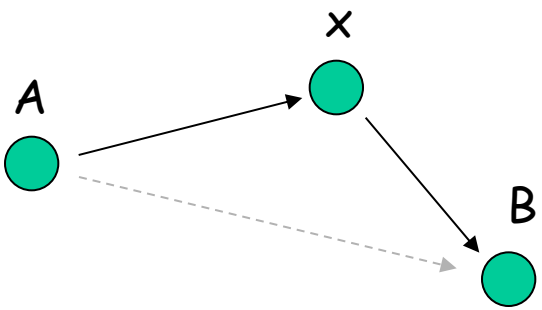
Indirezione

Indirezione: invece di riferire un'entità direttamente, lo si fa ("indirettamente") tramite un'altra entità, che poi accede (o accederà) all'entità originaria

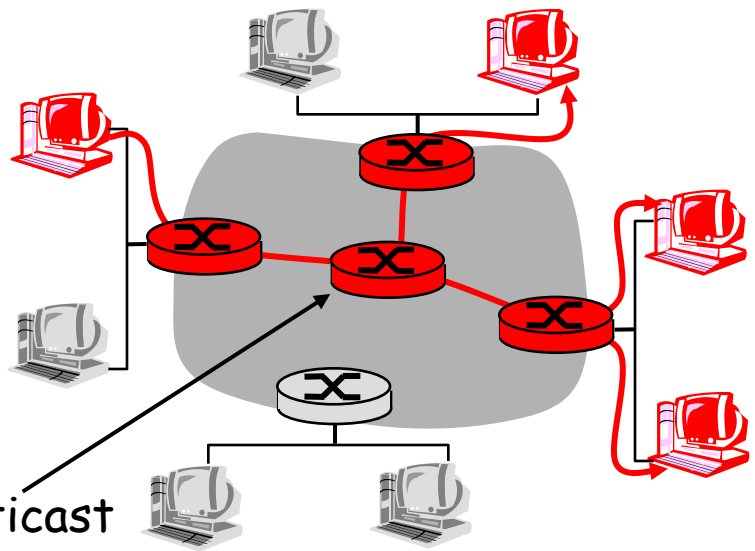


"Every problem in computer science can be solved by adding another level of indirection"
-- Butler Lampson

Multicast: un mittente, molti ricevitori

Multicast: processo di invio di un datagramma destinato a molteplici ricevitori con una singola operazione di trasmissione

Domanda: come realizzare il multicast?

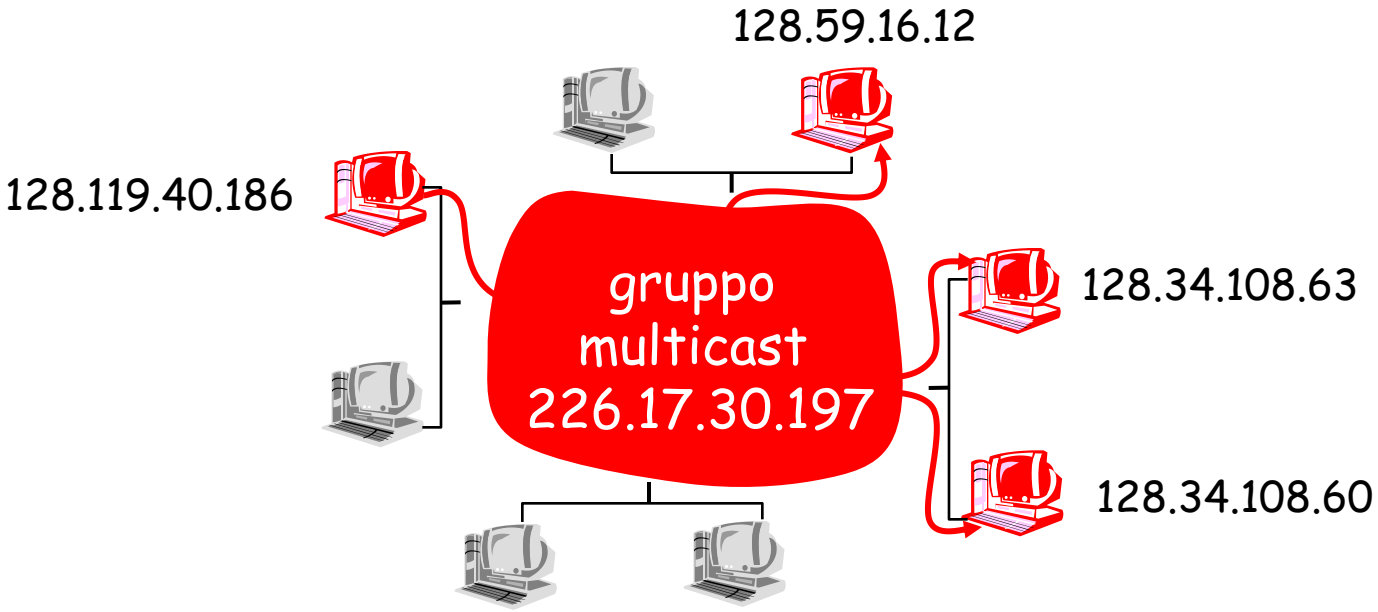


Multicast router (rossi) duplicano e inoltrano i datagrammi multicast

Network multicast

- i router partecipano attivamente al multicast, facendo all'occorrenza copie dei pacchetti e inoltrandoli verso i ricevitori

Internet Multicast – Modello di Servizio



concetto di gruppo multicast: uso della indirezione

- i mittenti indirizzano datagrammi IP verso gruppo multicast
- i router inoltrano i datagrammi multicast verso gli host che si sono "iscritti" a quel gruppo multicast

Multicast attraverso Indirezione: perchè?

Mobilità e indirezione

Come contatti un amico che si sposta?

Considera un amico (senza cellulare!) che viaggia in continuazione, come fai a rintracciarlo?

- chiami i suoi parenti?
- aspetti che sia lui a dirti dove si trova?
- lasci un messaggio da qualche parte, sperando che prima o poi lo legga?

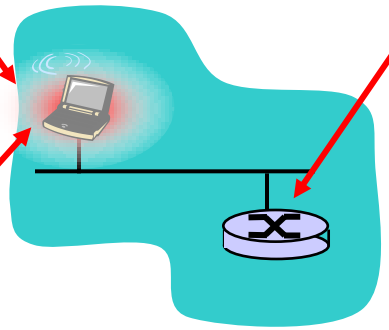
Mobilità e indirezione:

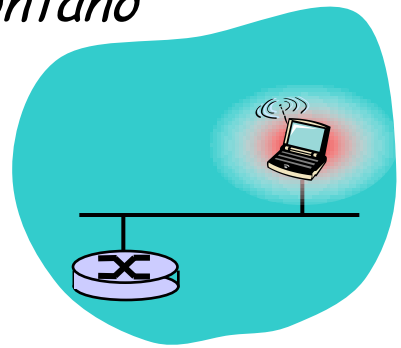
- nodi mobili che si spostano da una rete all'altra
- Un "correspondent" vuole mandare pacchetti al nodo mobile
- due approcci:
  - *routing indiretto:* il flusso dati dal correspondent al nodo mobile avviene attraverso un "home agent"
  - *routing diretto:* il correspondent ottiene il "foreign address" del nodo mobile, poi invia dati direttamente al nodo mobile

Mobilità: terminologia

home network: rete di "casa" permanente del nodo mobile (e.g., 128.119.40/24)

home agent: *entità che svolge le funzioni di mobilità per conto del nodo mobile, quando questo si trova lontano*





wide area network

correspondent

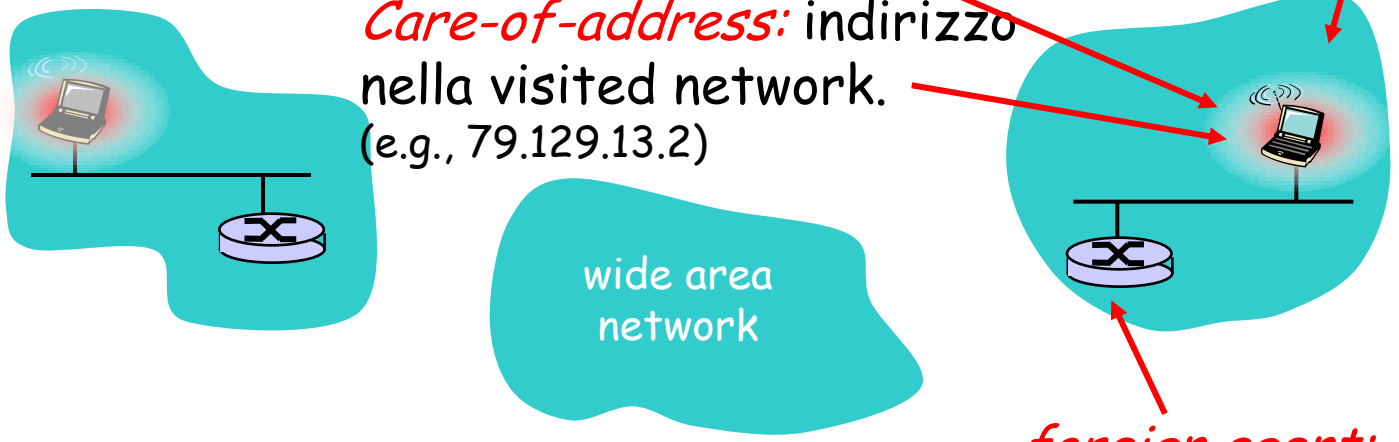
Permanent address: indirizzo nella home network, che può *sempre* essere usato per raggiungere il nodo mobile e.g., 128.119.40.186

Mobilità: terminologia (cont.)

Permanent address: rimane fisso (e.g., 128.119.40.186)

visited network: network in cui il nodo mobile risiede attualmente (e.g., 79.129.13/24)

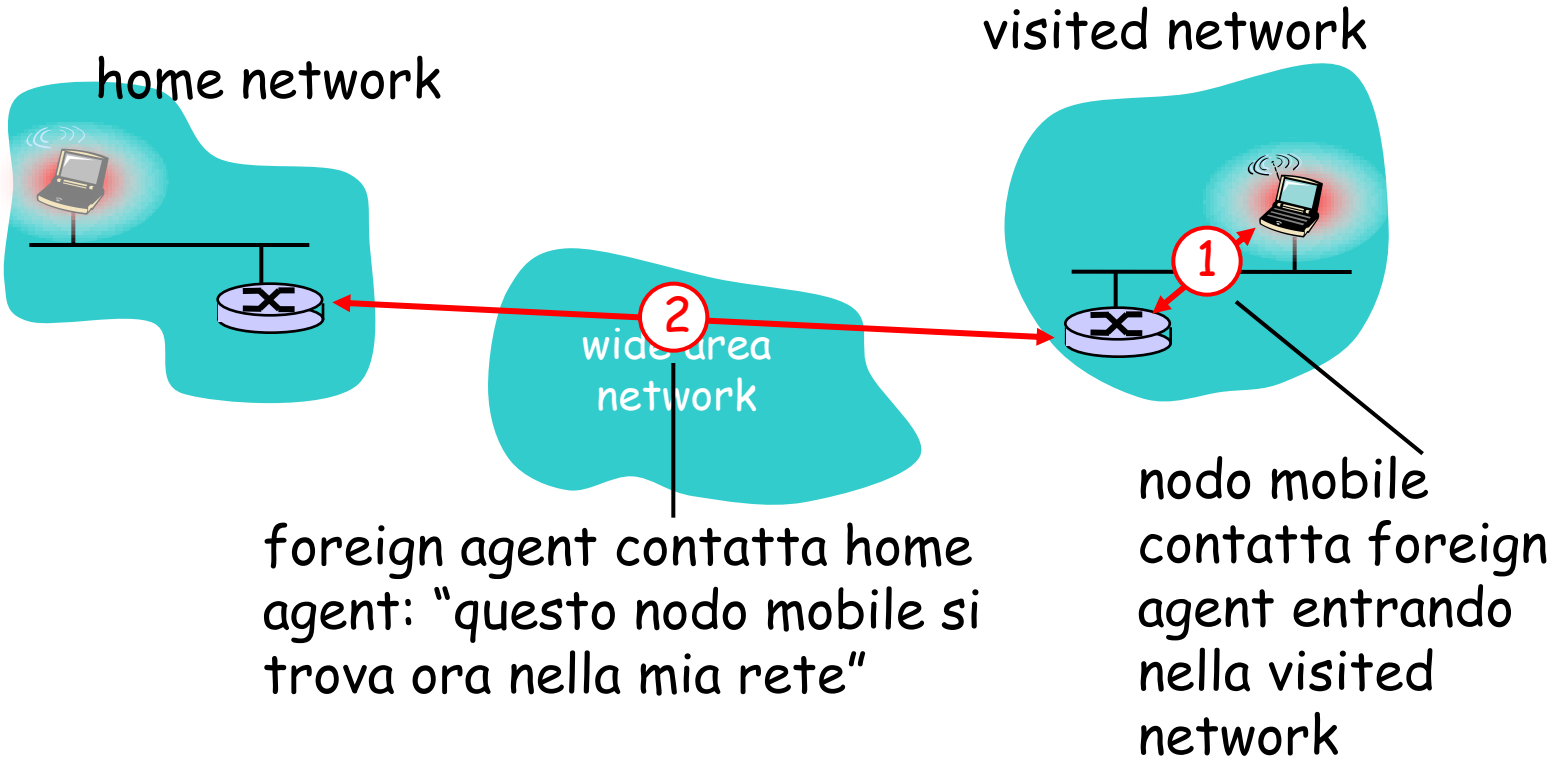
Care-of-address: indirizzo nella visited network. (e.g., 79.129.13.2)



correspondent: *vuole comunicare col nodo mobile*

foreign agent: *entità nella "visited network" che svolge le funzioni di mobilità per conto del nodo mobile.*

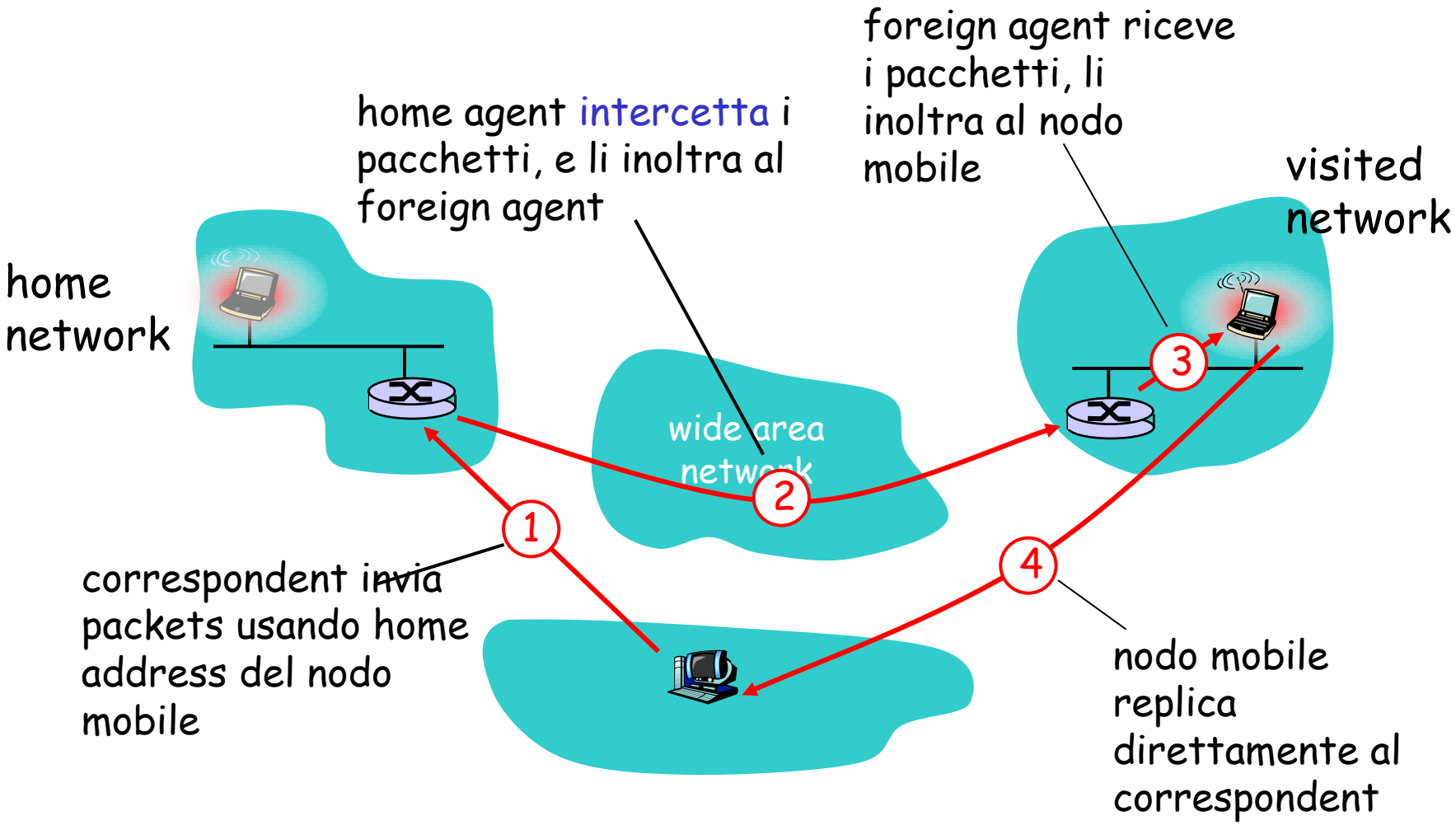
Mobilità: registrazione



Risultato:

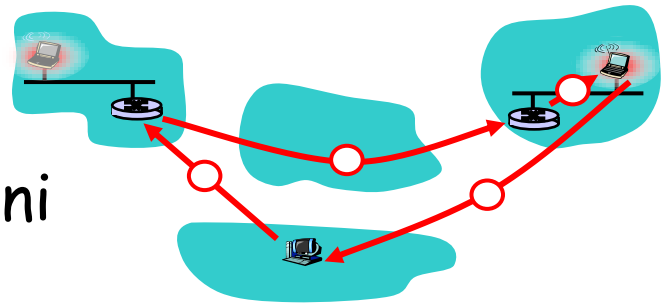
- foreign agent conosce il nodo mobile
- home agent concosce la posizione del nodo mobile

Mobilità con Routing Indiretto



Routing Indiretto: osservazioni

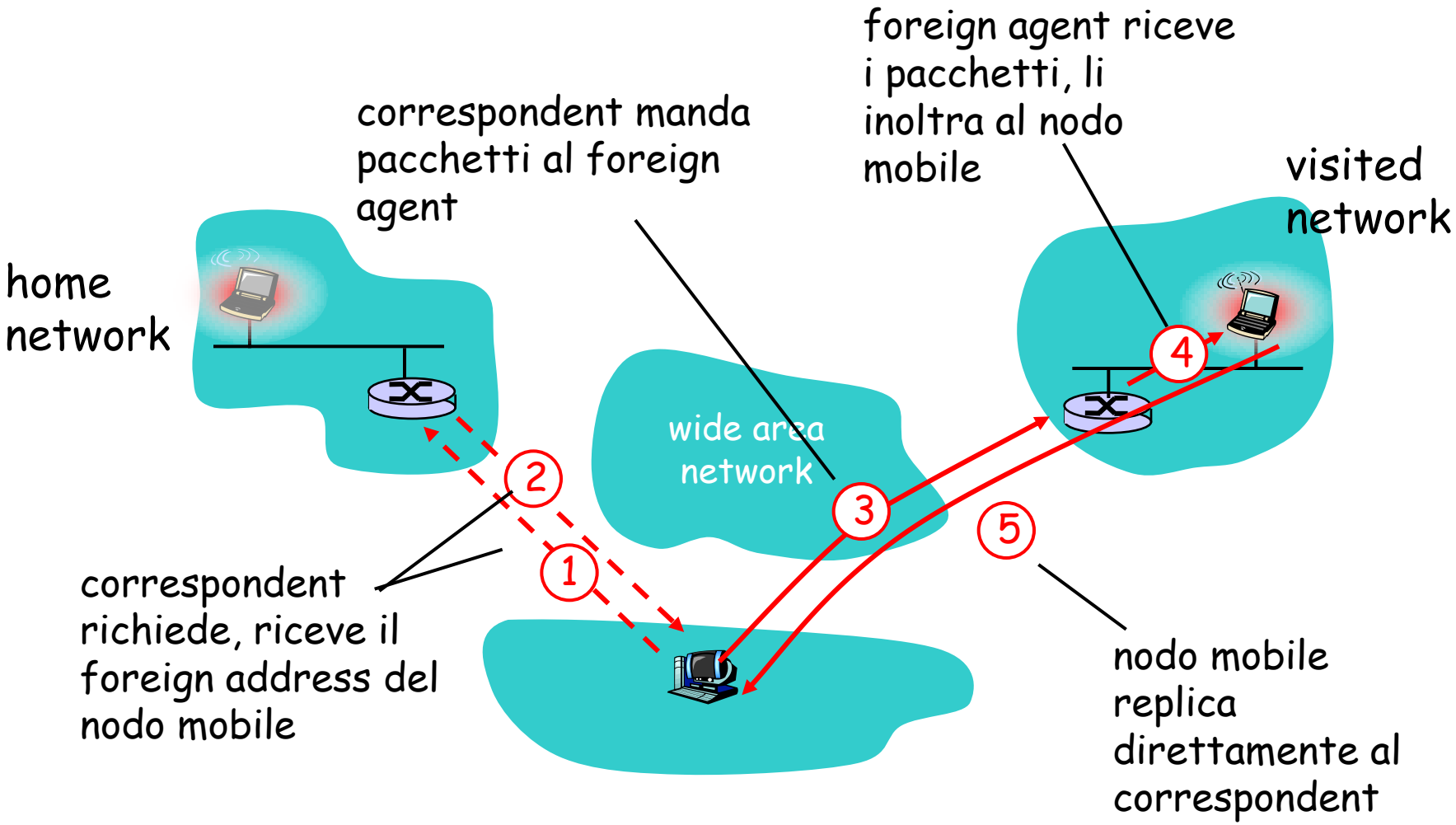
- nodo mobile usa due indirizzi:
  - permanent address: usato dal correspondent (dunque la posizione del nodo mobile è *transparent* per il correspondent)
  - care-of-address: usato dal home agent per inoltrare datagrammi al nodo mobile
- le funzioni del foreign agent potrebbero essere svolte dal nodo stesso
- routing a triangolo: correspondent-home-network-mobile
  - inefficiente quando corr, nodo mobile sono vicini



Routing Indiretto: spostamento tra reti

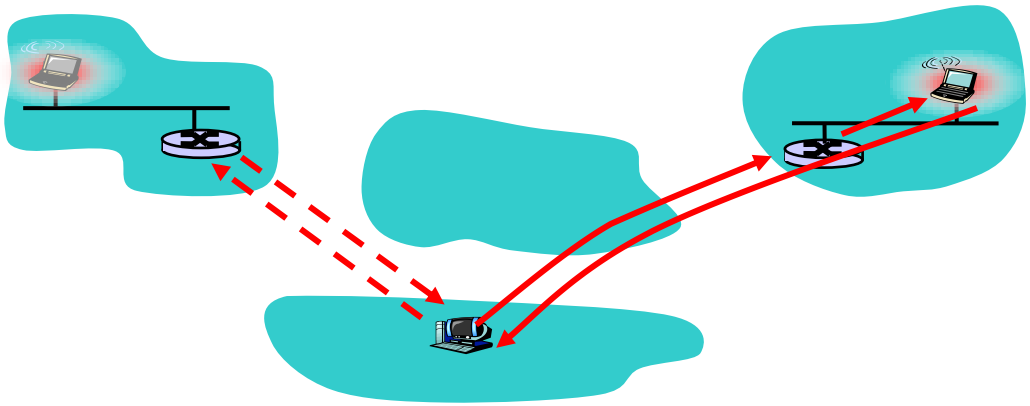
- supponiamo che un utente mobile si sposti in un'altra rete
  - si registra col nuovo foreign agent
  - il nuovo foreign agent si registra col home agent
  - home agent aggiorna care-of-address del nodo mobile
  - i pacchetti continuano a essere inoltrati al nodo mobile (ma col nuovo care-of-address)
- la mobilità, il cambiamento di foreign networks sono transparent per il mittente: *le connessioni in corso possono essere mantenute!*

Mobilità con Routing Diretto



Mobilità con Routing diretto: osservazioni

- evita il problema del routing a triangolo
- non transparent per il correspondent: correspondent deve ottenere il care-of-address dal home agent
  - cosa succede se il nodo mobile cambia rete?



IP Mobile

- RFC 3220
- ha parecchie delle caratteristiche che abbiamo visto:
  - home agents, foreign agents, registrazione col foreign-agent, care-of-addresses, incapsulamento (pacchetto dentro il pacchetto)
- tre componenti:
  - scoperta degli agenti
  - registrazione con home agent
  - routing indiretto dei datagrammi

Mobilità con indirezione: perchè indirezione?

- transparency per il correspondent
- "quasi" transparent per il nodo mobile (eccetto per il fatto che nodo mobile deve registrarsi col foreign agent)
  - transparent per router, resto della infrastruttura
  - possibili problemi se è attivo il filtraggio in uscita ("egress filtering") nella foreign network (nel caso che indirizzo IP sorgente del nodo mobile rimanga il suo home address): spoofing?

Costruzione di una Infrastruttura di Indirezione in Internet

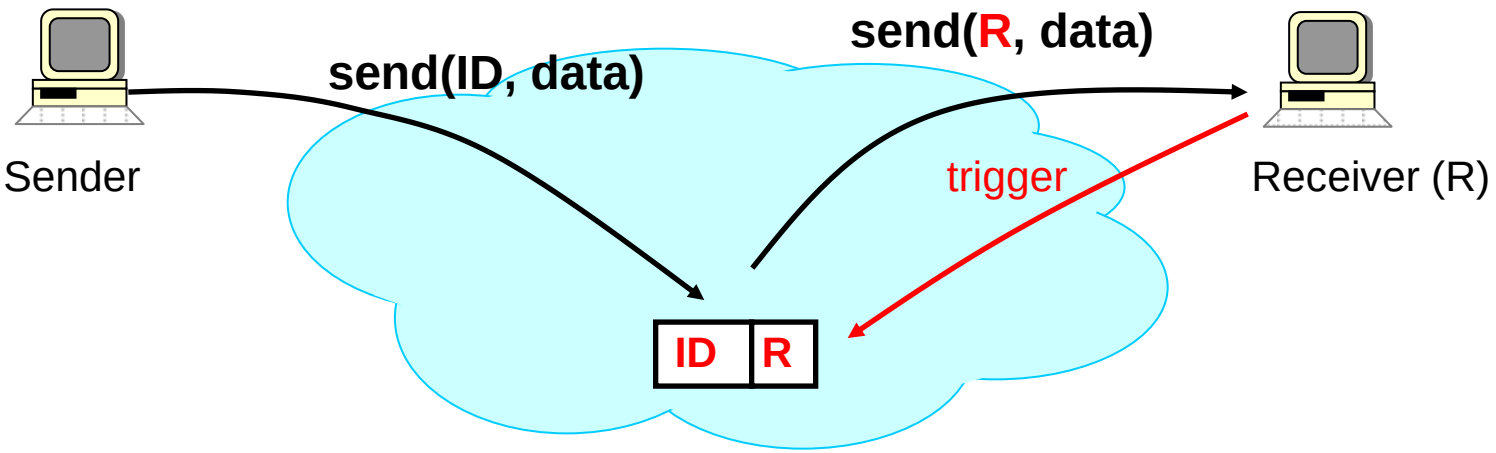
Motivazione:

- Internet è stato inizialmento concepito attorno al paradigma di comunicazione punto-punto:
  - invio di pacchetto "p" da host "A" a host "B"
  - un mittente, un ricevitore, in posizioni fisse e ben note
- … non appropriato per applicazioni che richiedono altre primitive di comunicazione:
  - multicast (uno a molti)
  - mobilità (uno a "uno in qualche altro posto")
  - anycast (uno a "uno tra molti")
- abbiamo visto come usare la indirezione per fornire questi servizi
  - idea: fare della indirezione un "servizio di base" (un oggetto di prima classe)

Infrastruttura di indirezione di Internet (i3)

I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. "Internet indirection infrastructure," IEEE/ACM Trans. Netw. 12(2), pp. 205-218, 2004.

- cambiamento nel paradigma di comunicazione: invece di punto-punto, si comunica attraverso ID del contenuto
  - ogni pacchetto ha un identificativo ID
  - per ricevere pacchetto con identificativo ID, il ricevente R installa un trigger (ID, R) nella rete
  - triggers mentenuto in nodi della rete di "overlay"



Modello di servizio

- API
  - sendPacket(p);
  - insertTrigger(t);
  - removeTrigger(t); // optional

Nota: uso di soft-state ☺. Perchè?

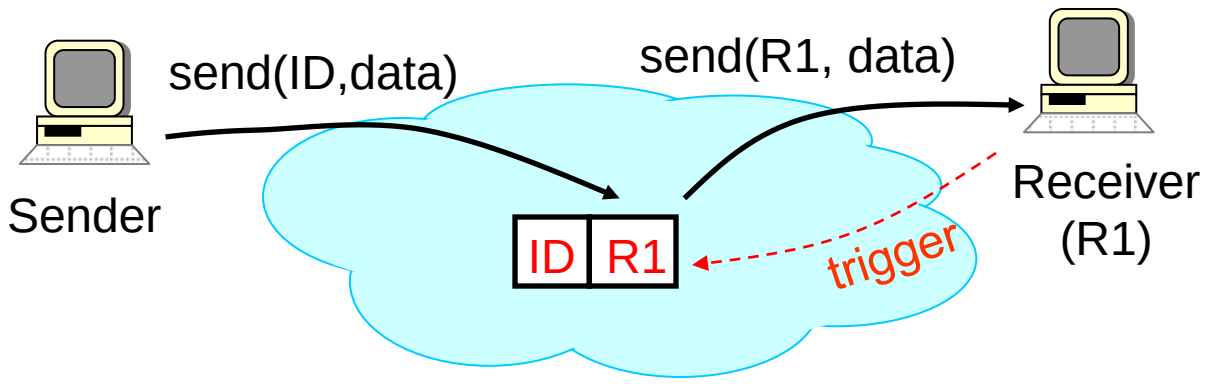
- modello sottostante best-effort (come IP)
- trigger rinfrescati periodicamente dagli hosts
- affidabilità, controllo di congestione, controllo di flusso implementati dai sistemi terminali e dai nodi di overlay che mantengono i triggers

Discussione

- trigger simile ad una entry di una tabella di routing
- essenzialmente un sistema di tipo publish-subscribe a livello applicazione
- infrastruttura di overlay a livello applicazione
- diversamente da IP, i sistemi terminali controllano i triggers, ovvero, sono responsabili per il settaggio e il mantenimento delle "tabelle di routing"
- fornisce supporto a
  - mobilità
  - multicast
  - anycast
  - servizi componibili

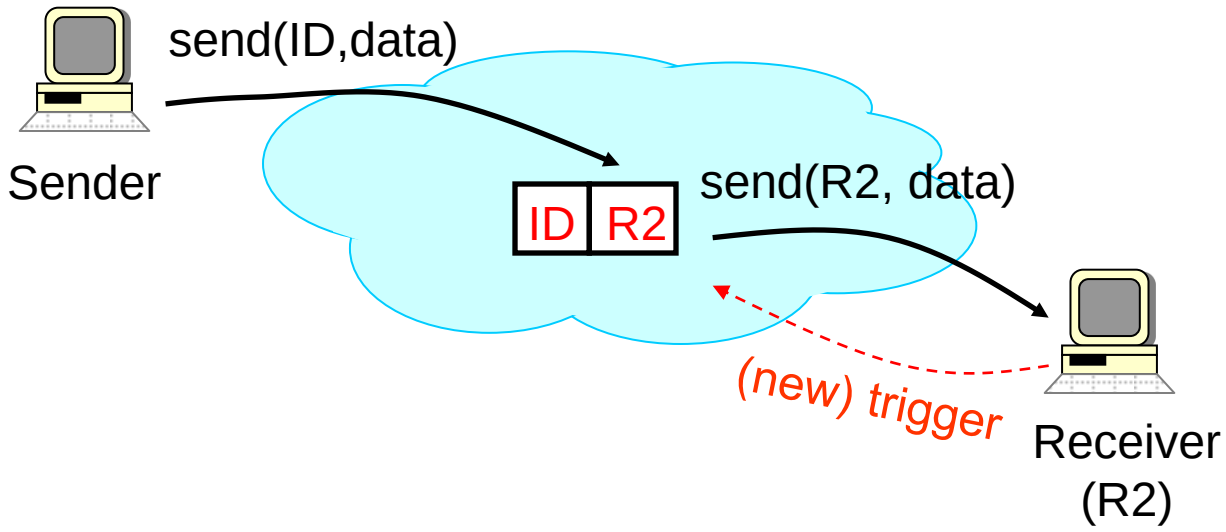
Mobilità

- ricevitore aggiorna il suo trigger quando si muove da una rete ad un altra
  - mobilità transparent per il mittente
  - privacy sulla posizione



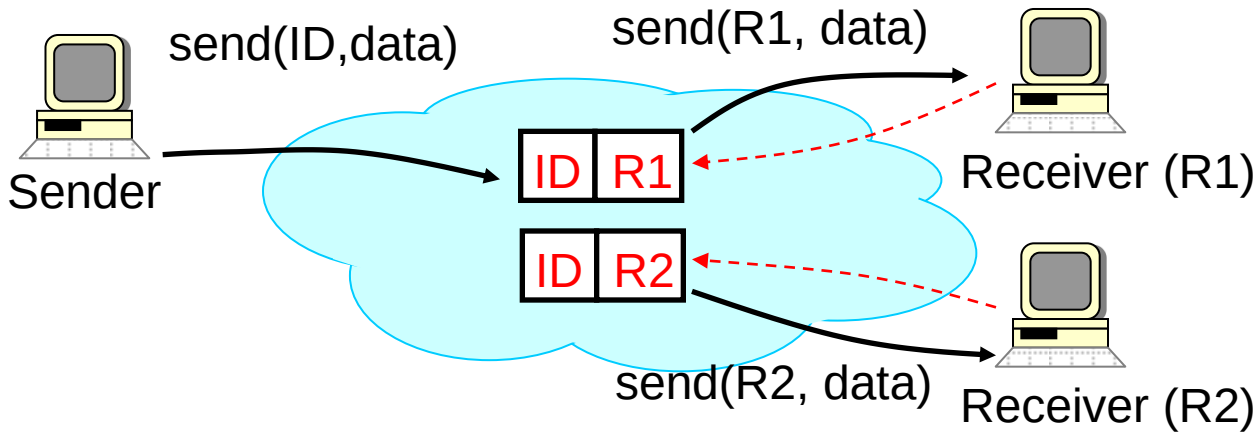
Mobilità

- ricevitore aggiorna il suo trigger quando si muove da una rete ad un altra
  - mobilità transparent per il mittente
  - privacy sulla posizione



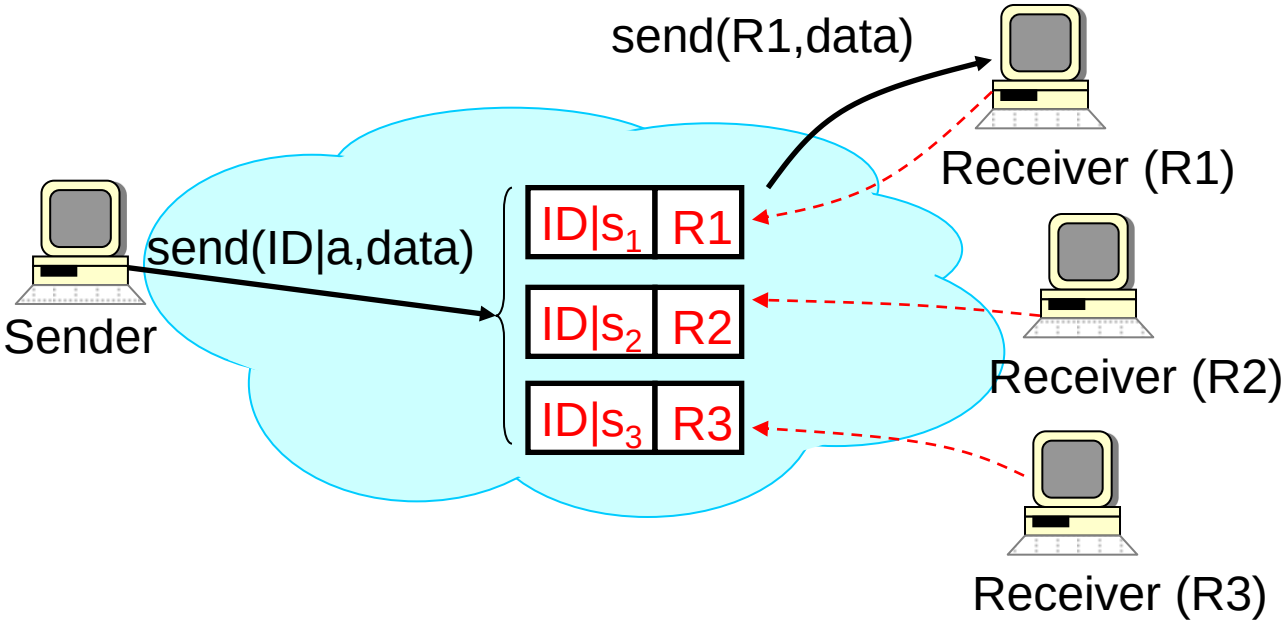
Multicast

- unifica i paradigmi multicast e unicast
  - multicast: ricevitori inseriscono triggers col medesimo ID
- applicazione passa in modo naturale da multicast a unicast, quando occorre
  - "impossibile" nel modello IP attuale



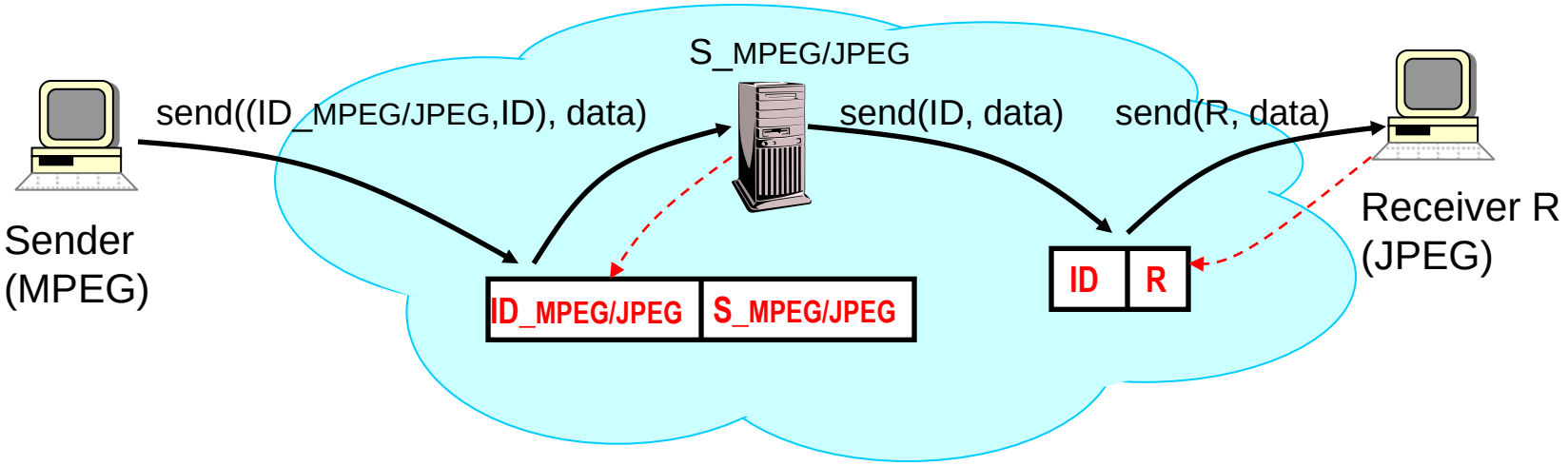
Anycast (cont.)

- comunicazione con "uno qualunque di un insieme di nodi"
- ricevitori Ri in gruppo anycast inseriscono triggers con stesso ID (e qualificatori anycast)



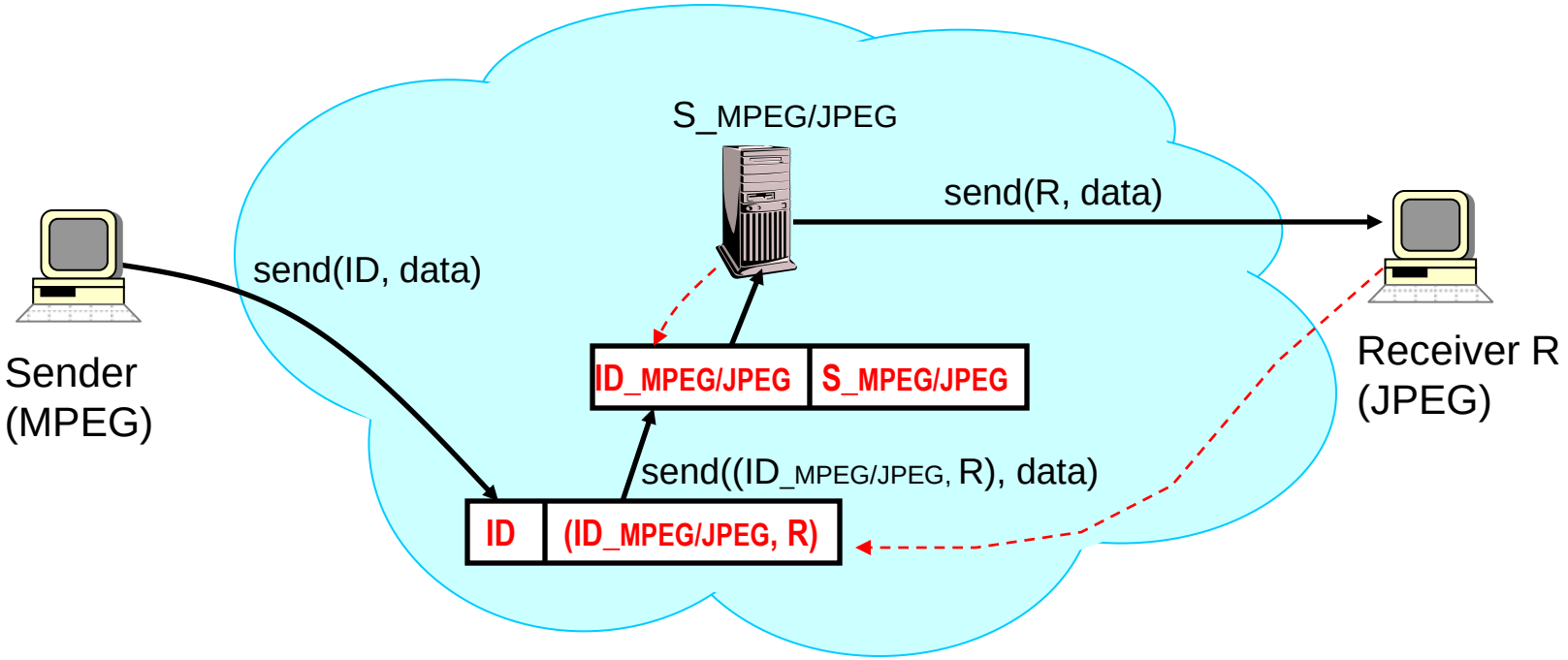
Servizi componibili

- si usa uno stack di ID per codificare operazioni successive da svolgere sui dati (es: transcodifica)
- non c'è bisogno di configurare percorsi tra i servizi



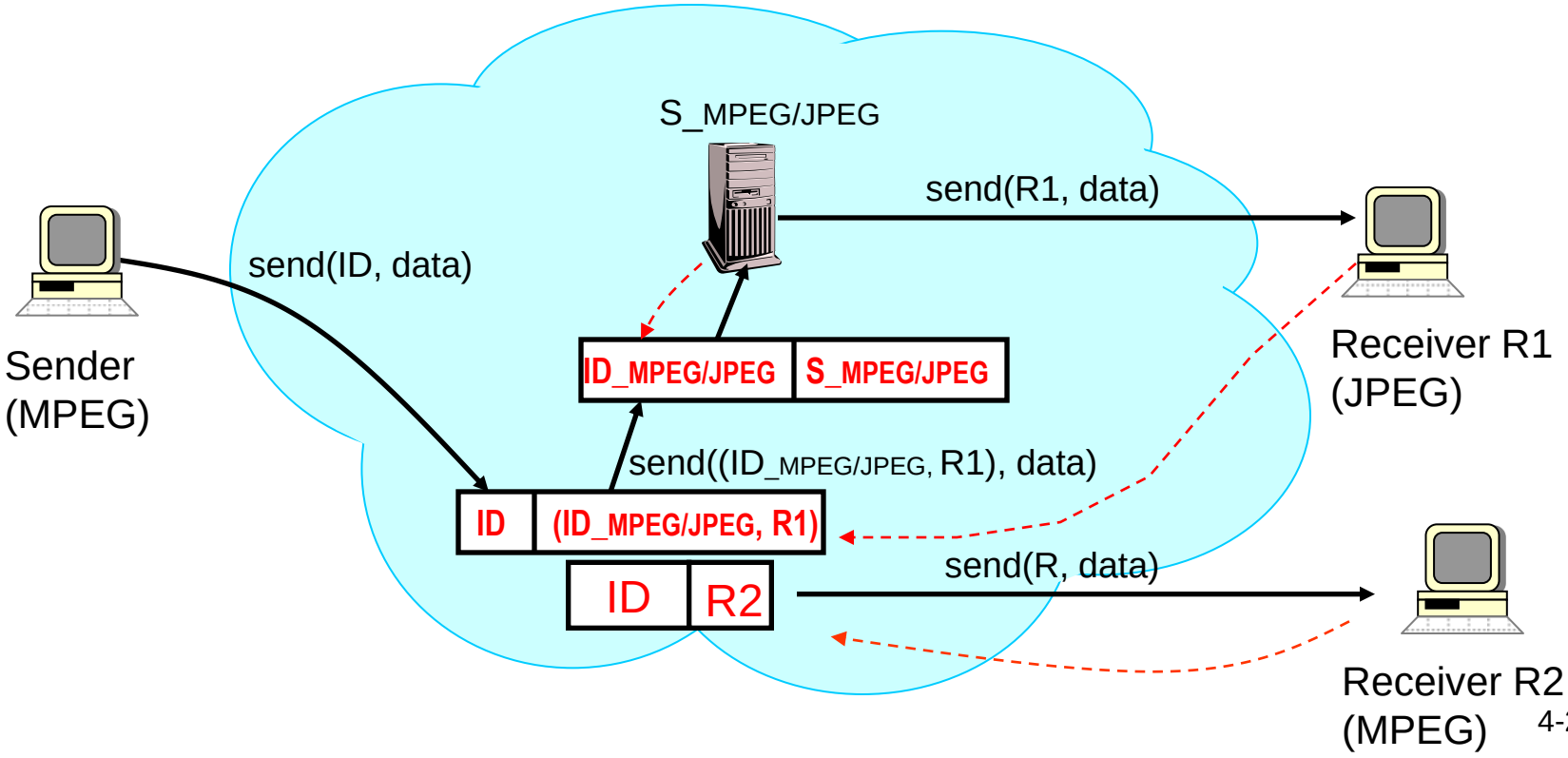
Servizi componibili (cont.)

- sia i ricevitori sia i mittenti posso specificare operazioni da eseguire sui dati



Multicast eterogeneo

- sia i ricevitori sia i mittenti possono specificare operazioni da eseguire sui dati



Indirezione: discussione

Abbiamo visto il concetto di indirezione applicato in vari contesti:

- multicast
- mobilità
- proposta I3

Vantaggi della indirezione:

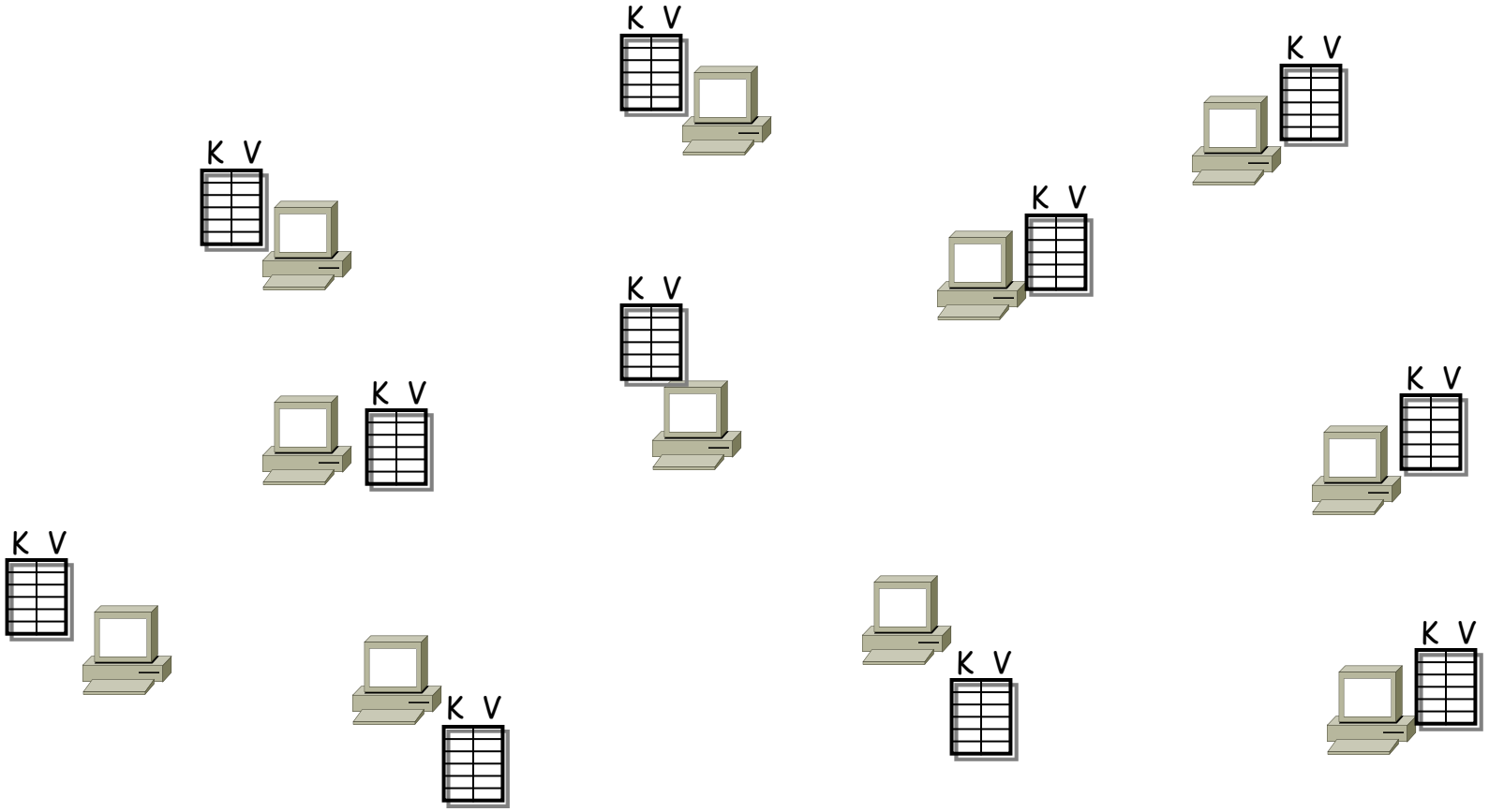
- mittente non deve conoscere il ricevitore/i, ovvero non si vuole che mittenti conoscano le identità coinvolte
- soluzione elegante
- "transparency" della indirezione è fondamentale
- prestazioni? è più efficiente?
- sicurezza: questioni importanti per I3

Implementazione di insert/retrieve: hash table distribuite (DHT)

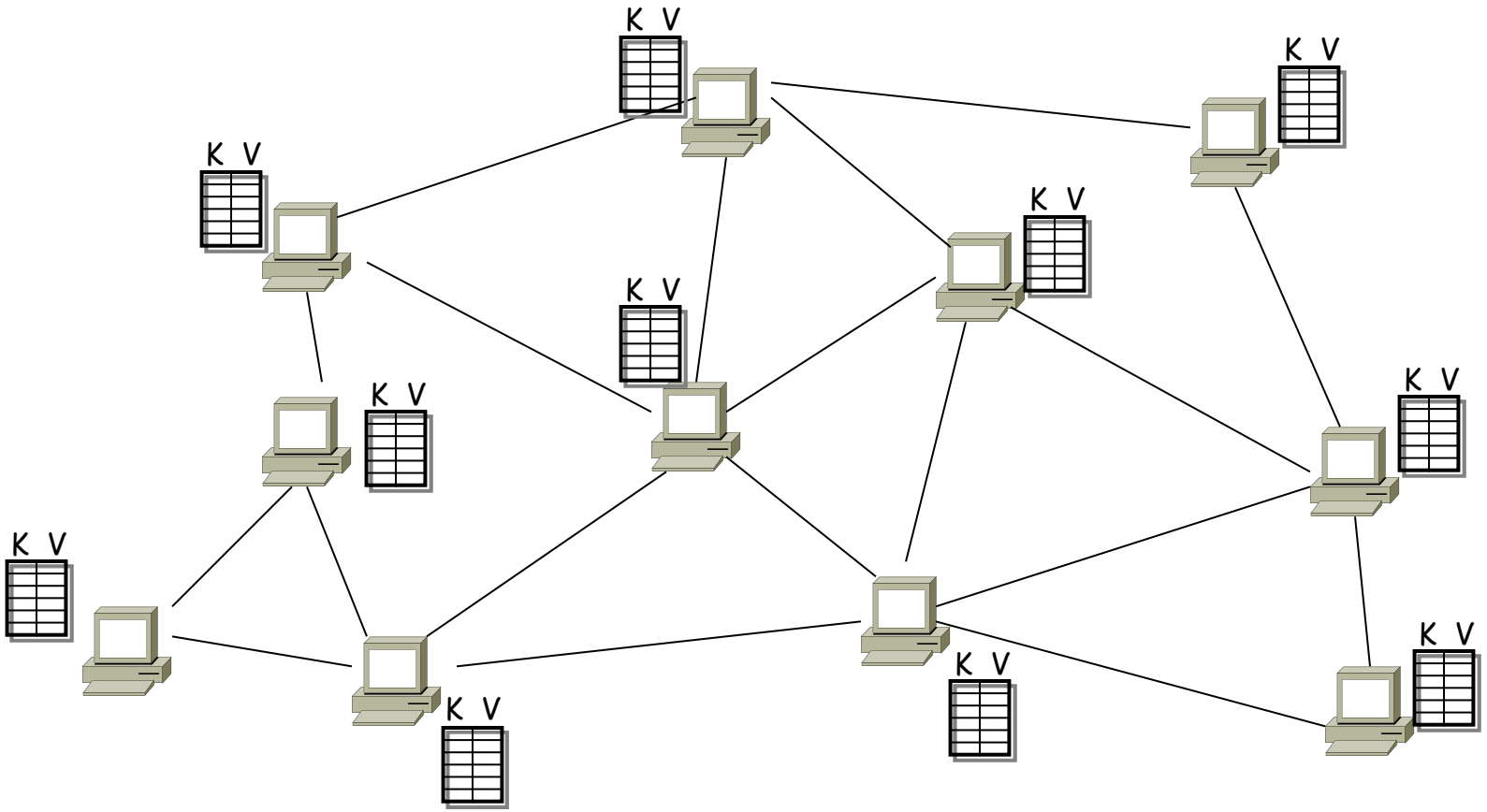
- hash table
  - struttura dati che mappa "chiavi" in "valori"
  - componente essenziale di molti sistemi software distribuiti: Distributed Hash Table (DHT)
  - analogamente nelle reti, ma sparsa su Internet
- interfaccia: come in I3
  - insert(key, value)
  - lookup(key)
- DHT: overlay di nodi (operante su Internet)
  - ogni nodo DHT supporta una singola operazione:

  "data una chiave in ingresso, inoltrare un messaggio verso il nodo che gestisce quella chiave"

DHT in azione

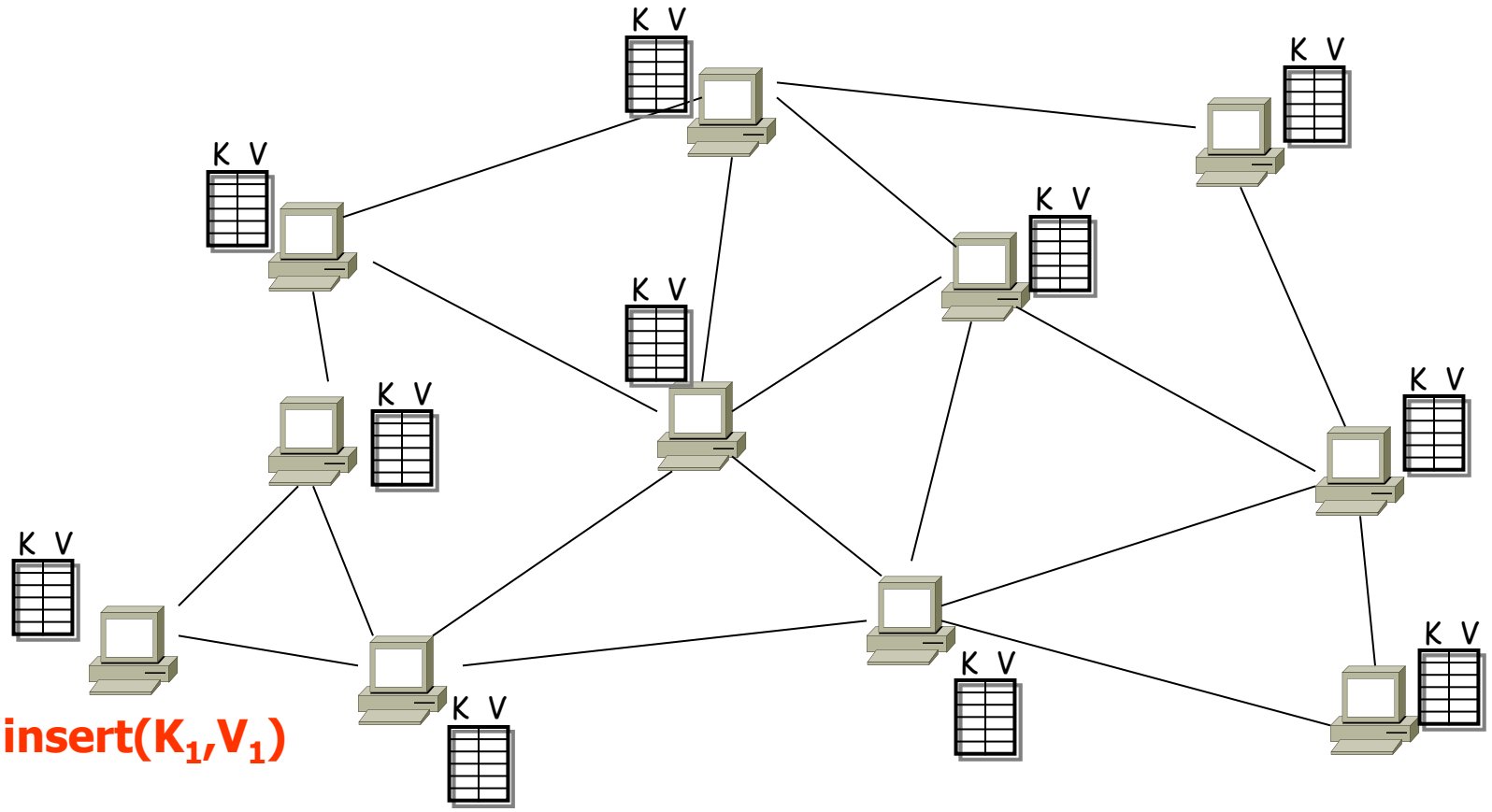


DHT in azione



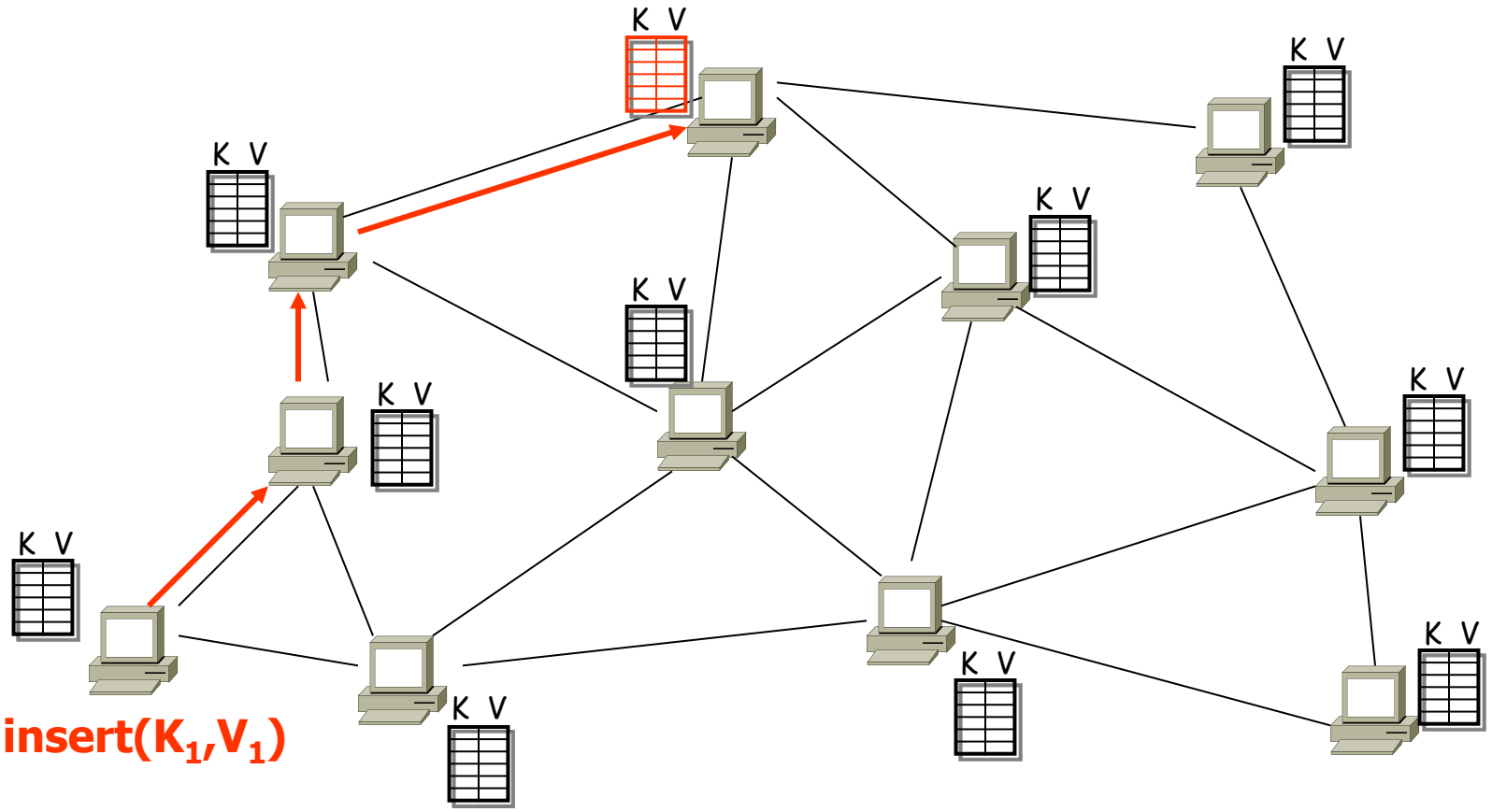
Operazione: data una *chiave* in ingresso; instrada messaggio al nodo che gestisce la *chiave*

DHT in azione: insert()



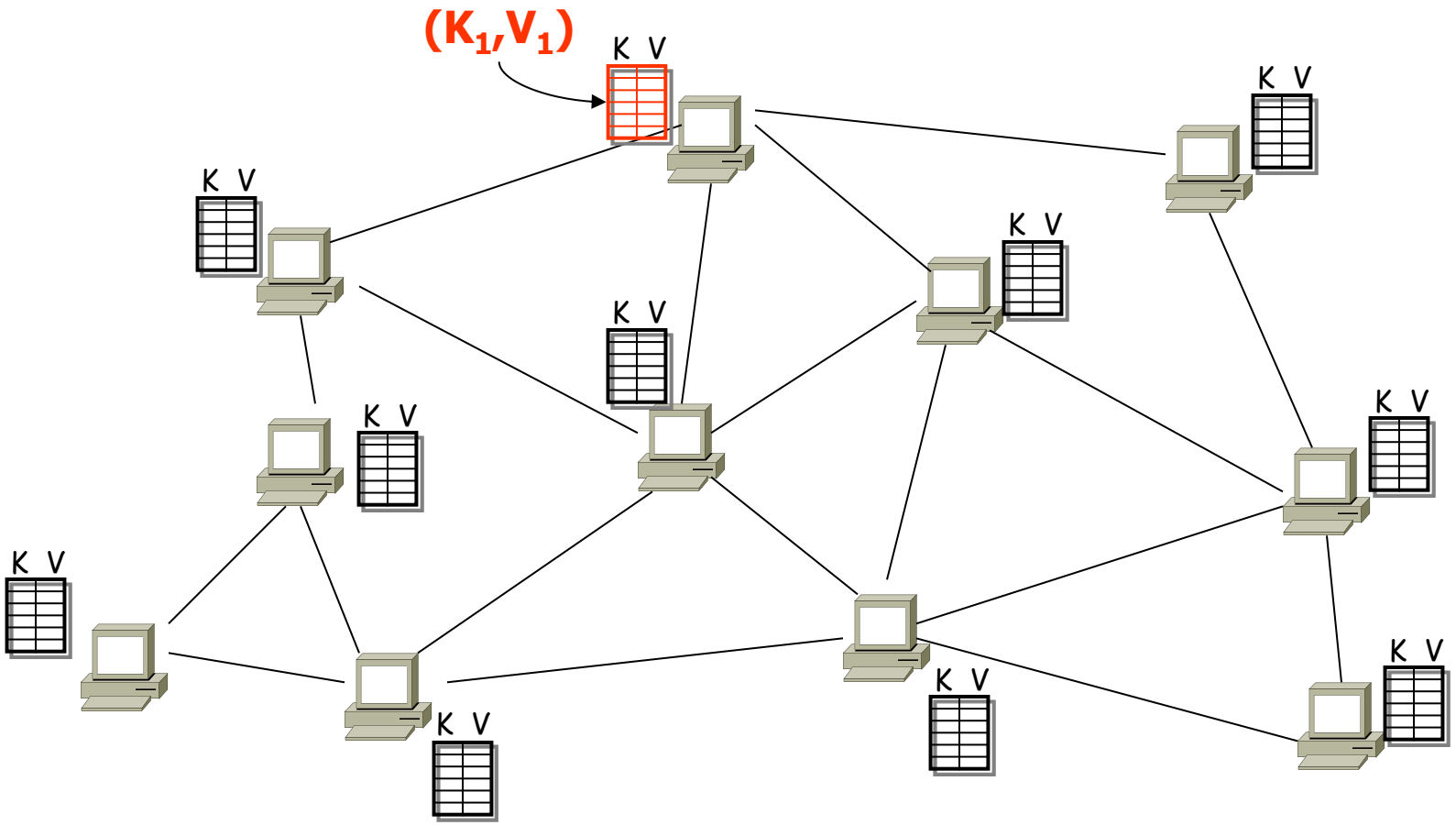
Operazione: data una *chiave* in ingresso; instrada messaggio al nodo che gestisce la *chiave*

DHT in azione: insert()



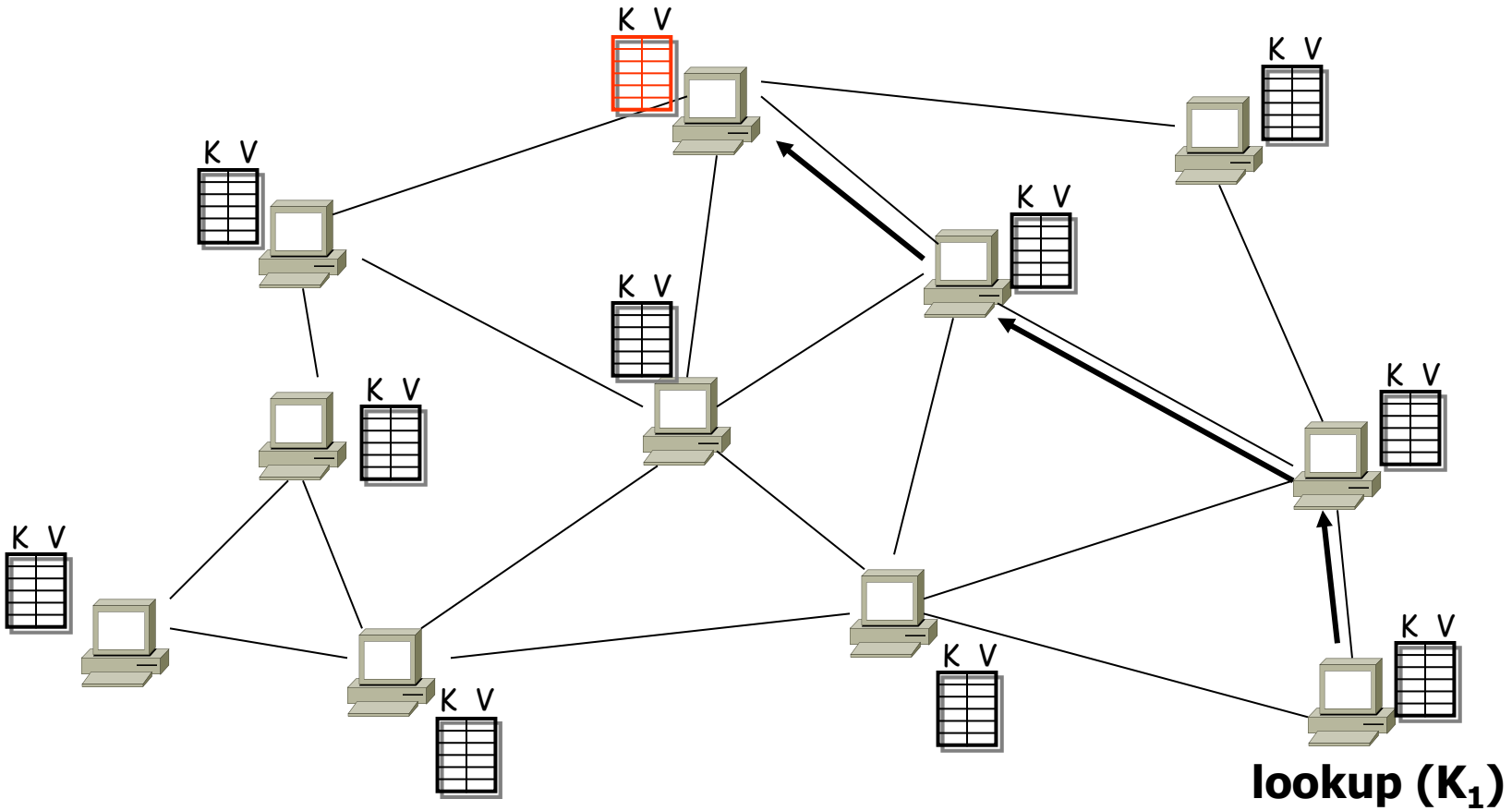
Operazione: data una *chiave* in ingresso; instrada messaggio al nodo che gestisce la *chiave*

DHT in azione: insert()



Operazione: data una *chiave* in ingresso; instrada messaggio al nodo che gestisce la *chiave*

DHT in azione: lookup()



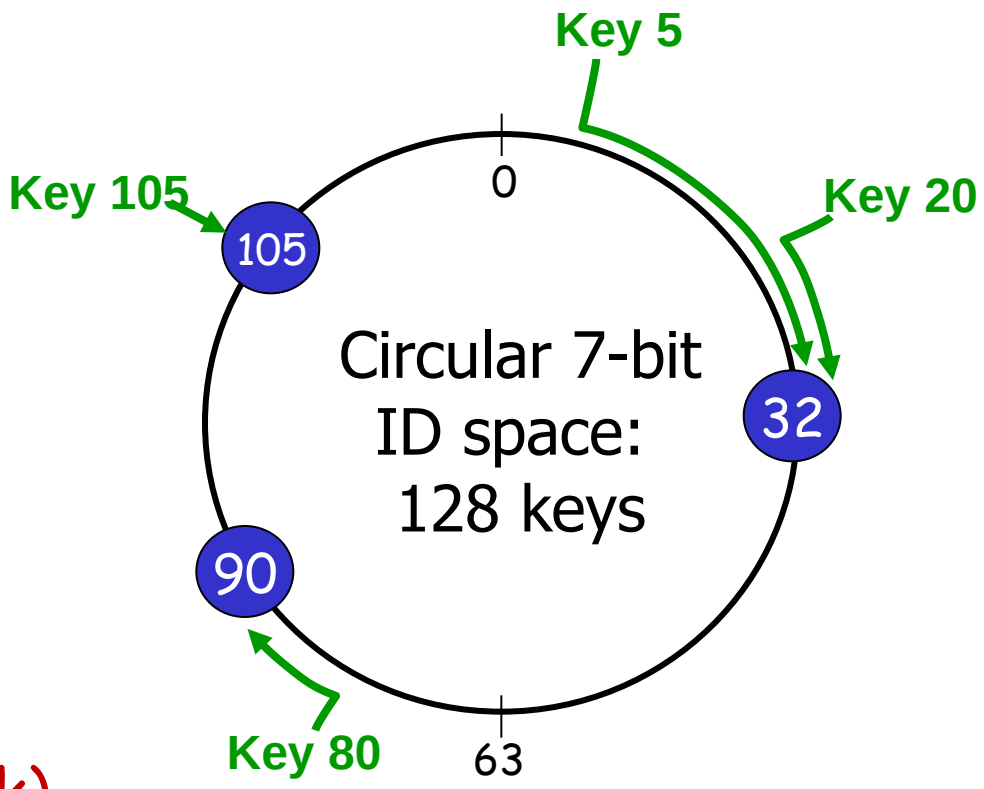
Operazione: data una *chiave* in ingresso; instrada messaggio al nodo che gestisce la *chiave*

DHT obiettivi di progetto

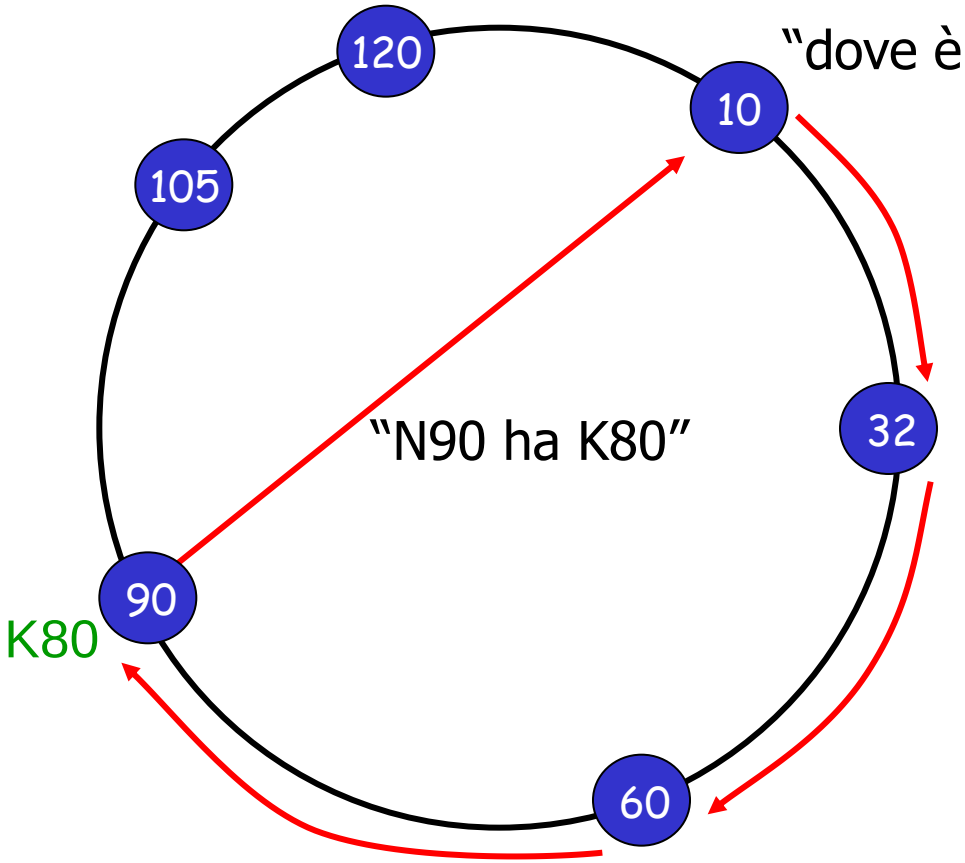
- una rete di "overlay" con:
  - mappaggio flessibile di chiavi e nodi fisici
  - diametro della rete piccolo
  - grado ridotto (fanout)
  - decisioni di instradamento locali
- un meccanismo di "storage" o "memoria" con
  - persistenza best-effort (tramite soft state)

Esempio di DHT: Chord

- spazio di identificativi (chiavi) a m bit: 2^m ids
- identificativi ordinati su *anello logico (Chord ring)* modulo 2^m
- ogni chiave, k, mappata su un nodo dell'anello
  - # chiavi può essere > # nodi
  - la chiava k viene mappata sul primo nodo con id più grande: successor (k)



Lookup base

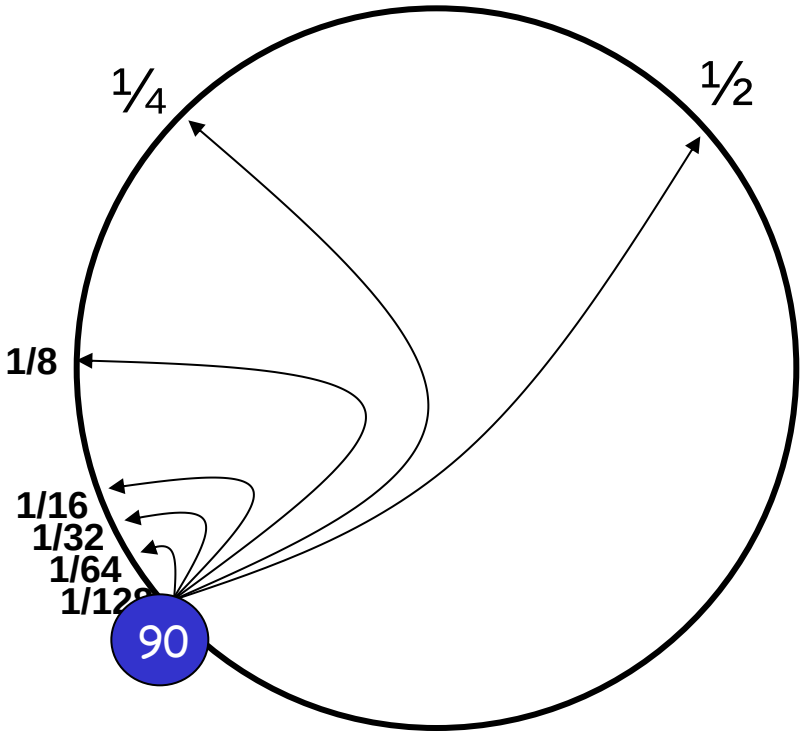


Lookup:

- nodo i inoltra al nodo i+1, finchè si raggiunge il nodo che mantiene k
- memoria O(1), ogni nodo deve conoscere solo un vicino
- tempo di instradamento: O(N)

Accelerazione dei lookup

- lookup vengono accelerati mantenendo informazione di routing addizionale
- ogni nodo mantiene una tabella di routing con (al più) *m* entries (dove N = 2^m) chiamata **finger table**
- la entry i-esima nella tabella contiene l'id del primo nodo, s, che dista da n almeno 2^{i-1} (sull'anello)
- *s* = successor(n + 2^{i-1}) (sempre in modulo N)

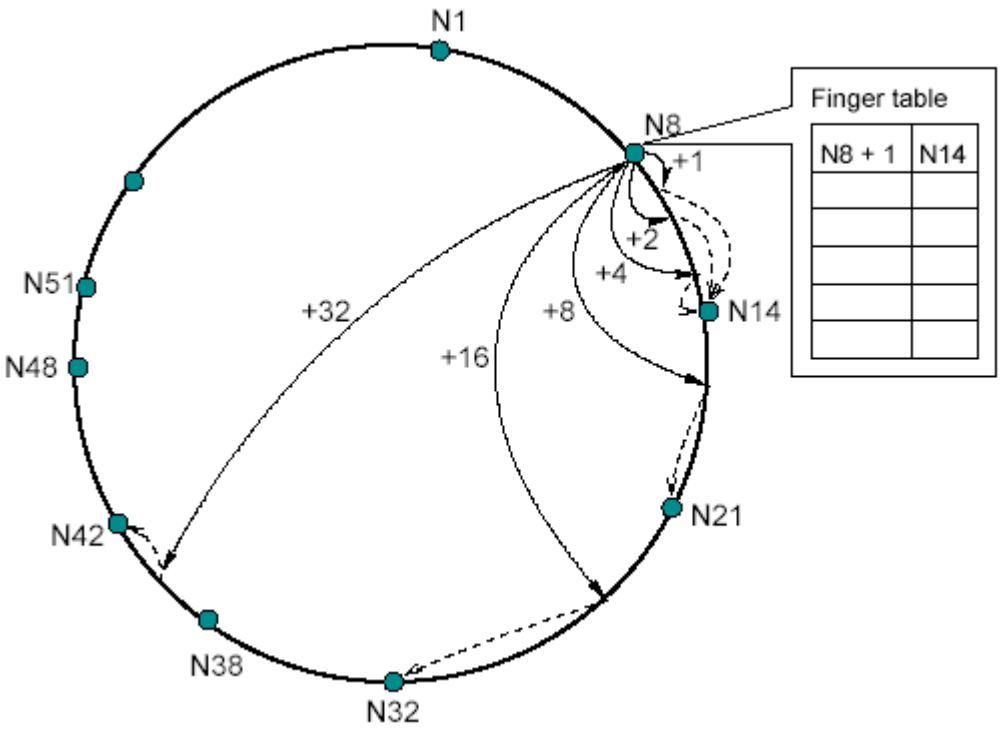


Chord – accelerazione del lookup

Finger table:

finger[i] =

successor $(n + 2^{i-1})$

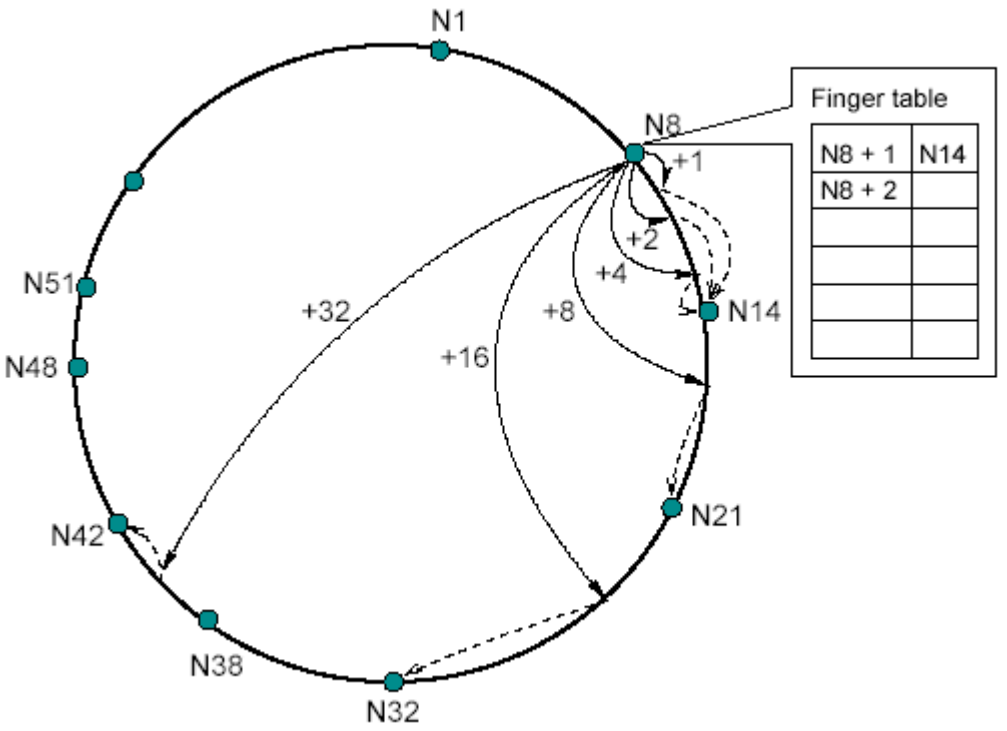


Chord – accelerazione del lookup

Finger table:

finger[i] =

successor $(n + 2^{i-1})$

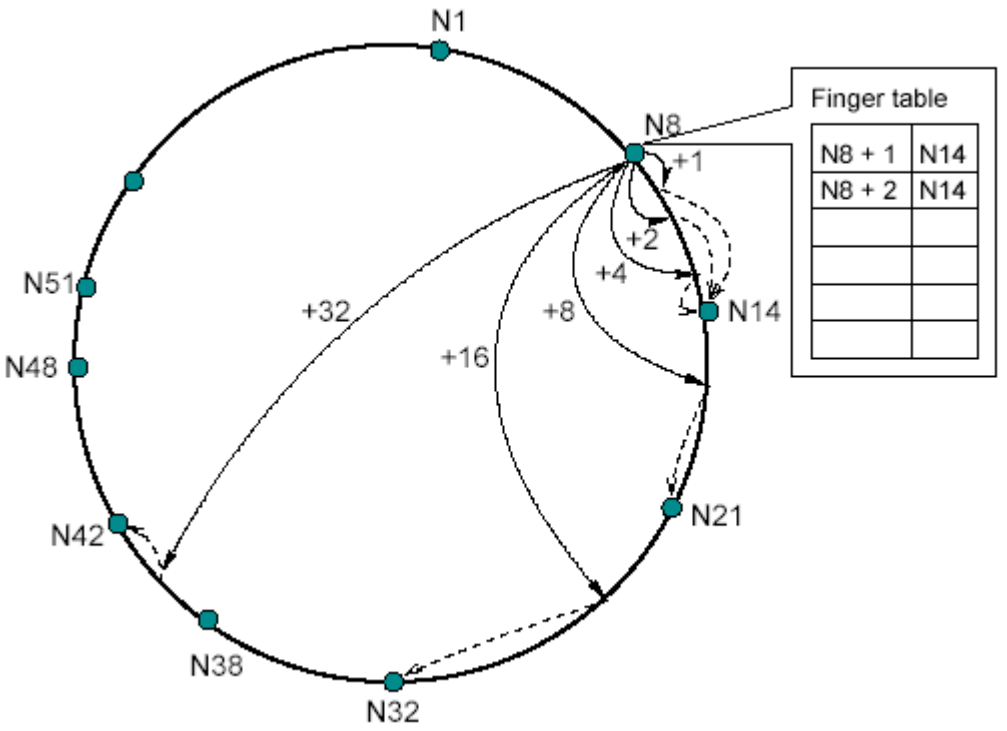


Chord – accelerazione del lookup

Finger table:

finger[i] =

successor $(n + 2^{i-1})$

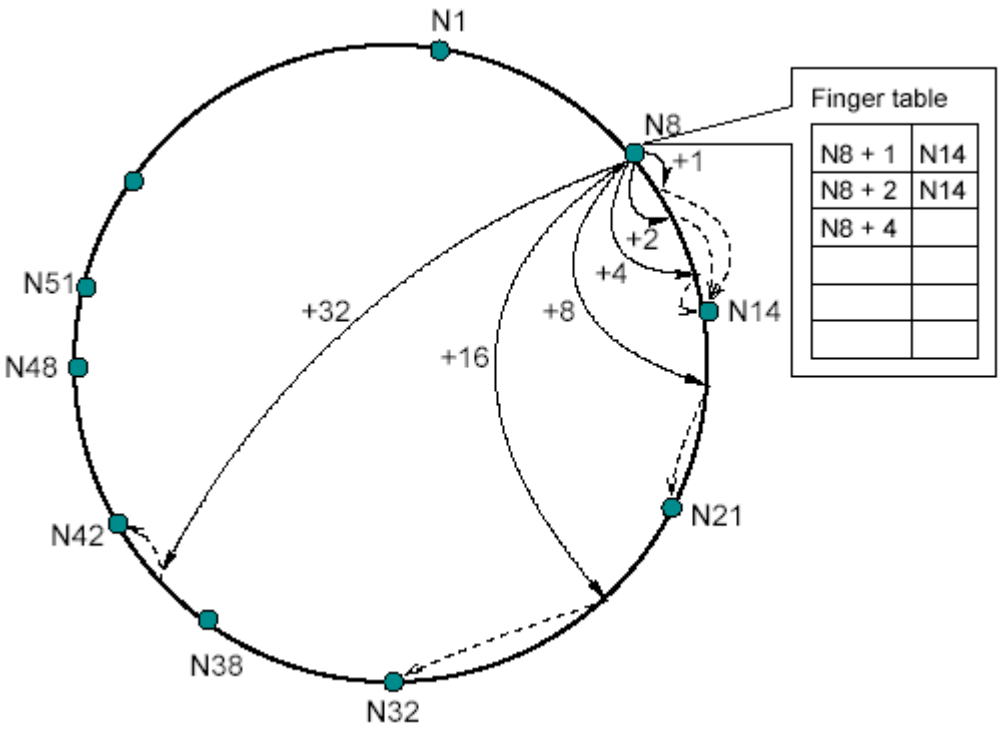


Chord – accelerazione del lookup

Finger table:

finger[i] =

successor $(n + 2^{i-1})$

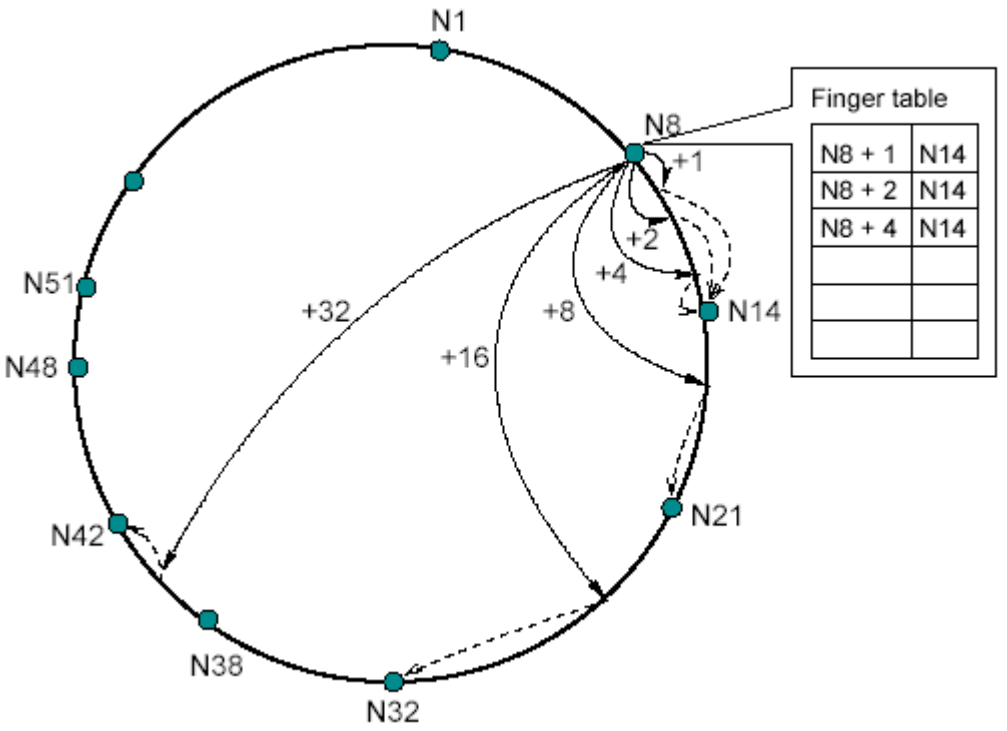


Chord – accelerazione del lookup

Finger table:

finger[i] =

successor $(n + 2^{i-1})$

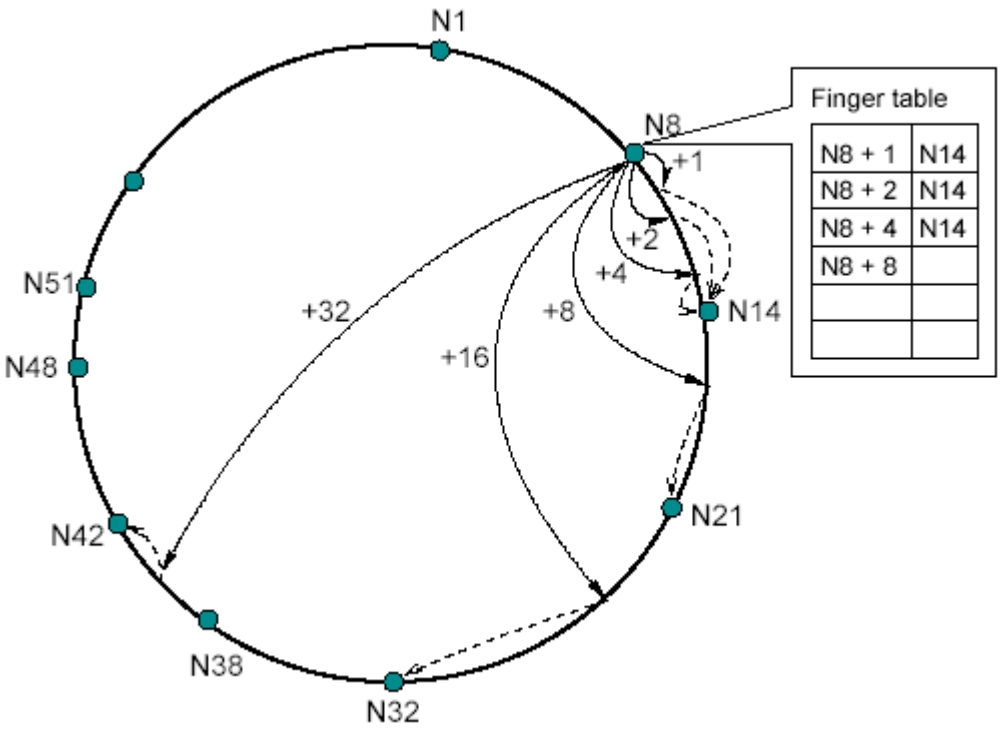


Chord – accelerazione del lookup

Finger table:

finger[i] =

successor $(n + 2^{i-1})$

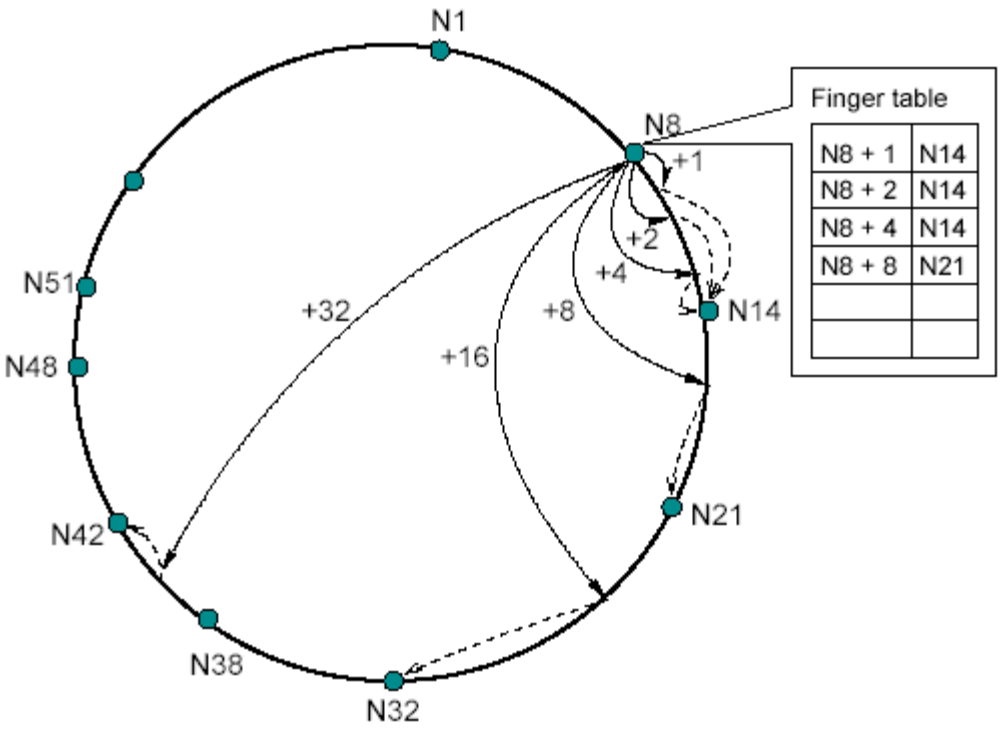


Chord – accelerazione del lookup

Finger table:

finger[i] =

successor $(n + 2^{i-1})$

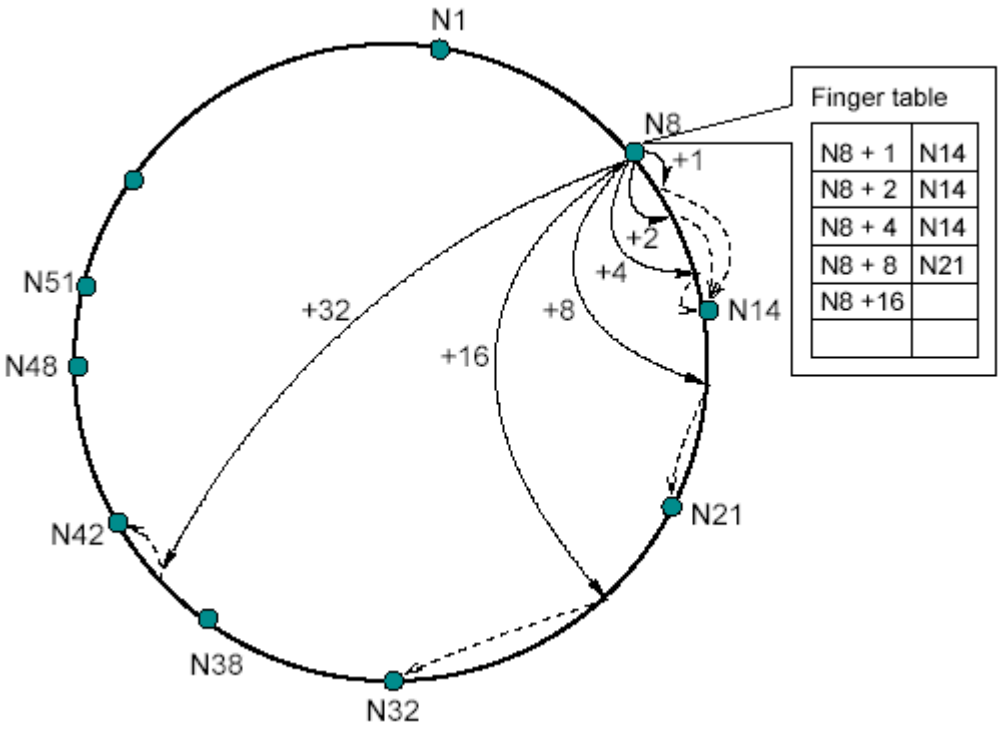


Chord – accelerazione del lookup

Finger table:

finger[i] =

successor $(n + 2^{i-1})$

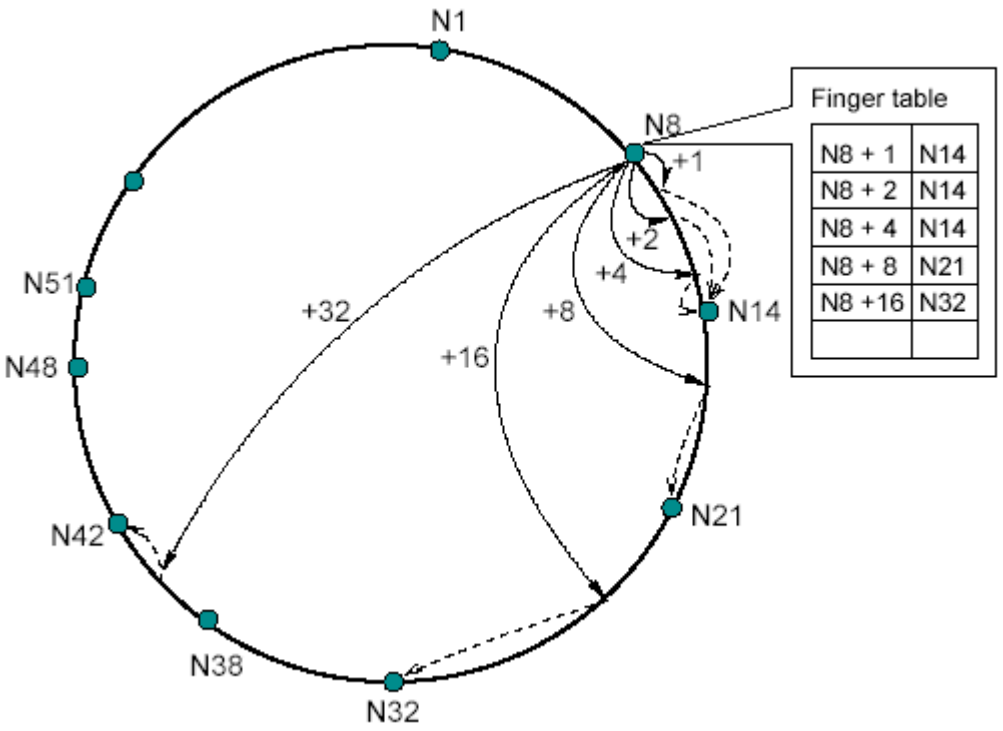


Chord – accelerazione del lookup

Finger table:

finger[i] =

successor $(n + 2^{i-1})$

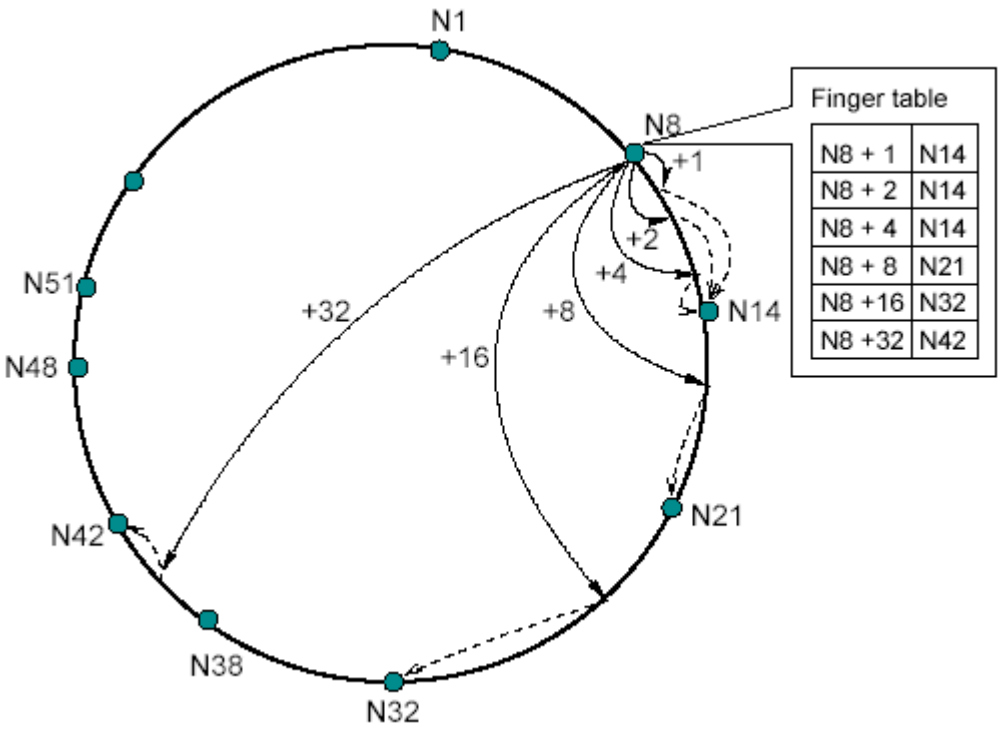


Chord – accelerazione del lookup

Finger table:

finger[i] =

successor $(n + 2^{i-1})$



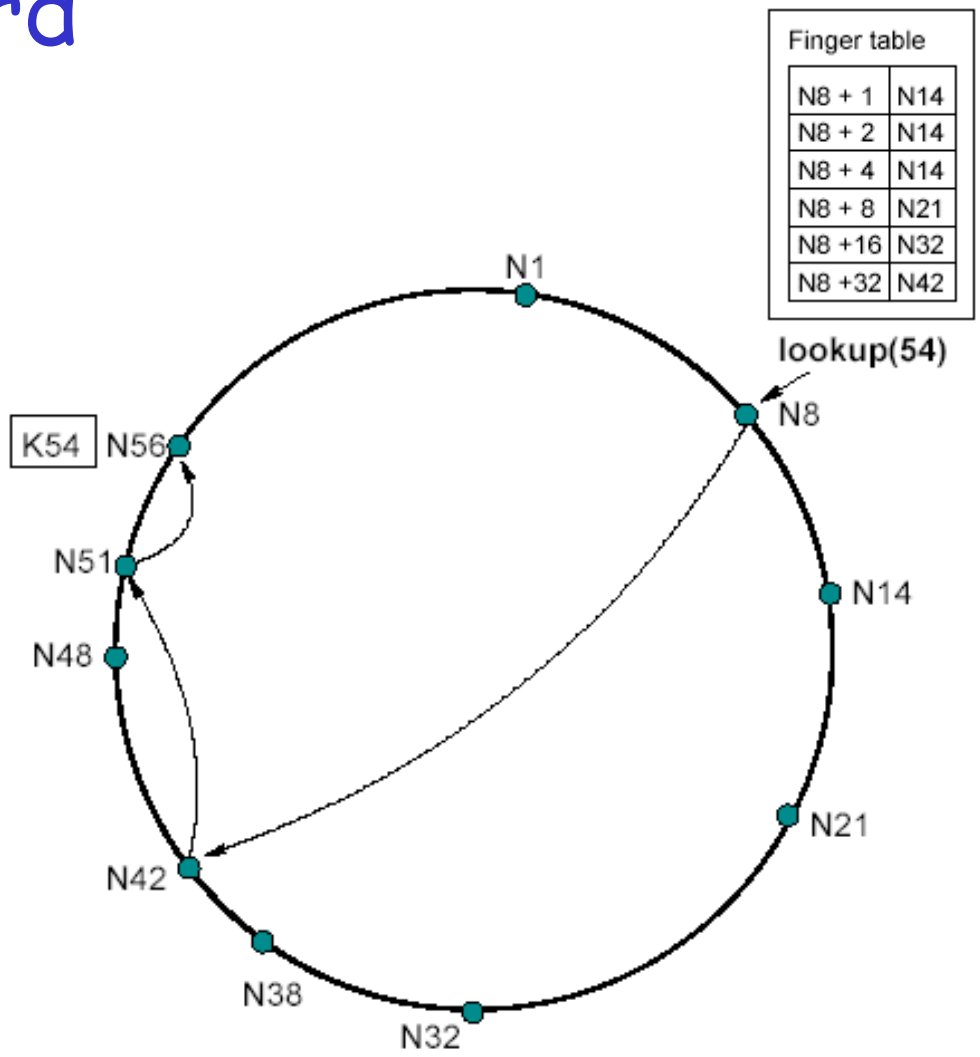
Algoritmo di Chord

Caratteristiche importanti:

- Ogni nodo mantiene informazione solo di un numero limitato di altri nodi (m)
- Un nodo ha una conoscenza più accurata della parte di anello seguente a lui più vicina, e più vaga della parte più lontana
- Una finger table in generale non contiene abbastanza informazione per determinare direttamente il successore di una generica chiave k
- Problema risolto per "approssimazioni successive", tramite query iterative dirette ai nodi concosciuti che immediatamente precedono quello cercato

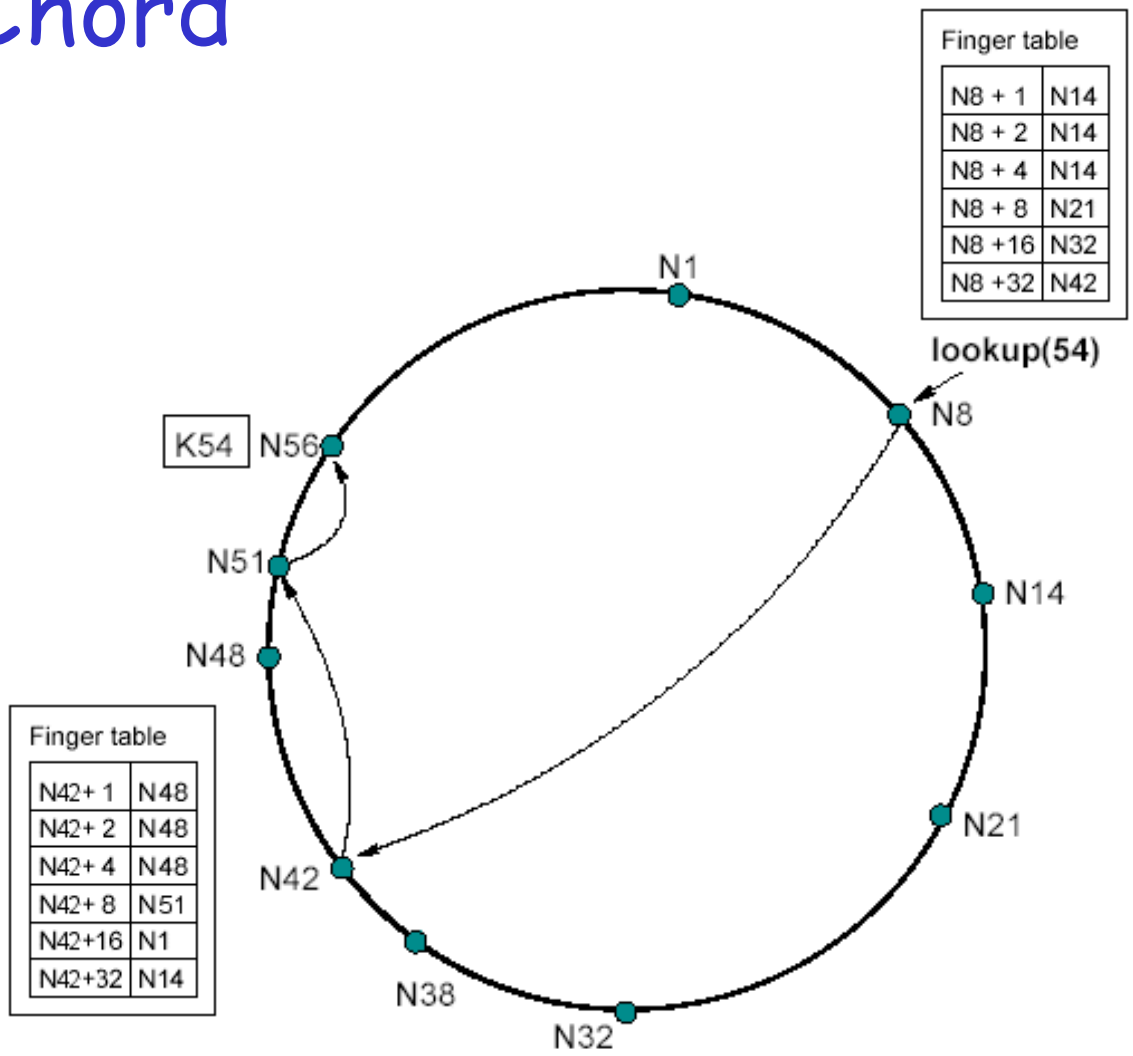
Algoritmo di Chord

- idea: cerca nella finger table il nodo che immediatamente precede l'id cercato
- Invoca find_successor da quel nodo



=> Numero di hop O(log N)!

Algoritmo di Chord



=> Numero di hop O(log N)!

Chord e DHT

- *churn:* nodi entrano ed escono dal sistema
  - le chiavi memorizzate vanno spostate altrove (nodo in uscita) o distribuite al nodo (in entrata) dai nodi vicini
  - le finger table devono essere aggiornate
  - troppo churn: troppo overhead
- occorre un middleware che gestisca memorizzazione e recupero dei valori
  - monitoraggio distribuito che reagisca in modo automatico al churn