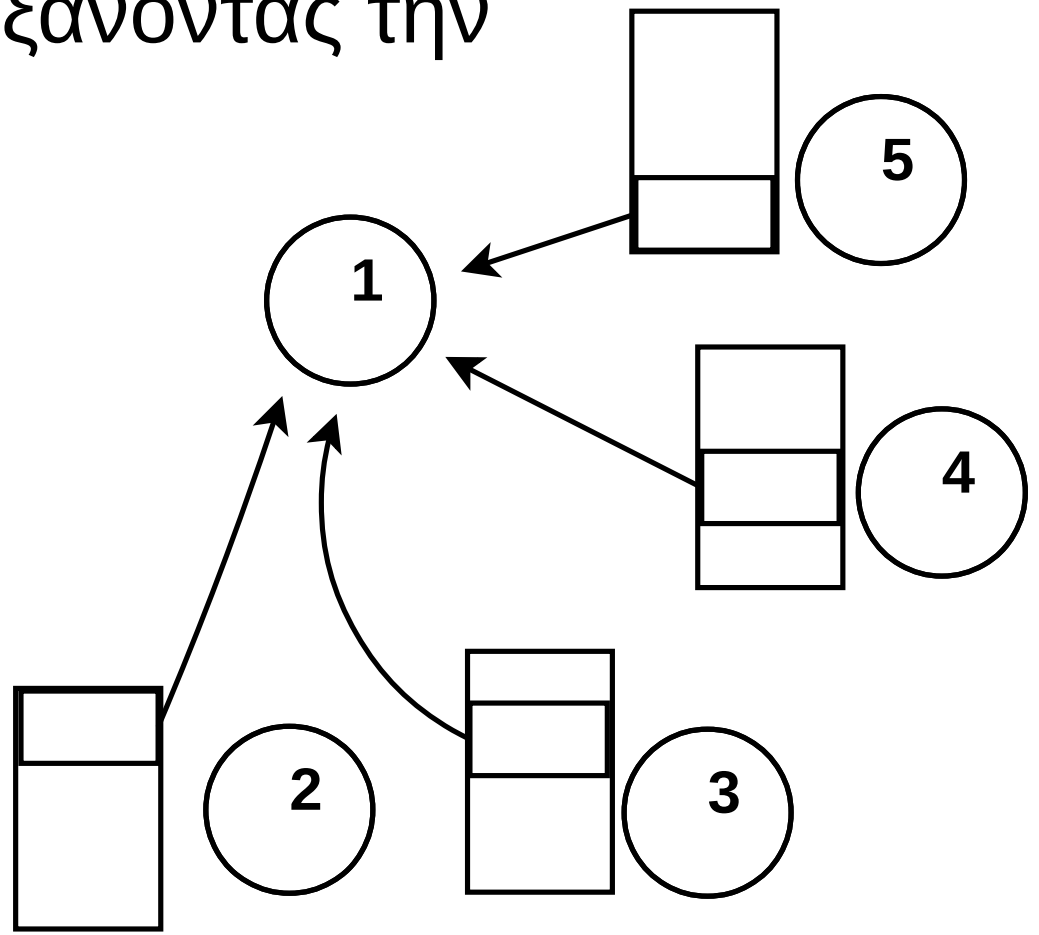
Chord: A scalable peer-to-peer lookup service for internet applications

Paper: Chord: A scalable peer-to-peer lookup service for internet applications

Peer-to-peer: Distributed Architecture που τμηματοποιεί εργασίες ανάμεσα σε peers/κόμβους ενός δικτύου.

Παράδειγμα peer-to-peer

- Πχ. ο κόμβος 1 θέλει έναν πίνακα. Οι υπόλοιποι κόμβοι συνεργάζονται και στέλνουν ένα μέρος του πίνακα στον 1, αυξάνοντας την αποδοτικότητα.



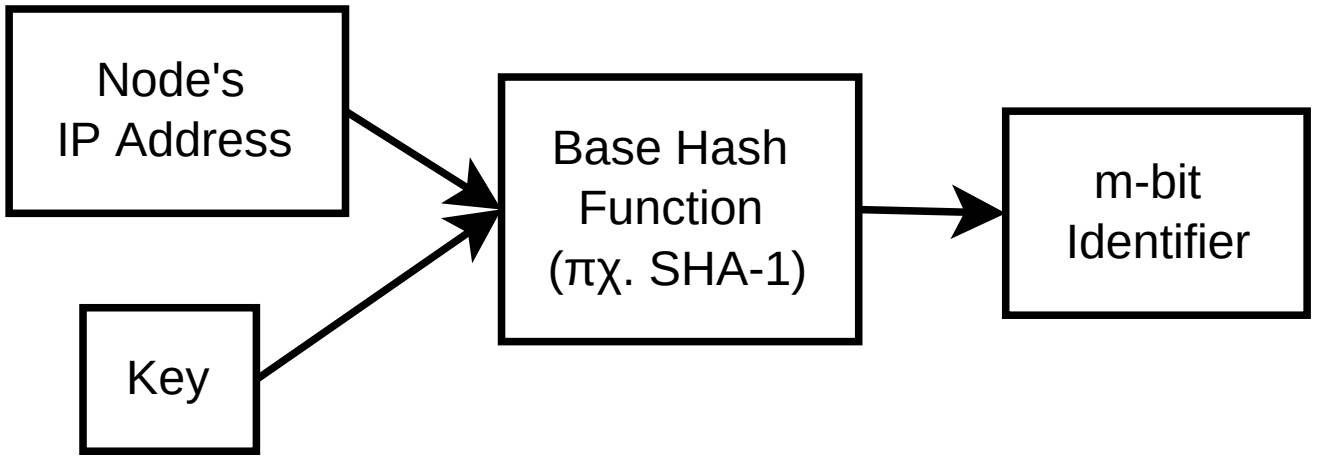
Βασική λειτουργία του Chord

- Δεδομένου ενός κλειδιού, αντιστοιχεί το κλειδί σε έναν κόμβου του δικτύου.
- Πολυπλοκότητα $O(\log_2 N)$
- Αυτό επεκτείνεται στην αποθήκευση δεδομένων αν αντιστοιχίσουμε τα δεδομένα σε κλειδί.
- Πώς γίνεται η αντιστοίχιση κλειδί->κόμβος?

Το Chord κάνει χρήση του Consistent Hashing

Consistent Hashing: Για κάθε κόμβο ή κλειδί δίνει έναν m-bit identifier

χρησιμοποιώντας κάποια base hash function (SHA-1).



Μπαίνει είτε ένα η IP address του κόμβου είτε ένα κλειδί και βγαίνει ένας m-bit identifier.

Παρακάτω, όταν αναφερόμαστε σε key/node θα εννοούμε και τον identifier του.

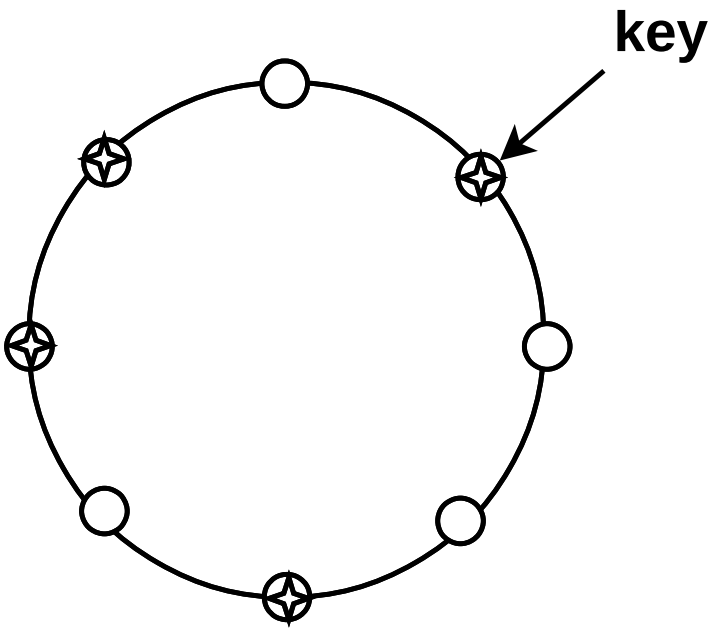
Αντιστοίχιση κλειδιού -> κόμβου με consistent hashing

- Όλοι οι δυνατοί m-bit identifiers (2^m στο πλήθος) τοποθετούνται σε identifier circle (σχήμα μετά).
- Οι κόμβοι τοποθετούνται στον κύκλο στις θέσεις που προκύπτουν από το hashing.
- Κλειδί k πάει στον κόμβο που:

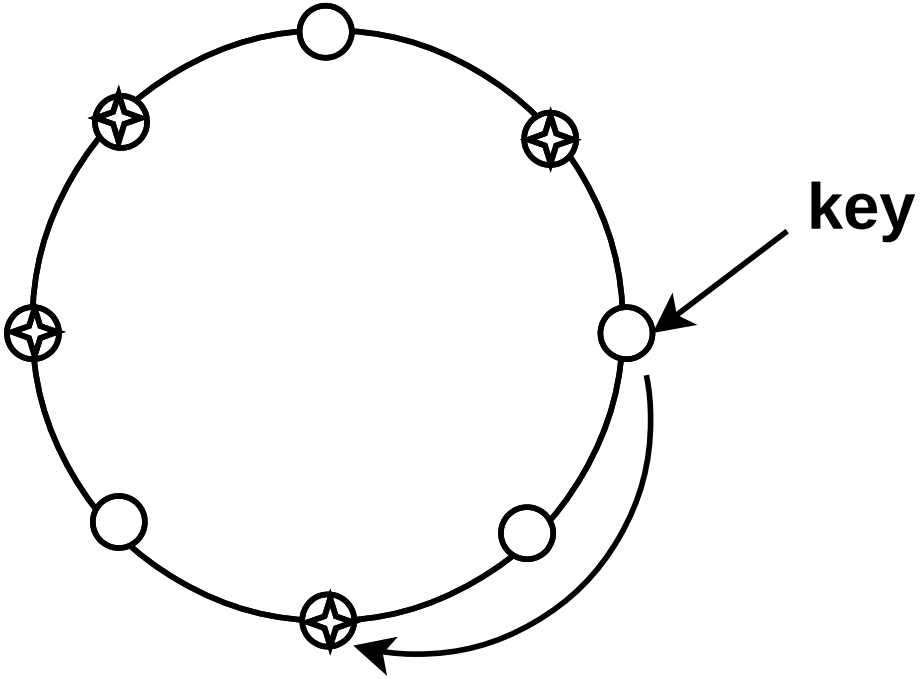
  - είτε είναι ίσος με το k

  - είτε είναι ο πρώτος μεγαλύτερος από αυτό.
- Στον identifier circle, το κλειδί θα πάει στον 1ο κόμβο που θα βρει κινούμενο clockwise.
- Συμβολισμός: successor(k)

Αντιστοίχιση κλειδιού -> κόμβου



κλειδί ίσο με κόμβο



κλειδί πάει στον πρώτο κόμβο που βρίσκει κινούμενο πάνω στον κύκλο.

Παράδειγμα

- m = 4 => 2^4 = 16 identifiers.
- 4 nodes: 0, 4, 10, 13.
- 5 keys: 0, 3, 4, 5, 11.
- Τα κλειδιά αντιστοιχούνται στους κόμβους ως εξής:

  successor(0) = 0

  successor(3) = 4

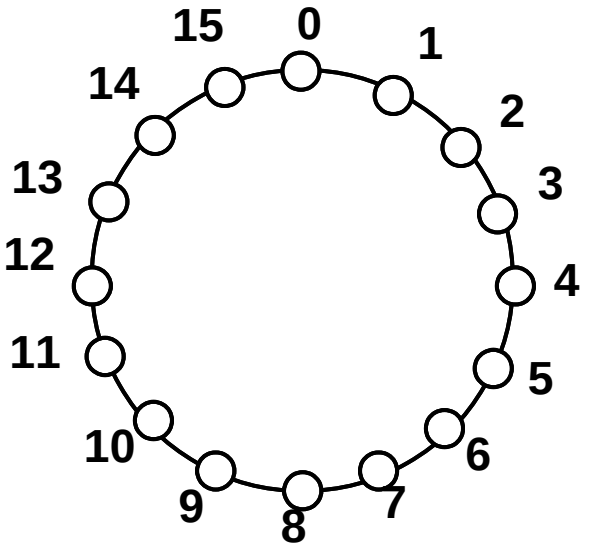
  successor(4) = 4

  successor(5) = 10

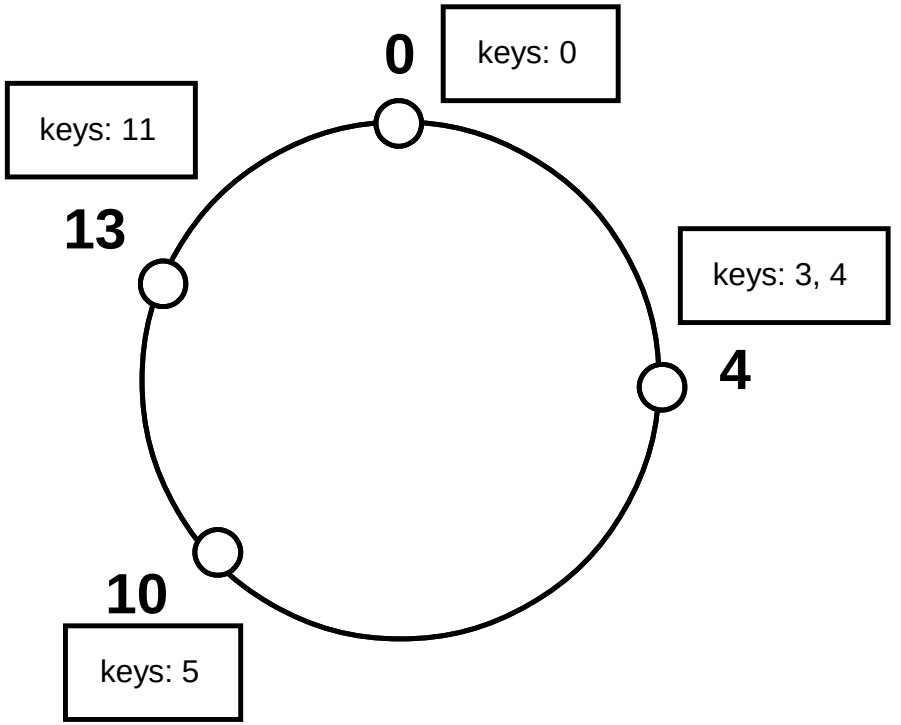
  successor(11) = 13

Παράδειγμα (συνέχεια)

- Identifier Circle:



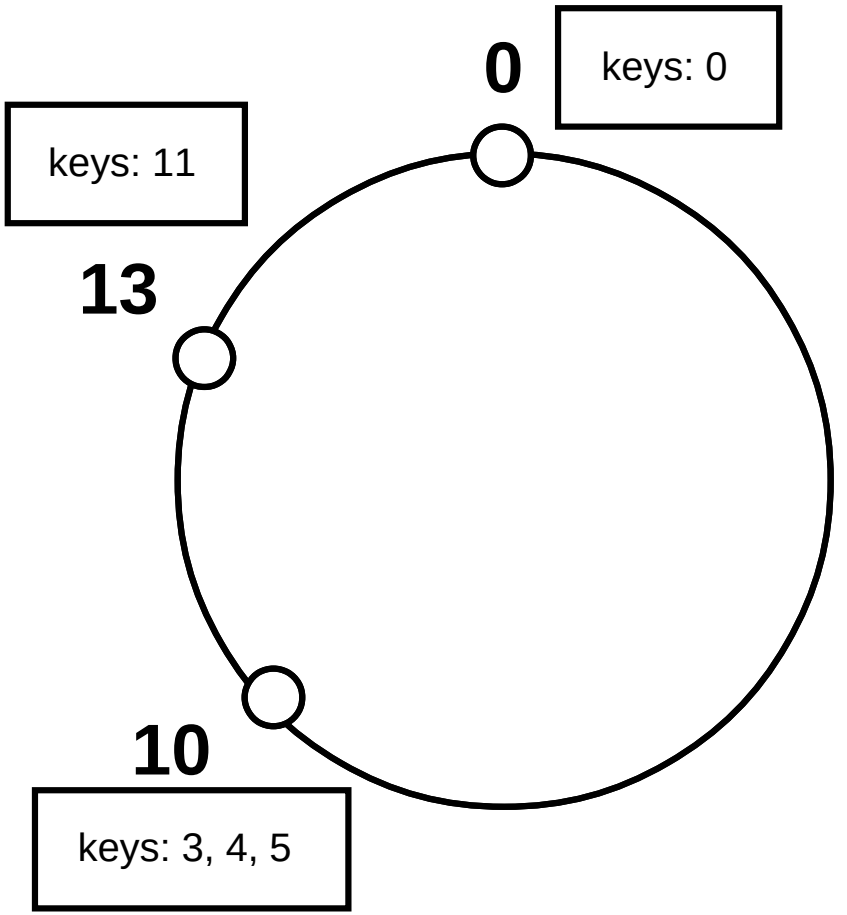
Με τα στοιχεία:



Node Leave

- Όταν κάποιος κόμβος n φύγει από το δίκτυο, τα κλειδιά που είχαν ανατεθεί σε αυτόν θα ανατεθούν στον successor(n).

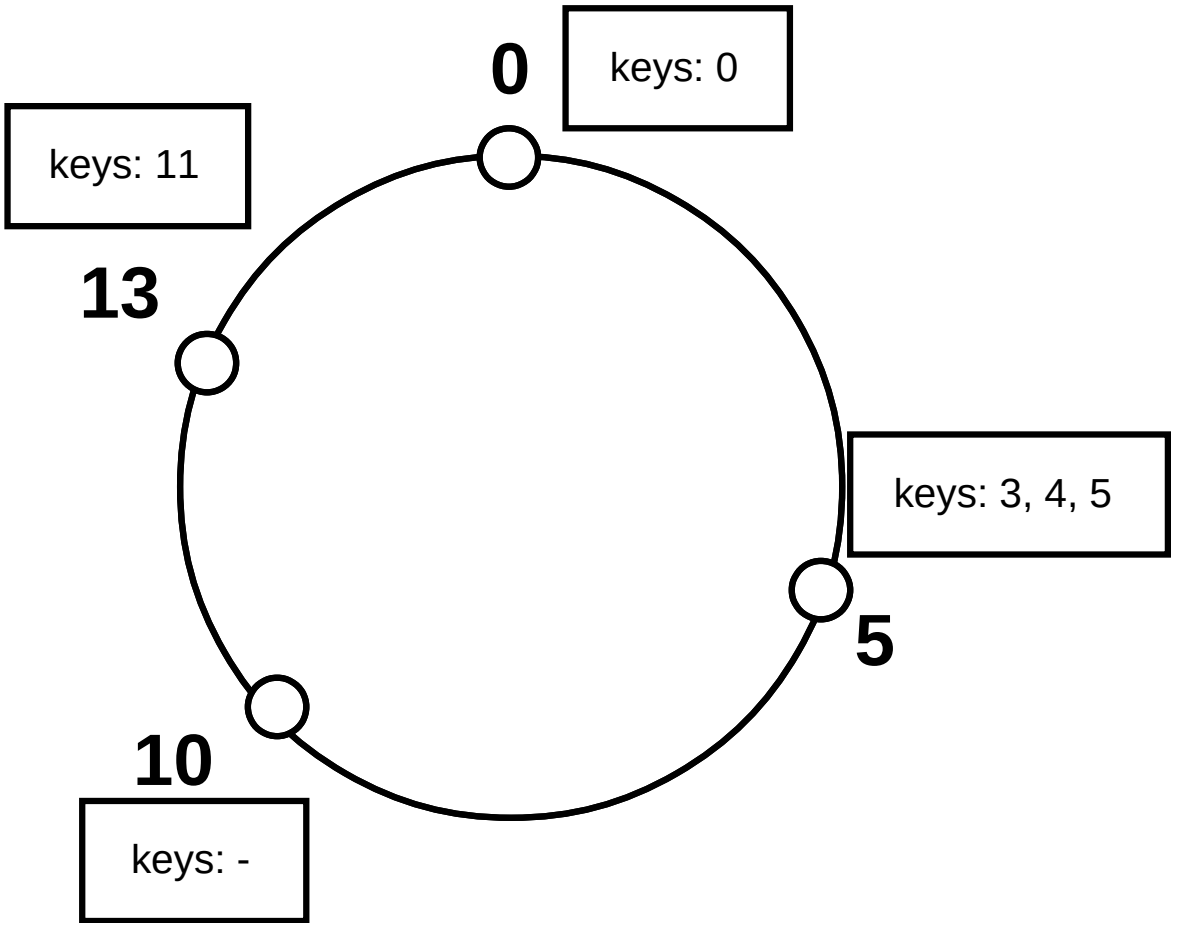
  πχ. node 4 leaves:



Node Join

- Όταν ένας κόμβος n συνδεθεί στο δίκτυο, κάποια από τα κλειδιά του κόμβου successor(n) θα ανατεθούν στον n.

  πχ. node 5 joins:

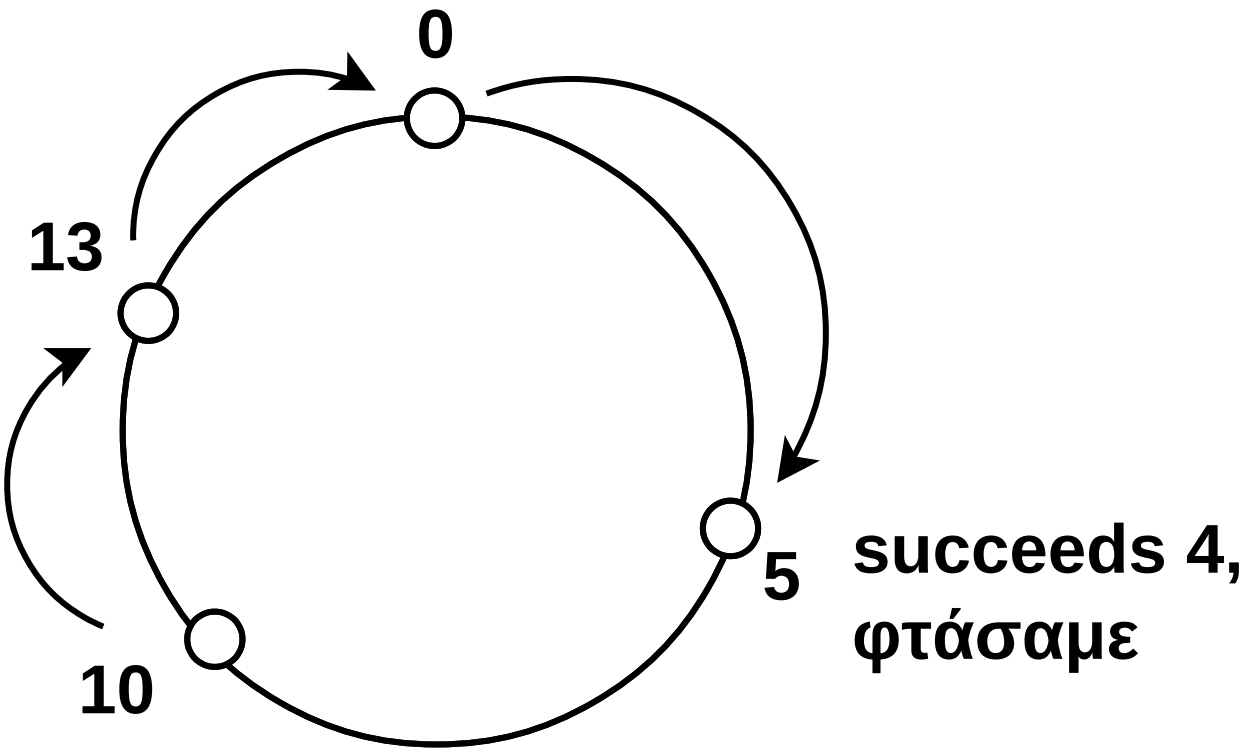


- Κατά μέσο όρο, κάθε κόμβος είναι υπεύθυνος για K/N κλειδιά, όπου K το πλήθος των κλειδιών και N το πλήθος των κόμβων.
- Όταν ένας κόμβος συνδέεται ή φεύγει, τότε O(K/N) κλειδιά αλλάζουν κόμβο (από/προς τον κόμβο που άλλαξε κατάσταση).

Scalability

- Ελάχιστη πληροφορία για δυνατή επικοινωνία: Κάθε κόμβος να γνωρίζει το successor του.

  πχ. Ο κόμβος 10 θέλει να βρει το successor(4):

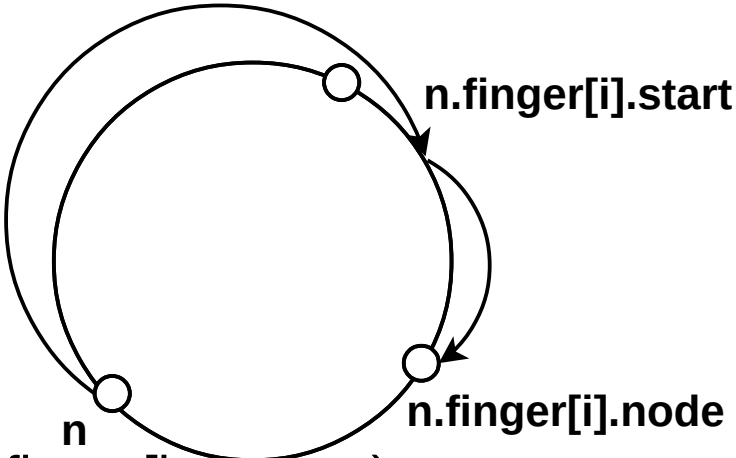


Scalability

- Όμως αργό (O(N))
- Στο chord κάθε κόμβος έχει περισσότερες συνδέσεις με χρήση του finger table που εξηγείται παρακάτω (O(logN)).

Finger Table

- Κάθε κόμβος έχει routing table με m καταχωρήσεις (finger table).
- Σε κόμβο n: Η i-οστή καταχώρηση περιέχει τον κόμβο s = successor(n + 2^{i-1})
- s: i-οστό finger του n



- Συμβολισμοί:

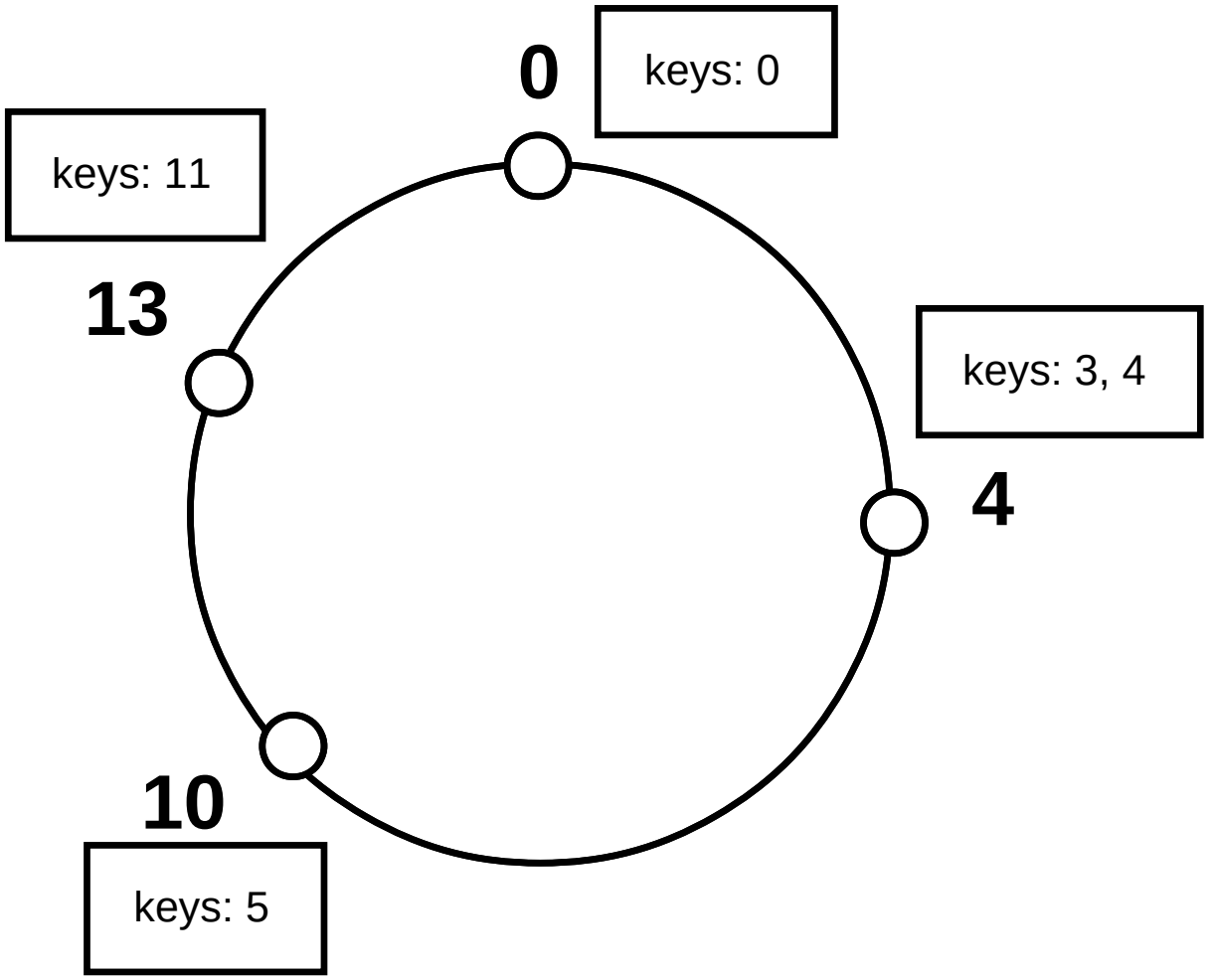
  s = n.finger[i].node

  n.finger[i].start = n + 2^{i-1}

  n.finger[i].interval = [n.finger[i].start , n.finger[i+1].start)

  n.successor(i) = n.finger[1].node
- Όλες οι πράξεις είναι modulo 16 ώστε να παραμένουμε πάνω στον κύκλο

Παράδειγμα



node 0: start	interval	successor
1	[1,2)	4
2	[2,4)	4
4	[4,8)	4
8	[8,0)	10
keys: 0		

node 4: start	interval	successor
5	[5,6)	10
6	[6,8)	10
8	[8,12)	10
12	[12,4)	13
keys: 3, 4		

node 10: start	interval	successor
11	[11,12)	13
12	[12,14)	13
14	[14,2)	0
2	[2,10)	4
keys: 5		

node 13: start	interval	successor
14	[14,15)	0
15	[15,1)	0
1	[1,5)	4
5	[5,13)	10
keys: 11		

Επικοινωνία

- πχ. ο κόμβος 10 θέλει το successor(1).
- Το 1 περιέχεται στο 10.finger[3].interval = [14,2).
- Άρα το 10 ρωτάει το 10.finger[3].node = 0.

  (πρέπει 0 να προηγείται του 1)
- Ο κόμβος 0 βλέπει άμεσα από το finger table του ότι successor(1) = 5, και το λέει στον 10.
- Αποστάστεις του finger table διπλασιάζονται => Αναζήτηση μειώνει στο μισό την απόσταση ανά βήμα => O(logN)

Node Join με Finger tables

- Ελάχιστη πληροφορία για σωστή επικοινωνία:

  - Σωστά successors

  - Για κάθε κλειδί k, ο κόμβος successor(k) είναι υπεύθυνος για αυτό.

- Για πιο γρήγορη επικοινωνία (όχι απαραίτητο):

  - Σωστό finger table

Node Join με Finger tables

- Για απλοποίηση, κάθε κόμβος έχει δείκτη προς τον προηγούμενό του (predecessor pointer).
- 3 Διεργασίες όταν συνδέεται νέος κόμβος n:

  - Initialize node n

  - Update existing nodes

  - Notify higher layer

Initialize node n

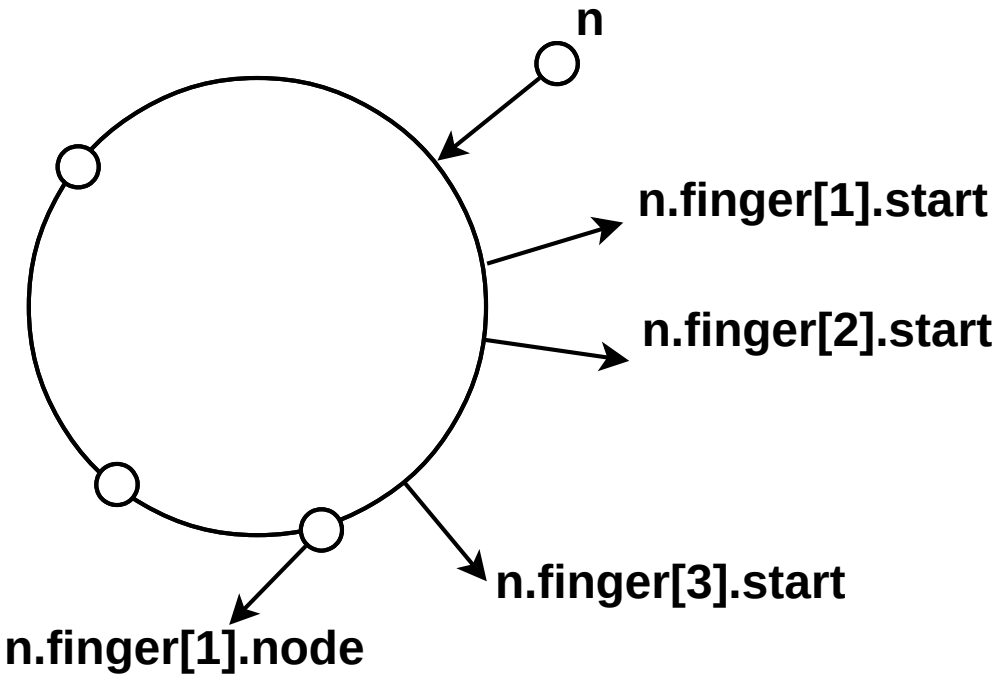
- Κόμβος n γνωρίζει για έναν άλλον κόμβο n' του δικτύου μέσω εξωτερικού μηχανισμού.
- Ο n ζητάει από n' να βρει τα στοιχεία του finger table του n.
- - Πιο απλό, αλλά αργό: ο n' καλεί find_successor(i) για κάθε i του finger table του n.

  - Πιο αποδοτικό: αν n.finger[i].node>=n.finger[i+1].start , τότε το i-οστό finger είναι και το (i+1)-οστό.
- O(logN)

Initialize node n παράδειγμα (αποδοτικός τρόπος)

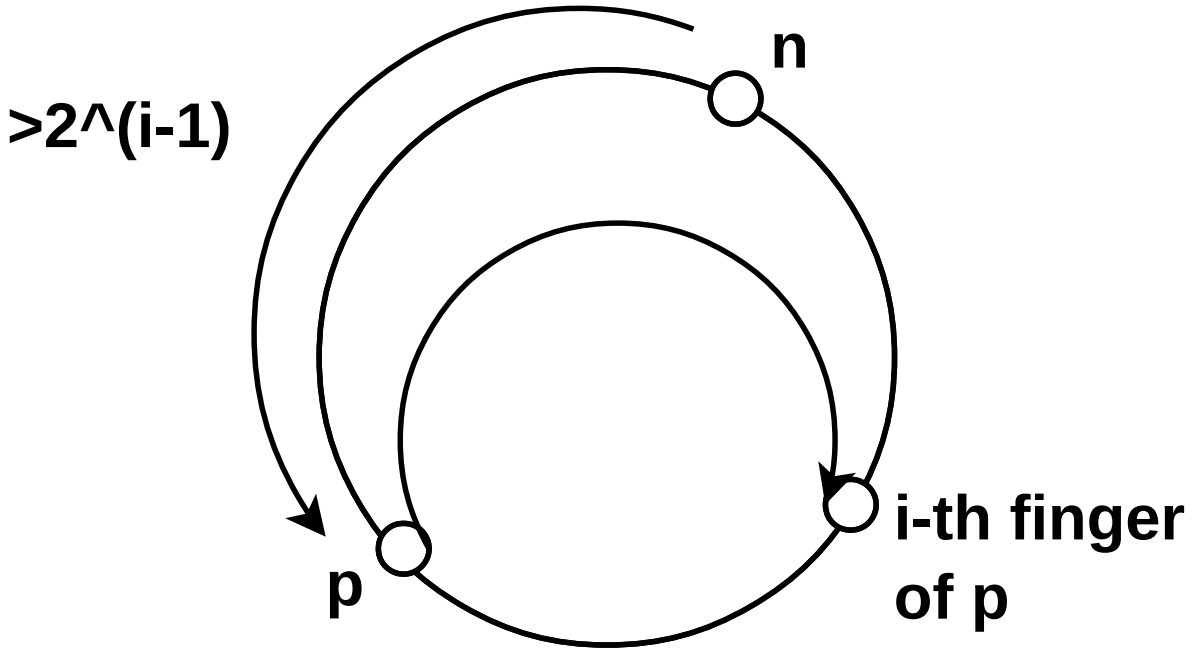
- n.finger[1].node = n.find_successor(1)
- n.finger[1].node >= n.finger[2].start =>

  n.finger[2].node = n.finger[1].node
- Όμοια για το 3:n.finger[3].node=n.finger[2].node



Update existing nodes' finger tables

- Ο νέος κόμβος n είναι το i-οστό finger του κόμβου p αν και μόνο αν:
  - p precedes n τουλάχιστον 2^{i-1}
  - i-οστό finger του p succeeds n



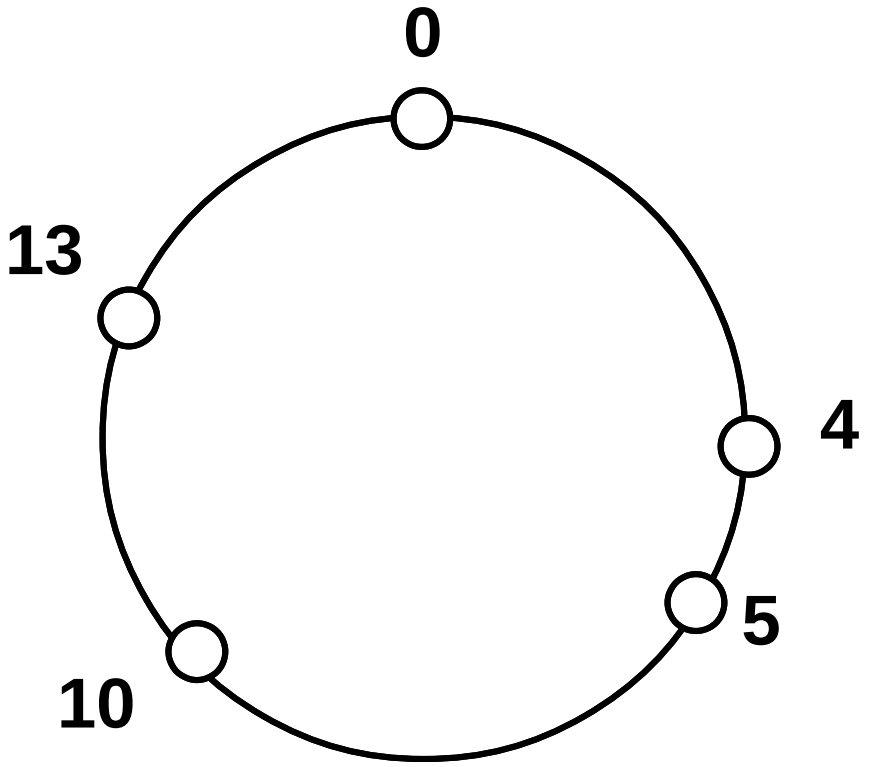
Update existing nodes αλγόριθμος

```
n.update_others(){
for i=1 to m
     p = find_predecessor(n – 2^(i-1))
     p.update_finger_table(n,i)}
```

```
p.update_finger_table(n,i){
if ( n belongs [p , finger[i].node) )
     {finger[i].node = n
      p = predecessor
      p.update_finger_table(n,i)}}
```

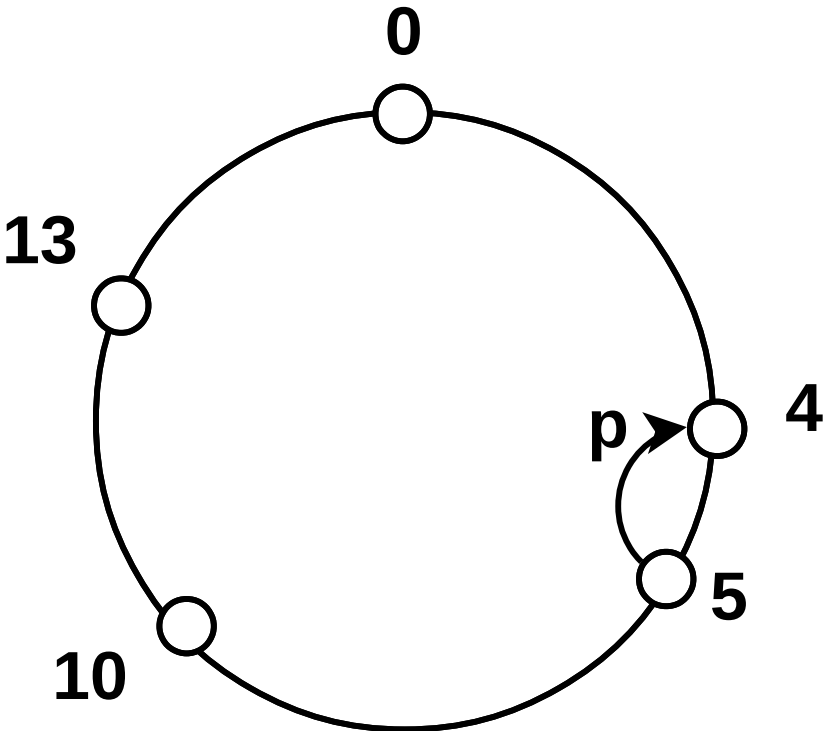
Update existing nodes Παράδειγμα

- Μπαίνει κόμβος 5



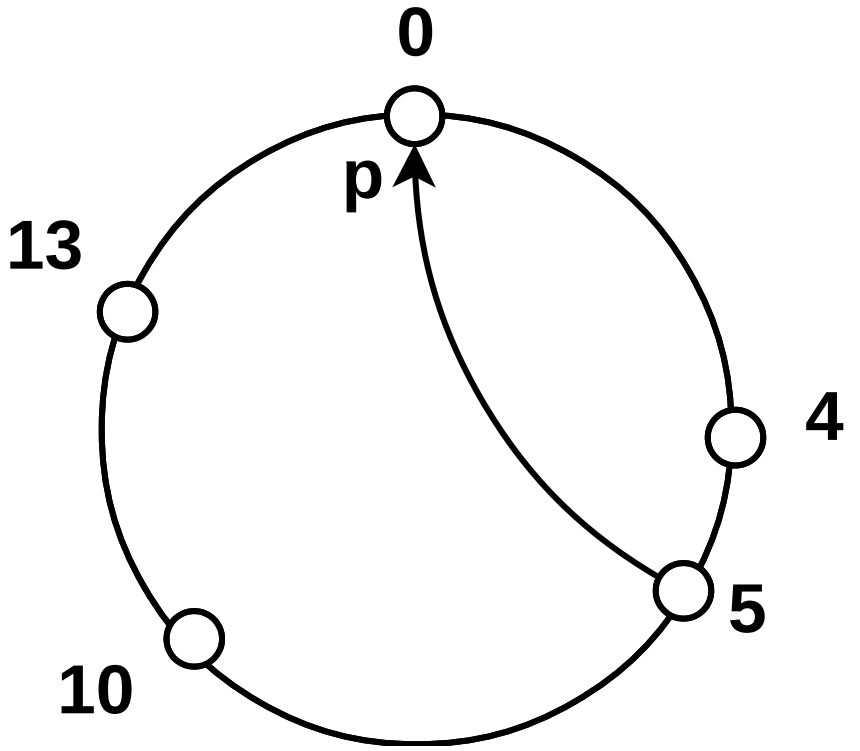
Παράδειγμα(συνέχεια)

- i = 1
- p = find_predecessor(5 – 2^{1-1}) => p = 4
- 5 ανήκει στο [4, 10) άρα
  4.finger[1].node = 5
- p = predecessor(4) = 0



Παράδειγμα(συνέχεια)

- p = 0
- 5 δεν ανήκει στο [0, 4) άρα σταματάμε εδώ για i =1.
- Για i = 2, 3 προκύπτει ότι δεν υπάρχει τίποτα.
- Πάμε να δούμε το i = 4.

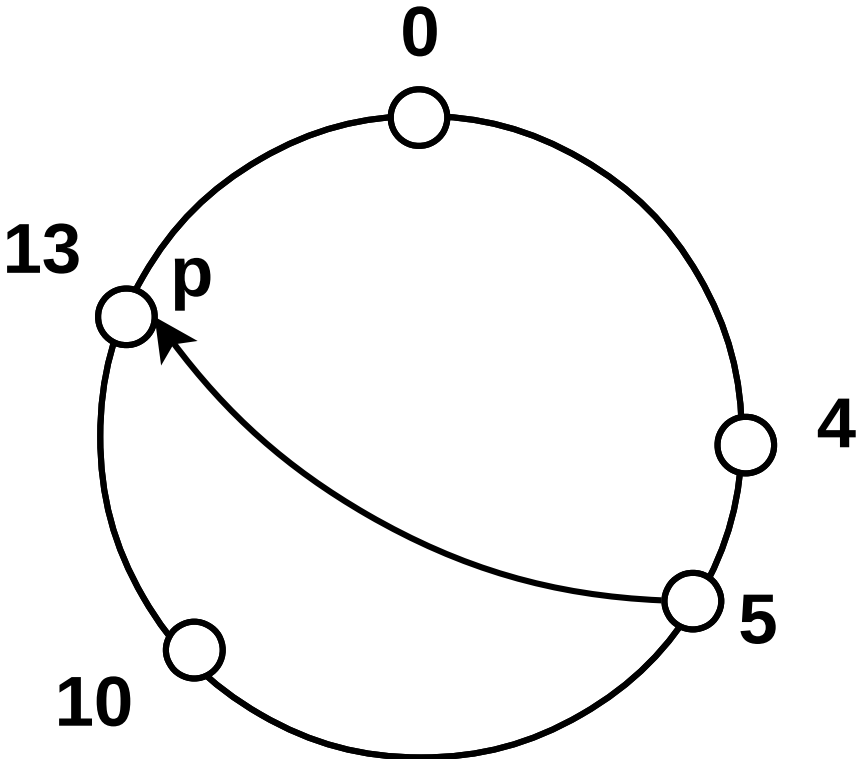


Παράδειγμα(συνέχεια)

- i = 4
- p = find_predecessor(5 – 2^{4-1}) = 13

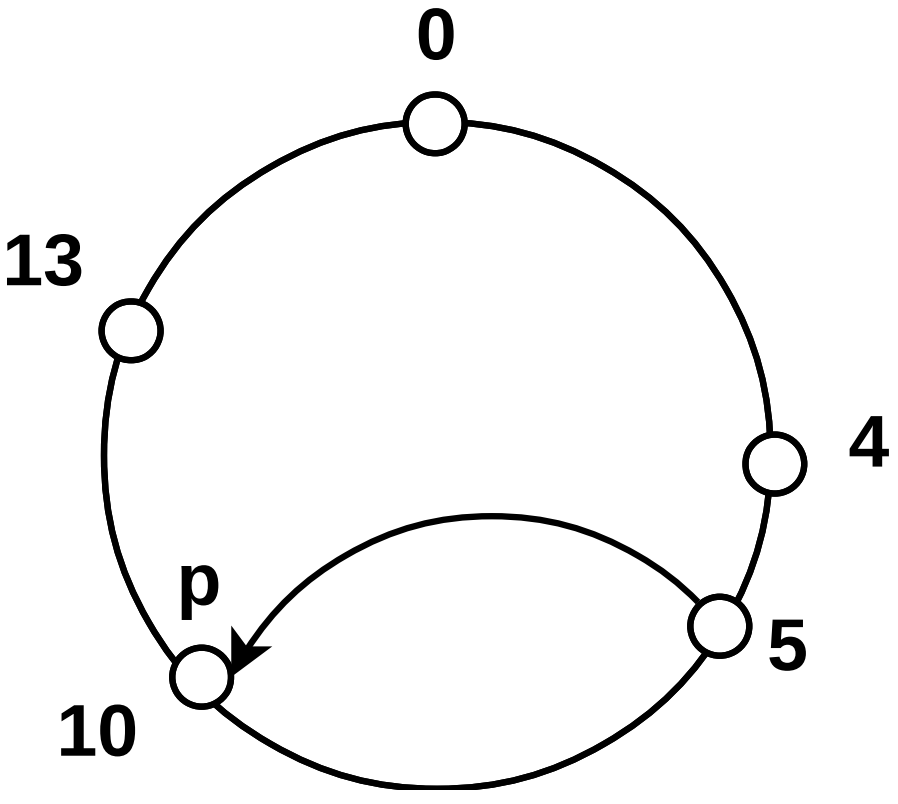
  (πράξεις με modulo 16 για να είμαστε στον κύκλο)
- 5 ανήκει στο [13 , 10) άρα

  13.finger[4].node = 5
- p = predecessor = 10



Παράδειγμα(συνέχεια)

- p = 10
- 5 δεν ανήκει στο [10, 4) άρα σταματάμε εδώ.

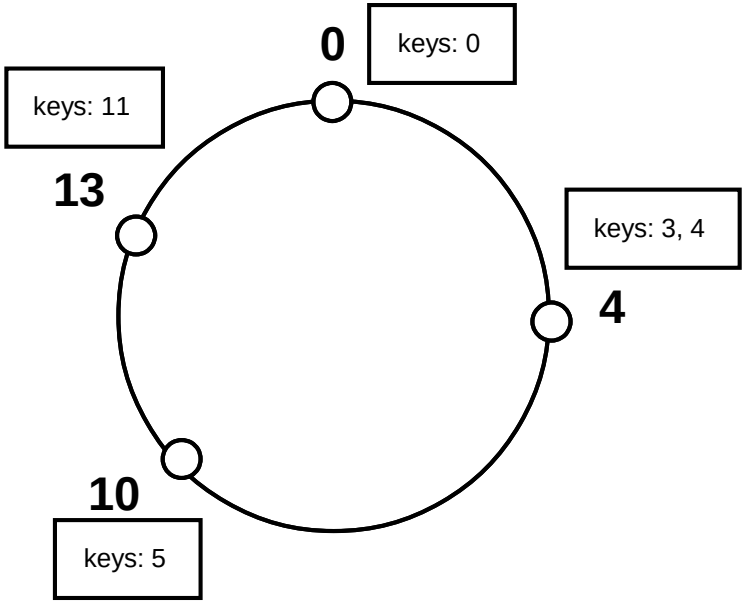


Update: Transferring Keys

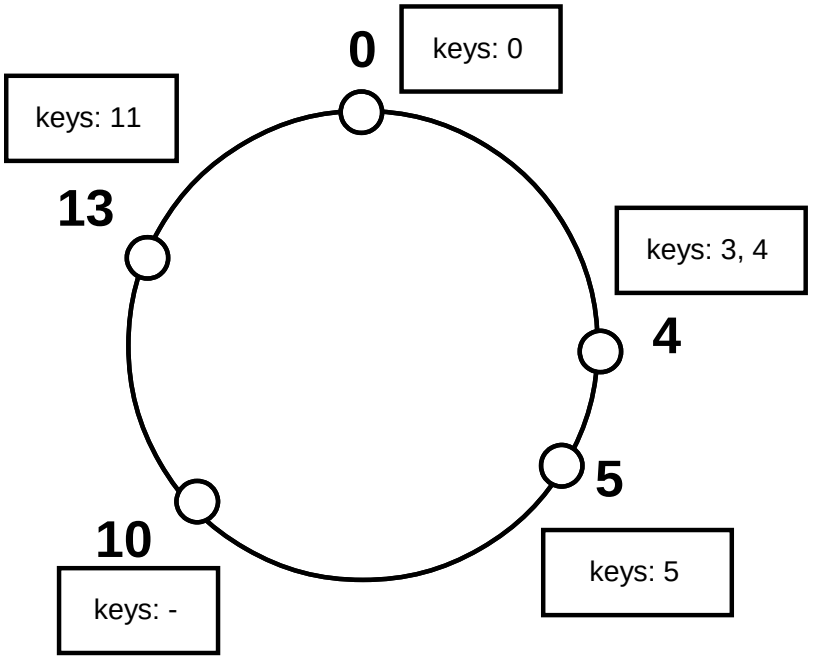
- Υλοποιείται από το υψηλότερου επιπέδου λογισμικό το οποίο χρησιμοποιεί το chord, δηλαδή από τον κόμβο του δικτύου.
- Τα δυνατά προς μετακίνηση κλειδιά βρίσκονται μόνο στον successor του νέου κόμβου, οπότε χρειάζεται να αναφερθεί μόνο σε αυτόν.

Παράδειγμα

Προηγούμενο δίκτυο:



Μπαίνει ο κόμβος 5:



Αρχικά finger tables:

node 0: start	interval	successor
1	[1,2)	4
2	[2,4)	4
4	[4,8)	4
8	[8,0)	10
keys: 0		

node 4: start	interval	successor
5	[5,6)	10
6	[6,8)	10
8	[8,12)	10
12	[12,4)	13
keys: 3, 4		

node 10: start	interval	successor
11	[11,12)	13
12	[12,14)	13
14	[14,2)	0
2	[2,10)	4
keys: 5		

node 13: start	interval	successor
14	[14,15)	0
15	[15,1)	0
1	[1,5)	4
5	[5,13)	10
keys: 11		

Αφού μπει ο κόμβος 5:

node 0: start	interval	successor
1	[1,2)	4
2	[2,4)	4
4	[4,8)	4
8	[8,0)	10
keys: 0		

node 4:start	interval	successor
5	[5,6)	10
6	[6,8)	10
8	[8,12)	10
12	[12,4)	13
keys: 3, 4		

node 13: start	interval	successor
14	[14,15)	0
15	[15,1)	0
1	[1,5)	4
5	[5,13)	5
keys: 11		

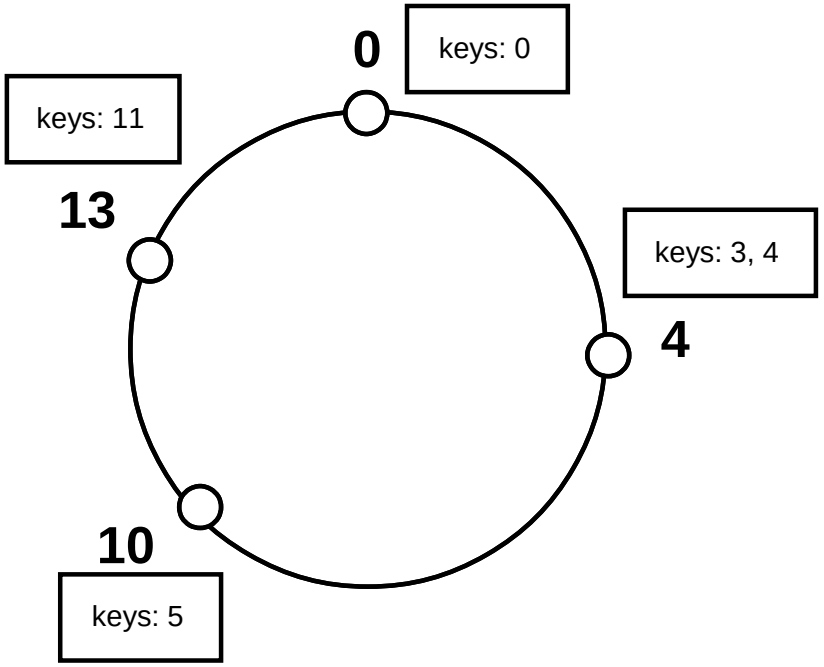
node 10: start	interval	successor
11	[11,12)	13
12	[12,14)	13
14	[14,2)	0
2	[2,10)	4
keys: -		

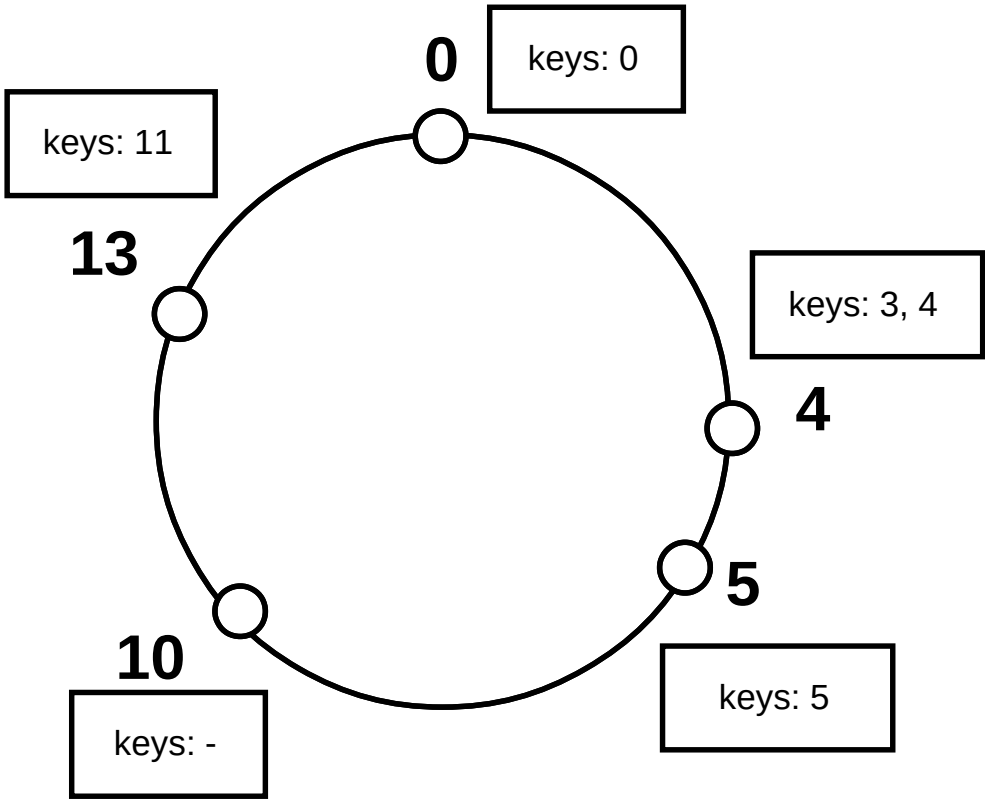
node 5: start	interval	successor
6	[6,7)	10
7	[7,9)	10
9	[9,13)	10
13	[13,5)	13
keys: 5		

Stabilization

- Διαδοχικές αφίξεις / αναχωρήσεις
- Ελάχιστη πληροφορία: σωστά successor pointers
- Basic Stabilization Protocol: Keeps them correct:
  - node n joins
  - n asks n' for info
  - network doesn't get notified of n
  - nodes run *stabilize* every period to get notified

Stabilization πχ.





- κόμβος 5 μπαίνει
- θεωρεί το 10 ως successor
- 10 ενημερώνεται, θέτει predecessor = 5
- 4 τρέχει stabilize, ζητάει 10's predecessor = 5
- Ο 5 ενημερώνεται και παίρνει 4 = predecessor(5)
- fix fingers διορθώνει τα fingers μετά από λίγη ώρα

Node Failures

- Κάποιος κόμβος n βγαίνει από το δίκτυο.
- Ελάχιστη πληροφορία: σωστά successor pointers
- Λύση: κάθε κόμβος έχει successor list με r πρώτα successors.
- Με την πάροδο χρόνου, η stabilize θα διορθώσει τα successor pointers και τα finger tables.

Τέλος

Κουτρουμπούχος Κωνσταντίνος

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών - ΕΜΠ