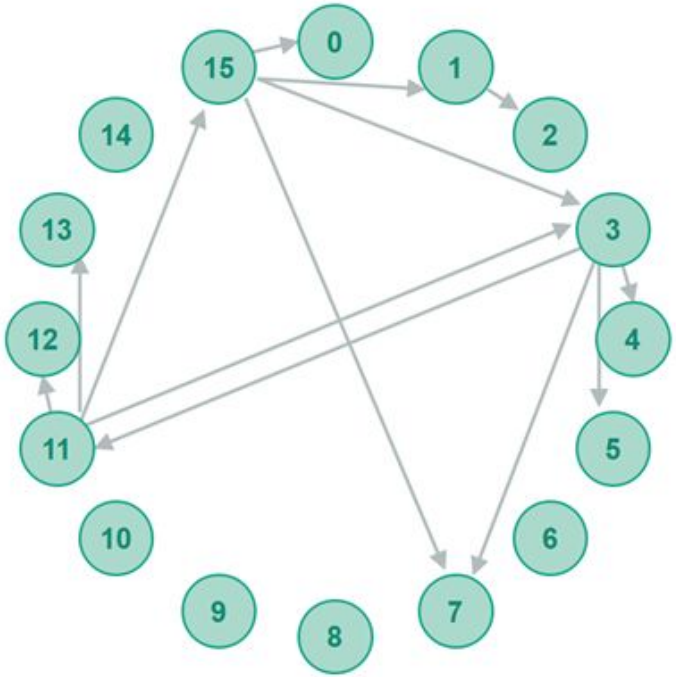
Distributed Hash Table (DHT) - *Chord*

- Se utiliza un DHT donde los nodos se organizan lógicamente en un anillo.
- Un elemento de datos con una clave k se asigna al nodo con el identificador más pequeño cuyo `hash(id_nodo) ≥ k`.
  - Este nodo se conoce como el sucesor de k.



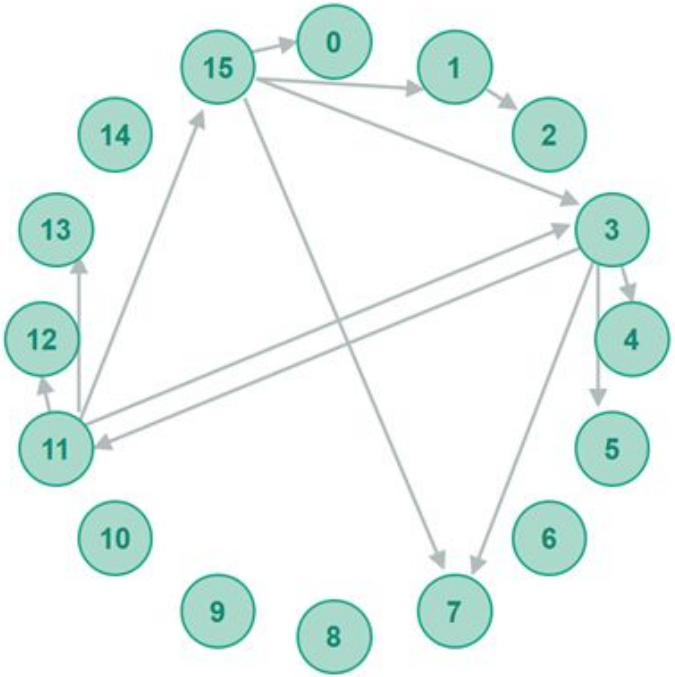
Distributed Hash Table (DHT) - *Chord*

- Se utiliza un DHT donde los nodos se organizan lógicamente en un anillo.
- Un elemento de datos con una clave k se asigna al nodo con el identificador más pequeño cuyo `hash(id_nodo) ≥ k`.
  - Este nodo se conoce como el sucesor de k.
- Cada nodo contiene una *Finger Table* que corresponde a su tabla de enrutamiento.
- La tabla se construyen de manera que la longitud de la ruta más corta entre cualquier par de nodos sea óptima.
  - Orden O(log N) con N igual al tamaño de la red.

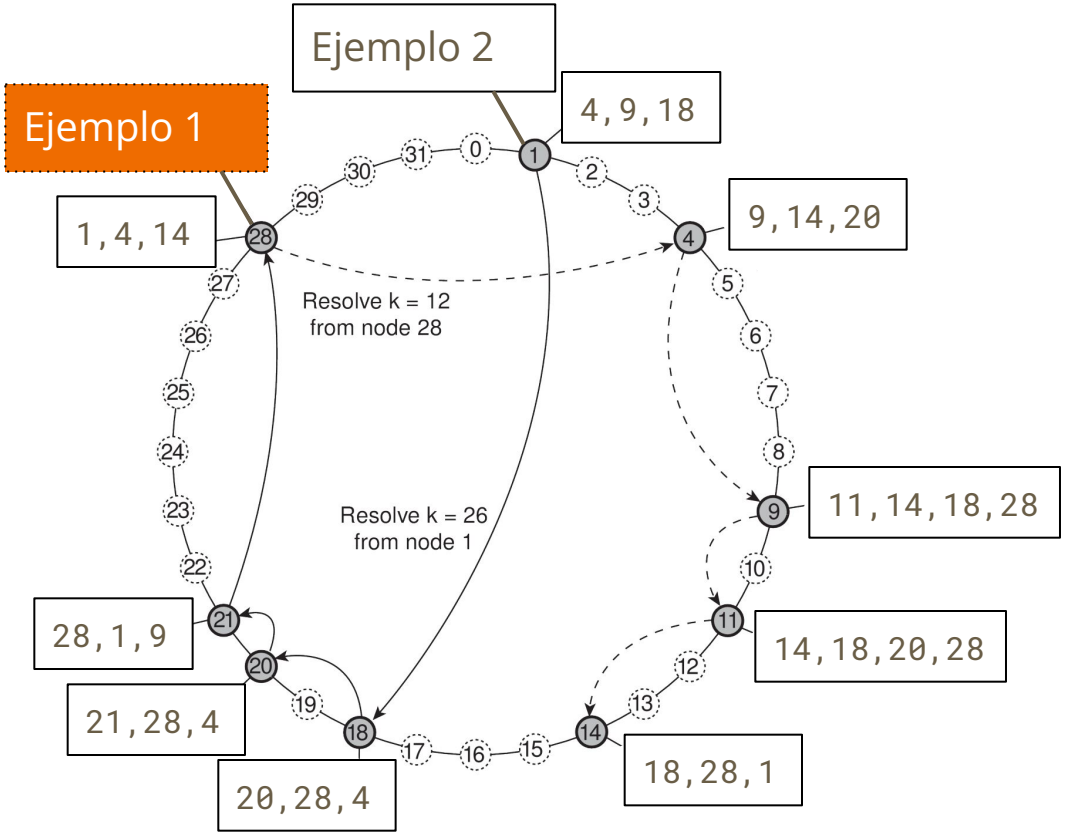


Distributed Hash Table (DHT) - *Chord* - Búsqueda

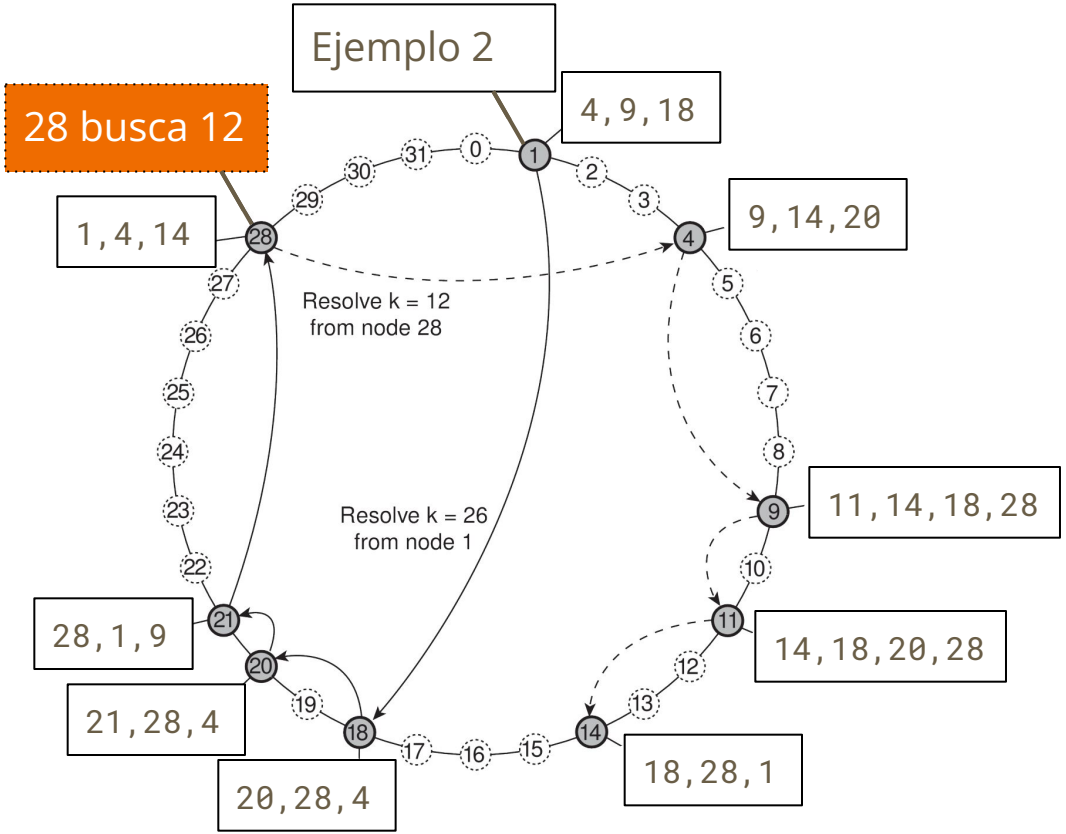
- Al momento de buscar una *key*, se busca el nodo más cercano **que no supere la *key***.
  - Aunque, si el nodo que busca ya está asociado una *key* mayor a la buscada, debe darse la vuelta en el anillo.
- Caso ideal: llegar al nodo cuya *key* es igual a la buscada.
- Caso realista: Se llega al nodo cuya *key* es la más cercana y él determina si existe el nodo deseado o debe "pasarse" para buscar la información con el sucesor de la *key*.



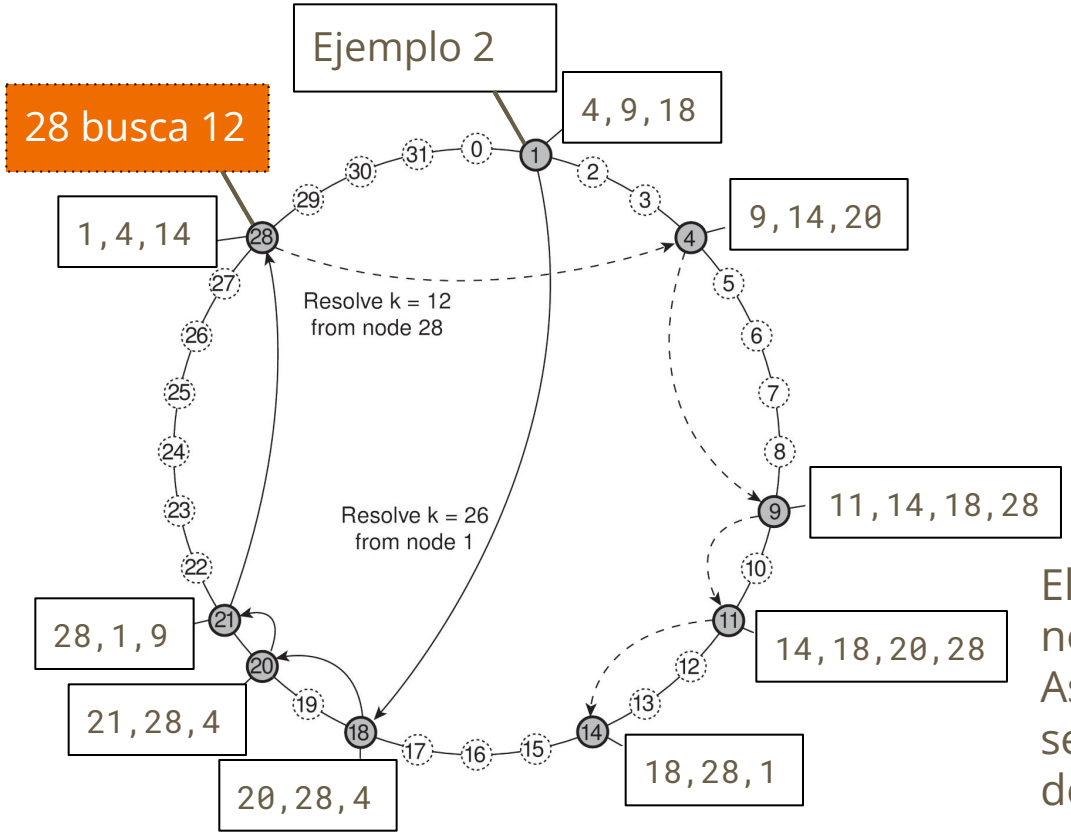
Distributed Hash Table (DHT) - *Chord* - Búsqueda



Distributed Hash Table (DHT) - *Chord* - Búsqueda

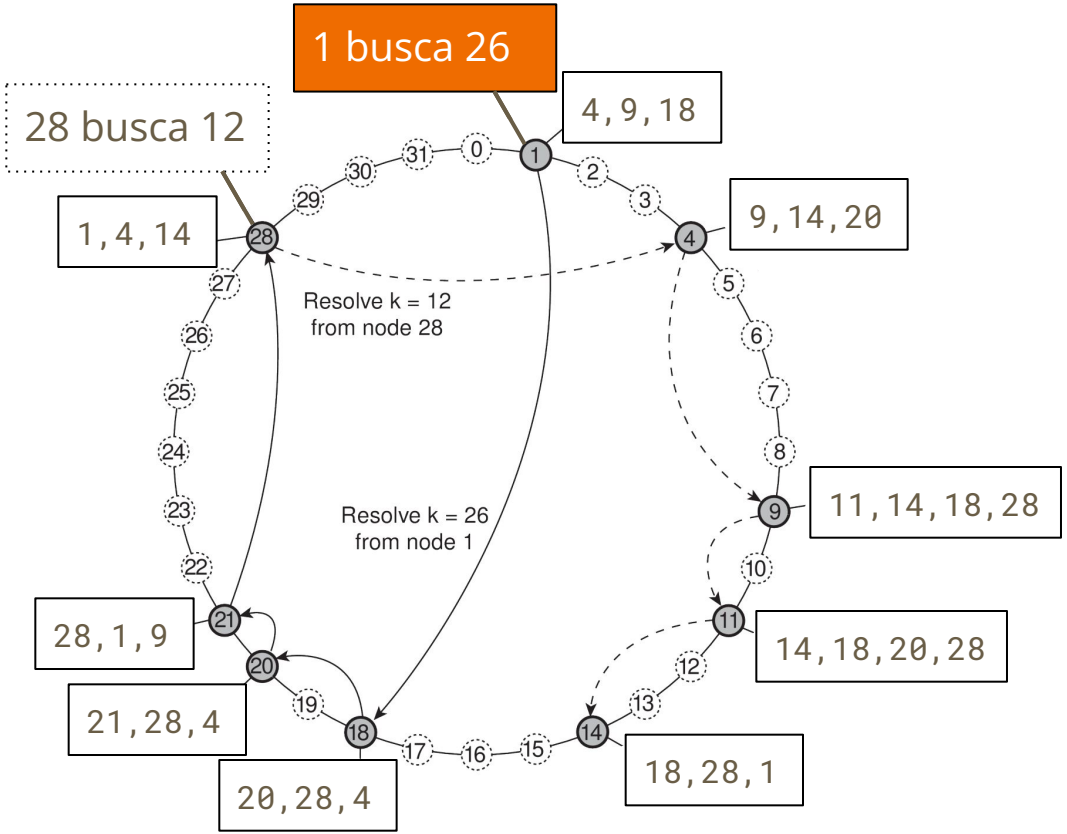


Distributed Hash Table (DHT) - *Chord* - Búsqueda



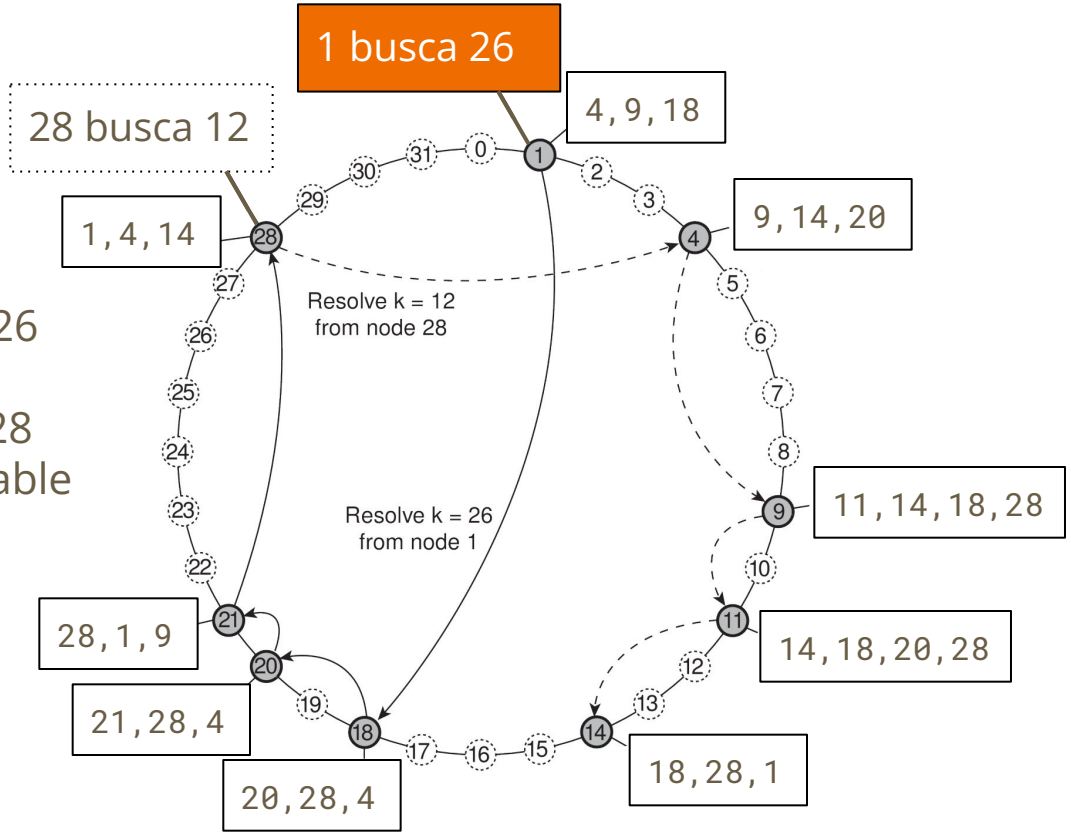
El nodo con key 12 no existe.
Así que el nodo 14 se hace responsable de sus datos.

Distributed Hash Table (DHT) - *Chord* - Búsqueda



Distributed Hash Table (DHT) - *Chord* - Búsqueda

El nodo con key 26 no existe.
Así que el nodo 28 se hace responsable de sus datos.



Distributed Hash Table (DHT) - *Chord* - *Churn*

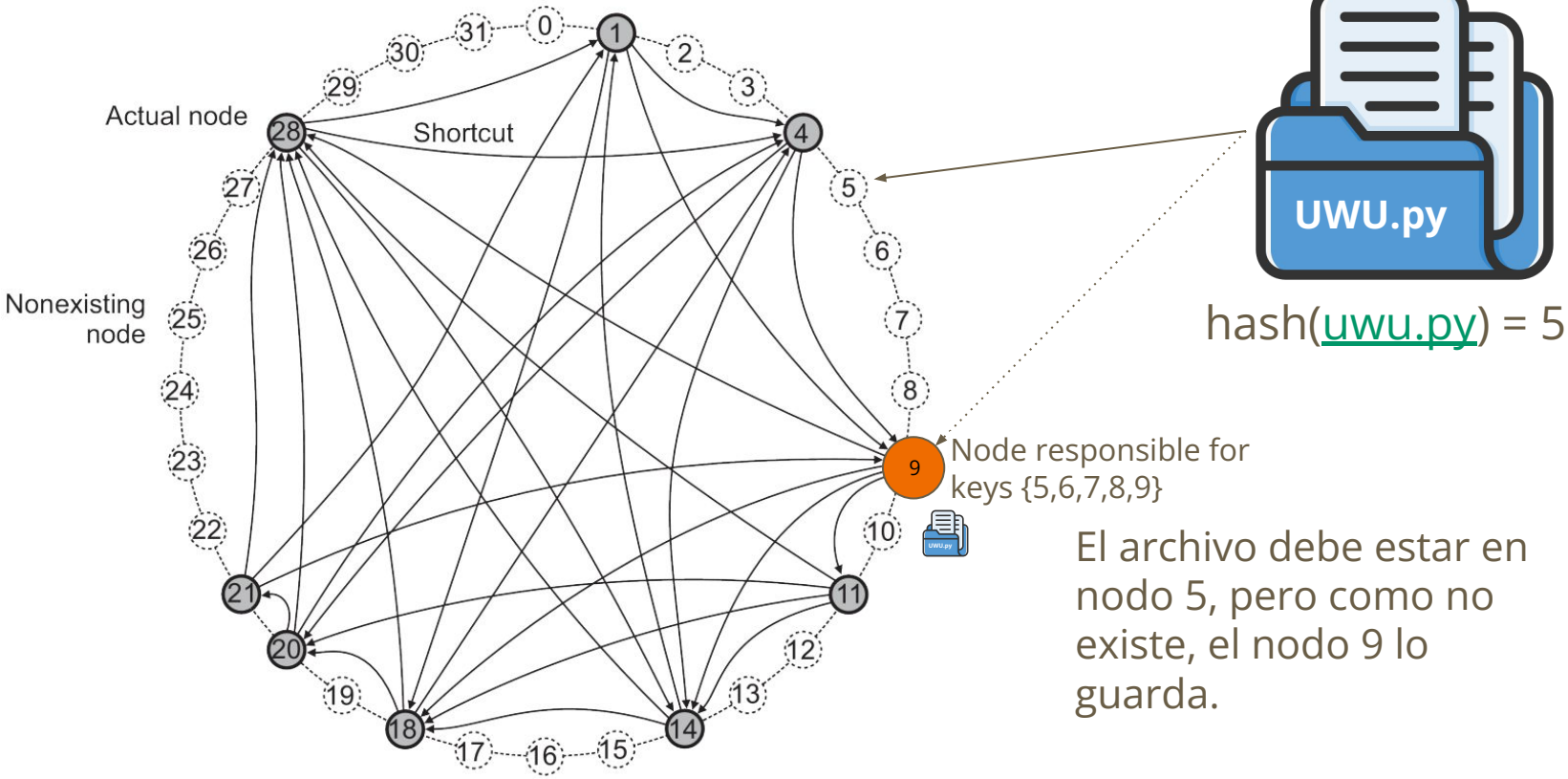
Entrada de nodo

- Cuando un nuevo nodo se une a la superposición, obtiene los datos necesarios para construir su *finger table* y el estado requerido de los miembros existentes a partir de su sucesor.
- El nodo se inserta en la posición que corresponda según su *key* (`hash(id_nodo)`), y los datos que él se debe responsabilidad se transfieren.

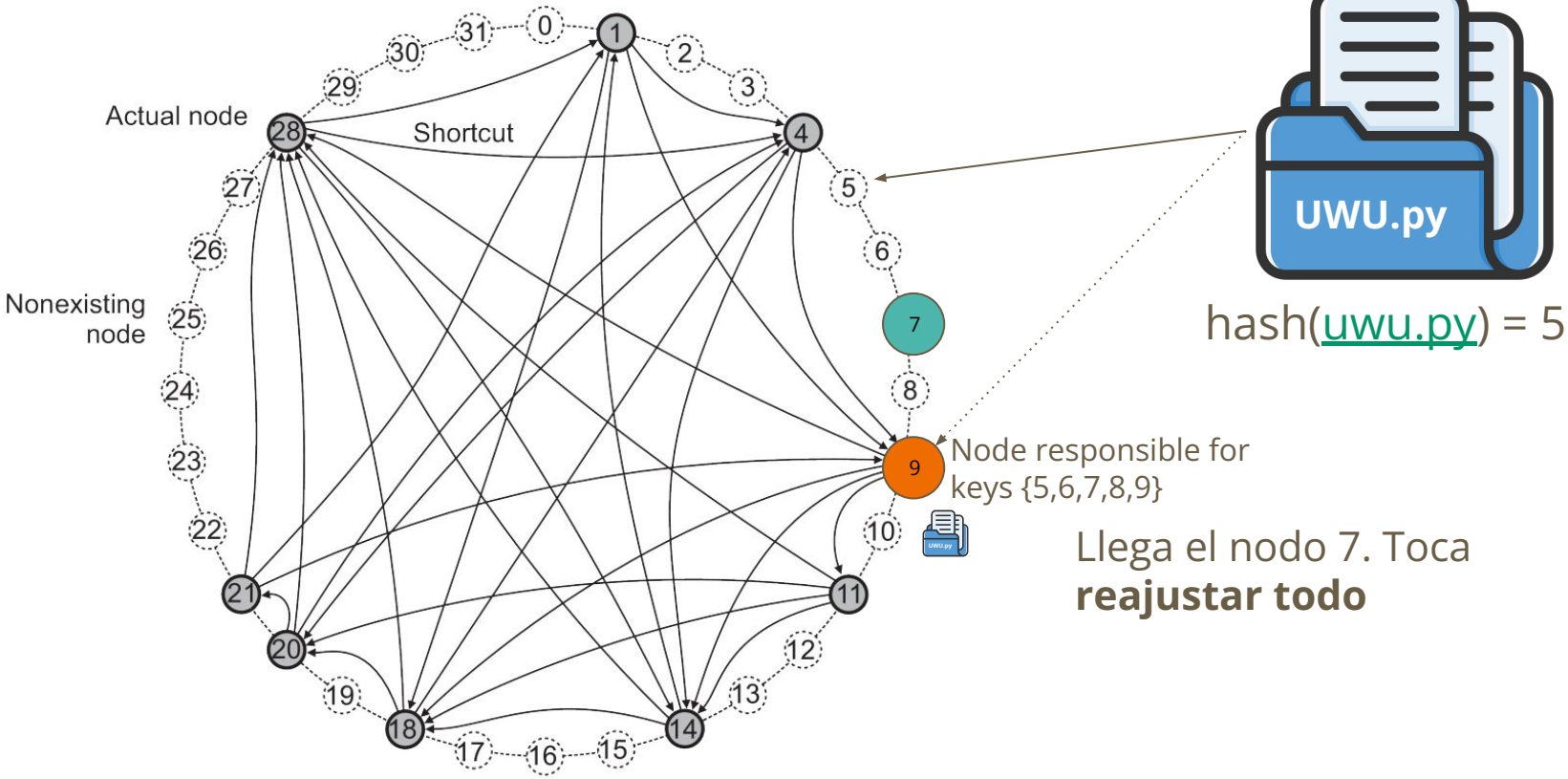
Salida salida de nodo

- Si un nodo falla o sale (voluntaria o involuntariamente), los nodos predecesores puede detectar su ausencia y reconfigurar su tabla para reflejar los cambios requeridos.

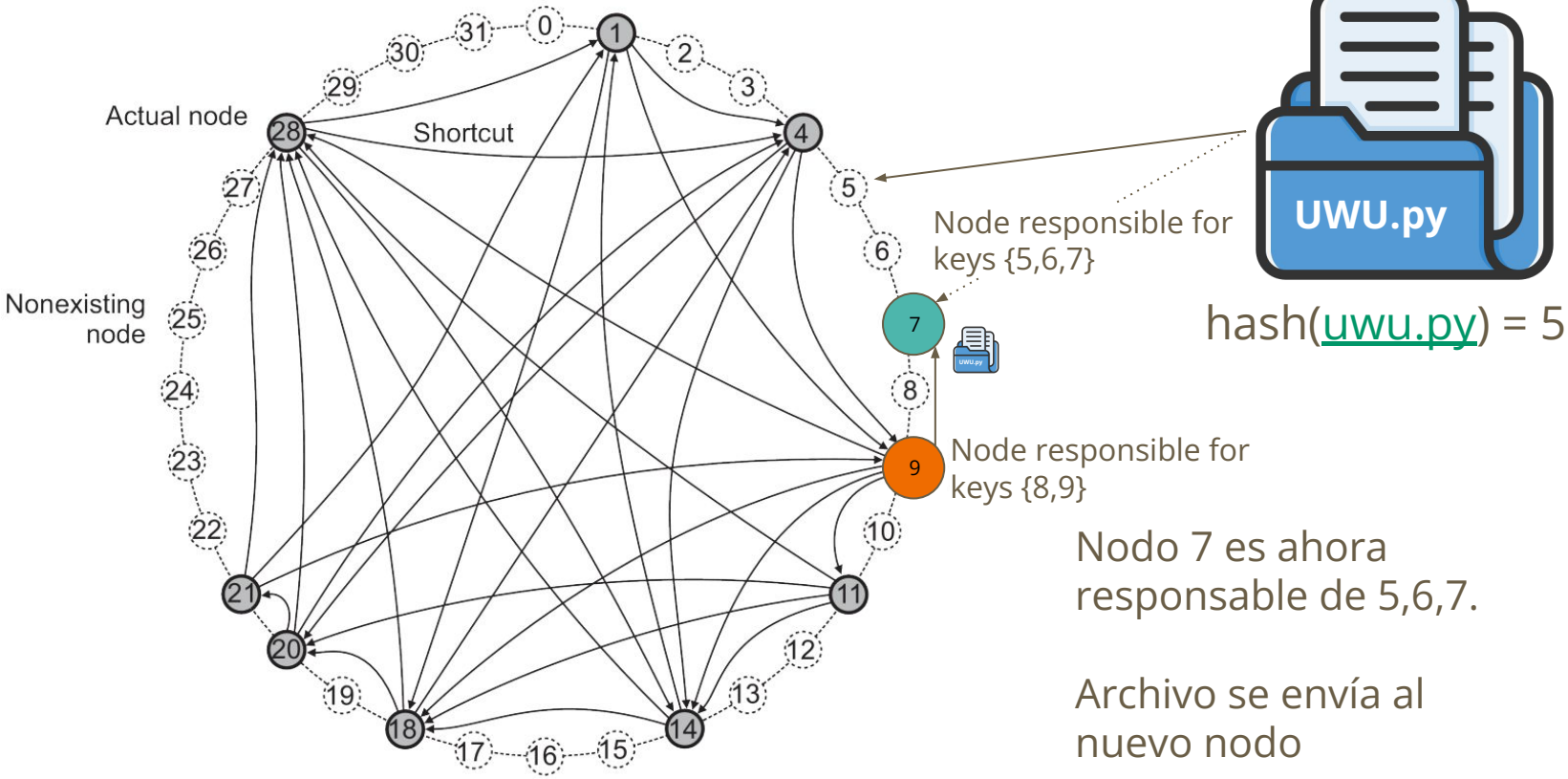
Distributed Hash Table (DHT) - *Chord* - *Churn*



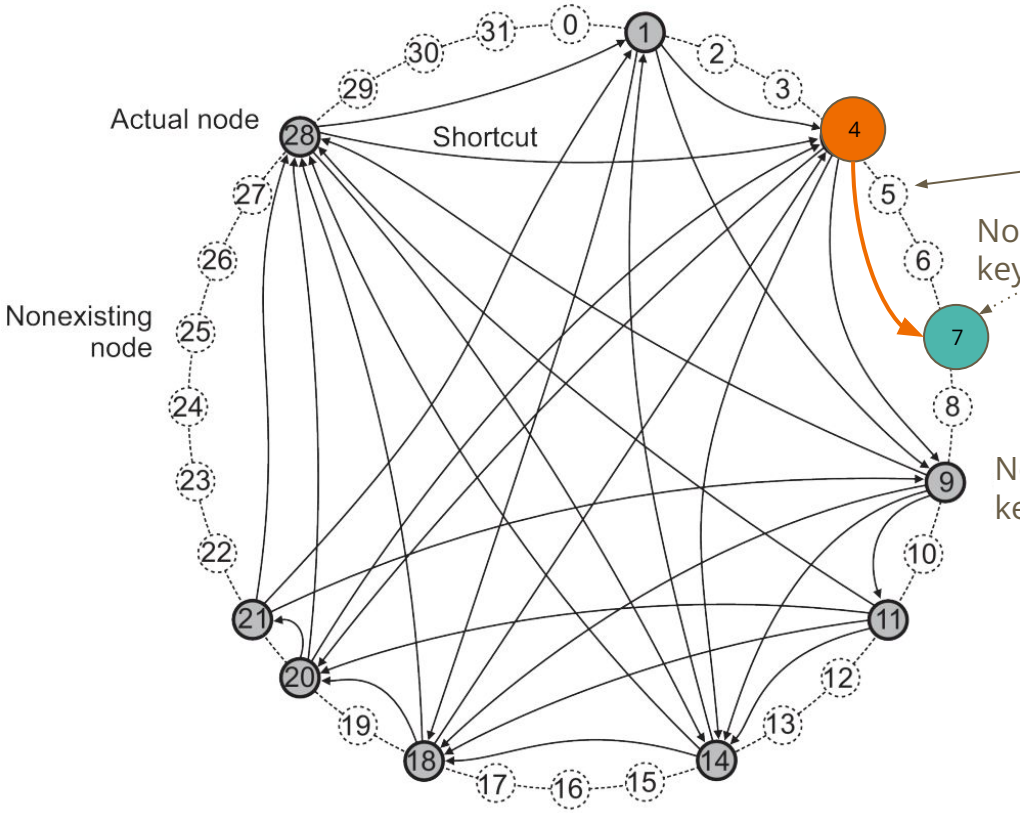
Distributed Hash Table (DHT) - *Chord* - *Churn*



Distributed Hash Table (DHT) - *Chord* - *Churn*



Distributed Hash Table (DHT) - *Chord* - Churn



Node responsible for keys {5,6,7}

hash(uwu.py) = 5

Node responsible for keys {8,9}

Luego queda actualizar *Finger Table* del nodo 4 para que sepa de la existencia del nodo 7.

Distributed Hash Table (DHT) - *Chord* - Seguridad

Chord hereda los aspectos de seguridad de una DHT.

1. Difícil secuestrar rutas sin alterar muchos nodos

***Finger tables* y rutas verificables**

- Las rutas están definidas por los *finger tables* y permite hacer verificación cruzada.
  - Si A consulta a B, y B le da una ruta falsa, A puede notar inconsistencias con otros nodos.
  - A pregunta también a C y compara.
- El atacante no puede insertar nodos intermediarios en todas las rutas sin ser detectado.

Distributed Hash Table (DHT) - *Chord* - Seguridad

Chord hereda los aspectos de seguridad de una DHT.

2. Nodos no pueden elegir qué datos almacenar

`sucessor(k)` *definido según hash*

- Si quiero almacenar una clave k, todos los nodos saben que debe buscar en `sucessor(k)`.
- El atacante no puede elegir fácilmente ser el nodo responsable de la clave k, a menos que pueda controlar su ID, lo que implica romper el *hash*.

Distributed Hash Table (DHT) - *Chord* - Seguridad

Chord hereda los aspectos de seguridad de una DHT.

3. La red se adapta a nodos que entran/salen

Redundancia de información y *Churn* eficiente de implementar.

- Normalmente, las *keys* se replican en `sucessor(k)`, `sucessor(sucessor(k))`, etc.
- Además, ante la llegada o salida de nodos, solo implica actualizar la *finger table* de algunos nodos.
- Si se detecta un nodo malicioso, es fácil eliminarlo de las *finger table* para ignorarlo y no perder la información que este almacenaba.

Distributed Hash Table (DHT) - Otros ejemplos

Pastry

- Usa prefijos comunes en las IDs para enrutar.
- Cada salto lleva a un nodo cuyo prefijo comparte más dígitos con el destino.
- Similar a un cartero que se acerca paso a paso según coincidencia del código postal.
- Ejemplo - `Buscar(4444)`
  - 4XXX → 44XX → 444X → 4444

Distributed Hash Table (DHT) - Otros ejemplos

Pastry

- Usa prefijos comunes en las IDs para enrutar.
- Cada salto lleva a un nodo cuyo prefijo comparte más dígitos con el destino.
- Similar a un cartero que se acerca paso a paso según coincidencia del código postal.
- Ejemplo - `Buscar(4444)`
  - 4XXX → 44XX → 444X → 4444

Kademlia

- Usa XOR para medir distancia entre nodos.
- Cada salto lleva a un nodo que comparte más dígitos, en cualquier posición, con el destino.
- Como un juego donde se pregunta a quienes tienen números más parecidos al objetivo.
- Ejemplo - `Buscar(4444)`
  - 4XX4 → 44X4 → 4444

Poniendo a prueba lo que hemos aprendido 👀

Se instala una red P2P para administrar las cámaras de vigilancia en distintas casas de un barrio. Este servicio permite alertar automáticamente a los demás vecinos ante cualquier problema. Un día, uno de los vecinos indica ser víctima de un **ataque Eclipse**.

¿Cuál de los siguientes casos corresponde a una **explicación** que justifique este ataque?

a. Varias cámaras de diferentes vecinos recibían mensajes que les indicaban desactivarse temporalmente.

b. La red P2P colapsó varias veces durante la noche debido a una sobrecarga masiva de solicitudes simultáneas provenientes de múltiples cámaras.

c. Una de las cámaras intentó asumir el rol de autoridad y comenzó a enviar alertas contradictorias a distintos vecinos.

d. Las cámaras de distintos vecinos comenzaron a detectar como vecinas a múltiples cámaras recién integradas en la red, que en realidad no existían físicamente y no respondían a las alertas.

e. Sus cámaras enviaban alertas sobre un robo, pero los demás vecinos no recibían la notificación.

Poniendo a prueba lo que hemos aprendido 👀

Se instala una red P2P para administrar las cámaras de vigilancia en distintas casas de un barrio. Este servicio permite alertar automáticamente a los demás vecinos ante cualquier problema. Un día, uno de los vecinos indica ser víctima de un **ataque Eclipse**.

¿Cuál de los siguientes casos corresponde a una **explicación** que justifique este ataque?

a. Varias cámaras de diferentes vecinos recibían mensajes que les indicaban desactivarse temporalmente.

b. La red P2P colapsó varias veces durante la noche debido a una sobrecarga masiva de solicitudes simultáneas provenientes de múltiples cámaras.

c. Una de las cámaras intentó asumir el rol de autoridad y comenzó a enviar alertas contradictorias a distintos vecinos.

d. Las cámaras de distintos vecinos comenzaron a detectar como vecinas a múltiples cámaras recién integradas en la red, que en realidad no existían físicamente y no respondían a las alertas.

e. Sus cámaras enviaban alertas sobre un robo, pero los demás vecinos no recibían la notificación.

Próximos eventos

Próxima clase

- Casos aplicados de sistemas distribuidos
  - ¿Cómo funciona VPN, TOR, Eduroam y *Bittorrent*?
  - ¿Qué relación tienen con los sistemas distribuidos?
  - ¿Cómo implementan la seguridad en dichos sistemas?

Evaluación

- Este jueves se publica el último control, va a evaluar estas últimas 4 clases.
- No pateen la investigación para tan al final 👀.

IIC2523
Sistemas Distribuidos

Hernán F. Valdivieso López
(2025 - 2 / Clase 23)