Προηγμένα Θέματα Αρχιτεκτονικής Συστημάτων

Θέματα Caching

Διδάσκων: Δημόκας Νικόλαος

Ορισμοί

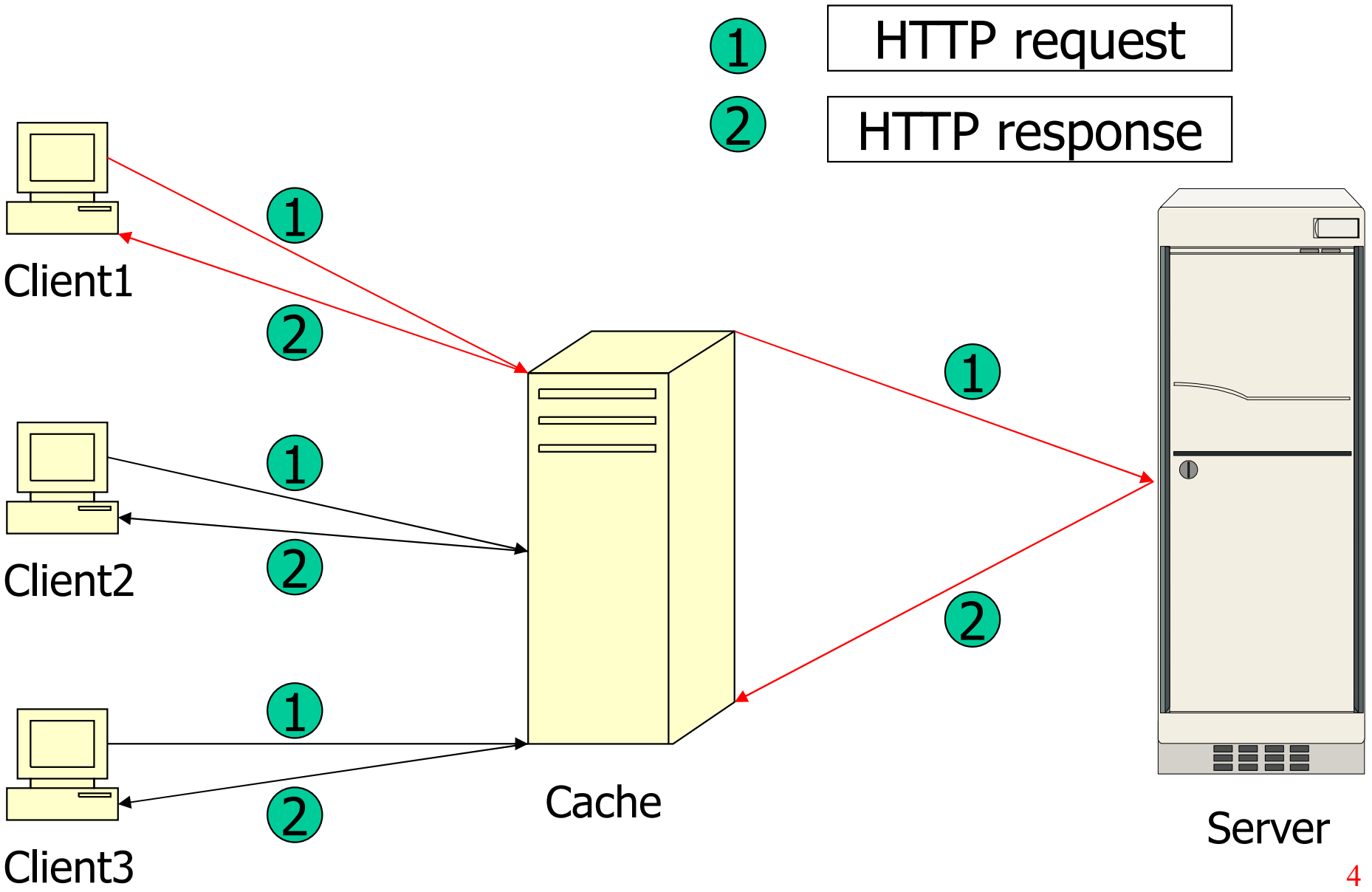
- **Replication**: Η διατήρηση πολλαπλών (συνεπών) αντιγράφων ενός αντικειμένου
  - Στατικό replication: ο αριθμός και η τοποθεσία των αντιγράφων ορίζονται στατικά.
  - Dynamic replication: ο αριθμός και η τοποθεσία των αντιγράφων ορίζονται δυναμικά (at run-time)
- **Caching**: Η διατήρηση ενός αντιγράφου ενός αντικειμένου στην τοπική μνήμη. *Το αντίγραφο αποκτάται όταν γίνεται η πρώτη προσπέλασή του*.
- **Prefetching**: *Η απόκτηση ενός αντιγράφου πριν την προσπέλασή του* (to hide access latency).
- **Hoarding**: Η προφόρτωση ενός αντιγράφου έτσι ώστε ο κινητός κόμβος να μπορεί να λειτουργήσει όσο διάστημα είναι αποσυνδεδεμένος από το δίκτυο (π.χ.. prefetching για ανοχή σε αποσυνδέσεις).

Εισαγωγικά στο Caching

- Η τεχνική του caching είναι ευρέως διαδεδομένη στο χώρο των:
  - Λειτουργικών Συστημάτων
  - Βάσεων Δεδομένων
- Η τεχνική του Caching για το Web έχει ερευνηθεί διεξοδικά:
  - Internet Cache Protocol
  - Cache Digests
  - Summary Cache

❖ Επιτρέπουν στους proxies να ανταλλάσσουν πληροφορίες σχετικά με αποθηκευμένα δεδομένα

Web caching


1
HTTP request
2
HTTP response
Client1
Client2
Client3
Cache
Server

Proxy Caching Systems

- Ένας proxy αναπτύσσεται συνήθως σε μια άκρη του δικτύου.
- Εάν μια αίτηση για δεδομένα μπορεί να ικανοποιηθεί από τη proxy cache, τότε λέμε ότι έχουμε ένα ***cache hit***, διαφορετικά, έχουμε ένα ***cache miss***.
- Ένας browser ενός πελάτη ανακτά ένα Web αντικείμενο αρχικοποιώντας μία HTTP GET εντολή με τη διεύθυνση του αντικειμένου.

Proxy Caching Systems (σσν.)

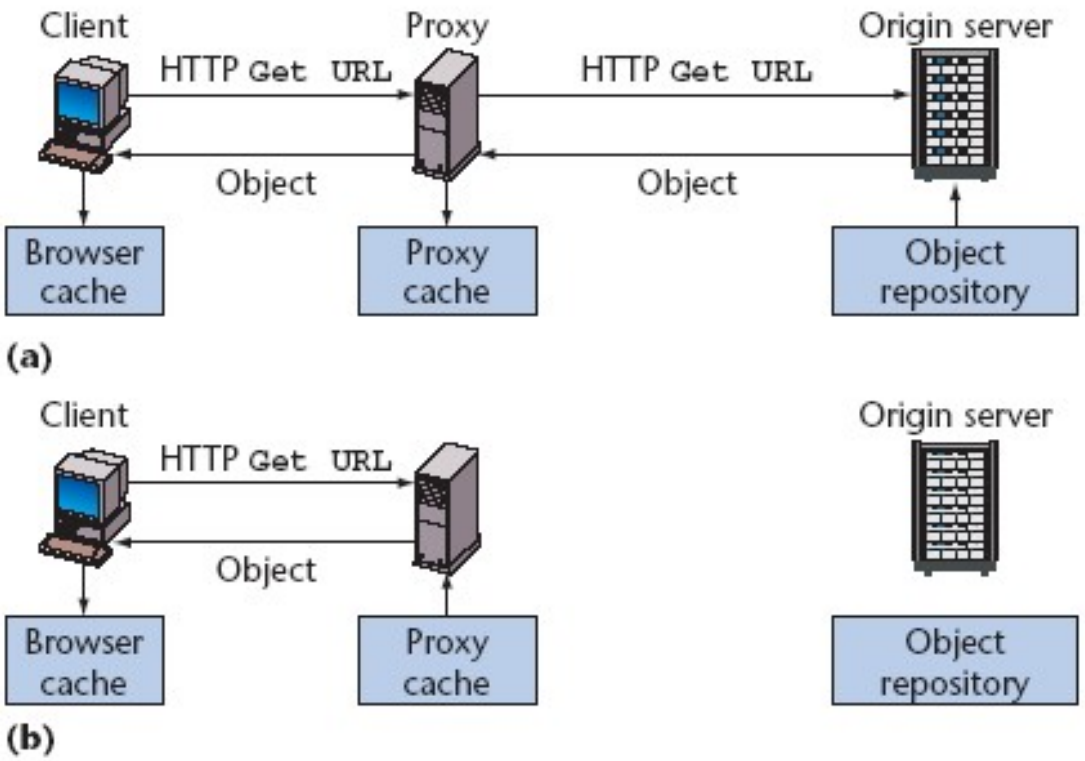


Figure 1. A stand-alone proxy. The browser initiates an HTTP GET command, and if it can't satisfy the request from local cache, it sends the request to the proxy. (a) When the proxy can't satisfy a request, a cache miss occurs. (b) A cache hit: the proxy finds the requested object and returns it to the client.

Caching Challenges and Solutions

- **Αντικατάσταση της Cache** και **prefetching**, **συνέπεια της cache**, **και συνεργατικό caching** είναι από τα σημαντικότερα θέματα.
- Διαφορετικές λύσεις:
  - Αρχικά, υπάρχει ένα θέμα σχετικά με το **μέγεθος**. Το διαδίκτυο είναι το μεγαλύτερο interconnected network.
  - Web application users εμφανίζουν επίσης **μεγάλη ετερογένεια** σε ότι αφορά το υλικό, το λογισμικό, το bandwidth, και τη συμπεριφορά πρόσβασης.
  - Επιπρόσθετα, εξαιτίας της μη κεντρικοποιημένης διαχείρισης, θέματα που αφορούν την **ασφάλεια** και την **ιδιωτικότητα** είναι περίπλοκα.

Cache Replacement and Prefetching

- Πολιτική αντικατάστασης της cache
  - Έρχεται αντιμέτωπη με μη επαρκή χωρητικότητα του δίσκου. Ο proxy πρέπει να αποφασίσει ποια αντικείμενα πρέπει να απομακρύνει από τη cache όταν εισέρχεται σε αυτή ένα νέο αντικείμενο.

- Πολιτική cache prefetching
  - Το cache prefetching σχετίζεται με την πολιτική αντικατάστασης, αλλά αντίθετα με το caching των δεδομένων, το prefetching φορτώνει εκ των προτέρων τα δεδομένα από το server στις caches έτσι ώστε να διευκολύνει τη πρόσβαση των δεδομένων στο κοντινό μέλλον.

Cache Replacement and Prefetching

- Όμως, μία πολιτική cache prefetching πρέπει να είναι πολύ προσεκτικά σχεδιασμένη: *εάν αποτύχει να προβλέψει τις μελλοντικές ανάγκες των χρηστών σε δεδομένα, τότε συνεπάγεται σπατάλη του bandwidth του δικτύου και του χώρου της cache*.

- Μπορούμε να ταξινομήσουμε τις **πολιτικές prefetching σε τρεις κατηγορίες βασισμένες στο τύπο της πληροφορίας** που χρησιμοποιεί ο μηχανισμός πρόβλεψης:
  - *Πρότυπο μικτής πρόσβασης*
  - *Πρότυπο πρόσβασης ανα χρήστη*
  - *Με βάση τη δομή της πληροφορίας*

ICP (Internet Cache Protocol)

- ICP είναι ένα πρωτόκολλο που χρησιμοποιείται για το συντονισμό και συνεργασία των web caches
- Σκοπός του πρωτοκόλλου είναι η **εύρεση** της περισσότερο κατάλληλης **τοποθεσίας** για την **ανάκτηση ενός ζητούμενου αντικειμένου** όταν χρησιμοποιούνται πολλαπλές caches.
- Στόχος:
  - **Αποδοτική χρησιμοποίηση των caches**
  - **Ελαχιστοποίηση των ερωτημάτων προς τον server**.
- Σχεδιάστηκε για μη αξιόπιστες αλλά γρήγορες συνδέσεις
- Χρησιμοποιεί το πρωτόκολλο UDP

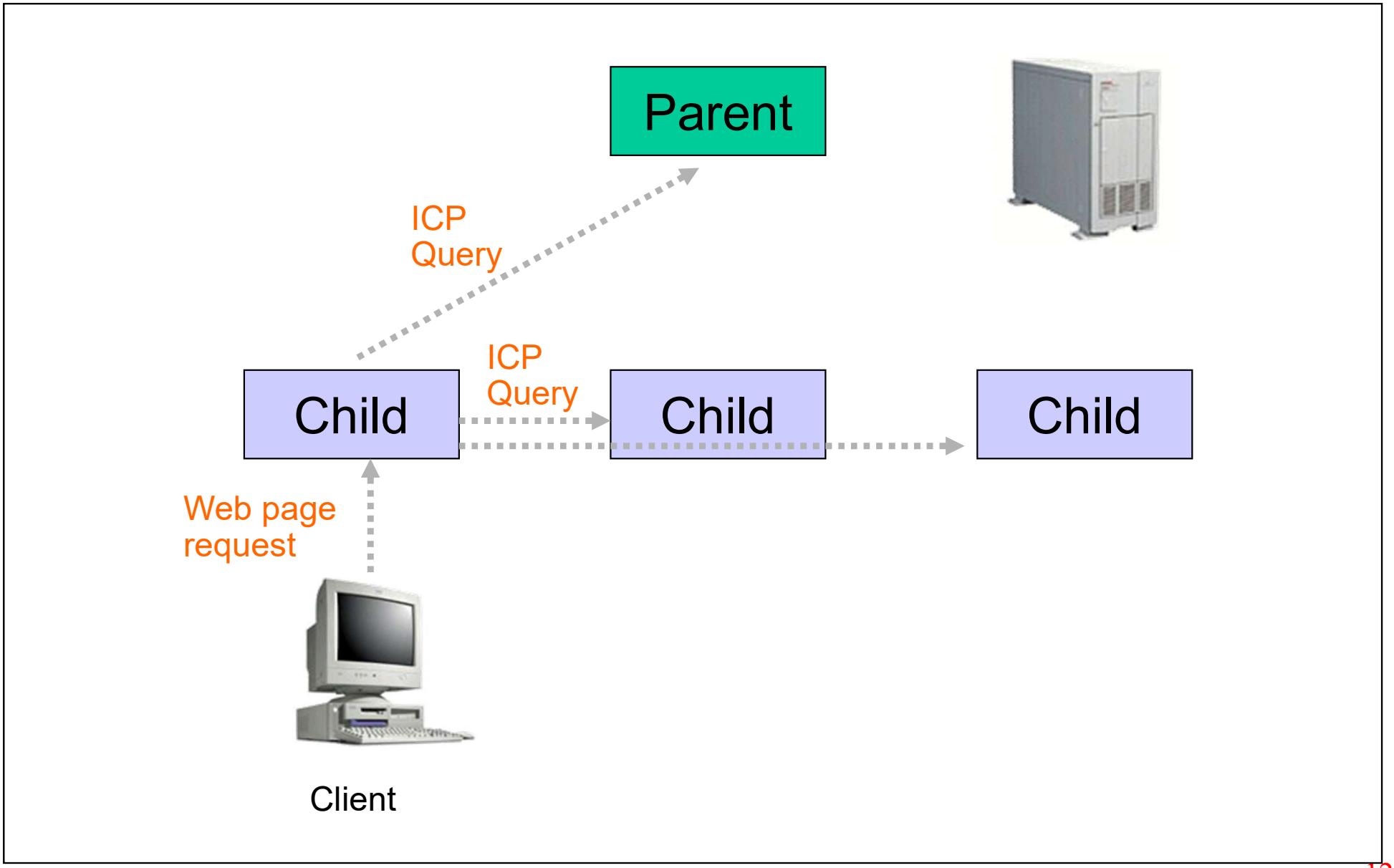
ICP (Internet Cache Protocol)

ICP Header

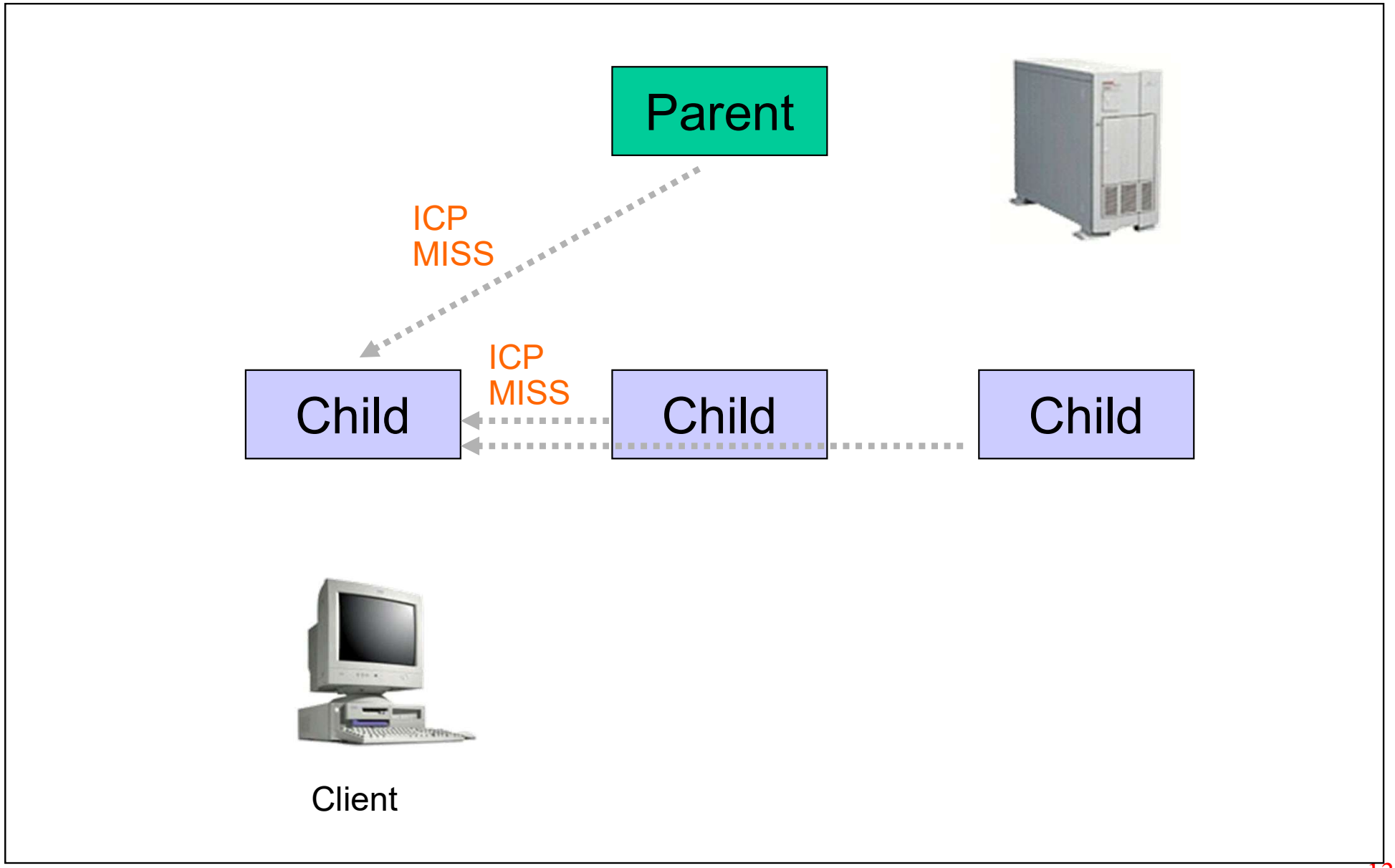
Bit offset	Bits 0–7	8–15	16-31
0	Opcode	Version	Message Length
32	Request number		
64	Options		
96	Option Data		
128	Sender Host Addresss		
160+	Data		

- Opcode
  - 1 → ICP_OP_QUERY
  - 2 → ICP_OP_HIT
  - 3 → ICP_OP_MISS … και άλλα
- ICP αριθμός έκδοσης του πρωτοκόλλου
- Μήκος μηνύματος (πλήθος χαρακτήρων)
- Αριθμός ερωτήματος. Η τιμή αυτή πρέπει να αντιγραφεί και στο μήνυμα της απάντησης
- Η IPv4 διεύθυνση του κόμβου που στέλνει το ICP μήνυμα

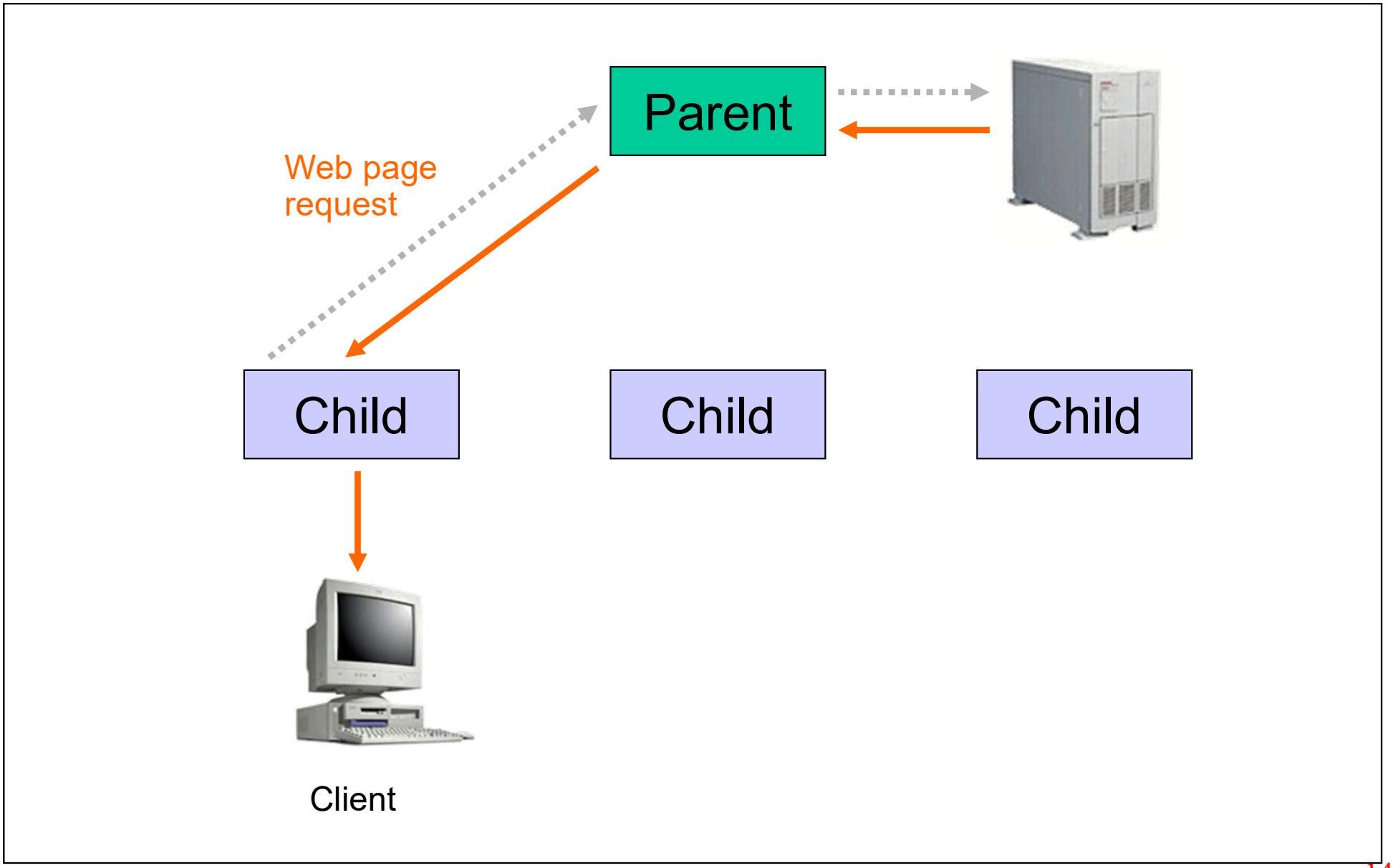
Squid


Parent
ICP
Query
ICP
Query
Child
Child
Child
Web page
request
Client

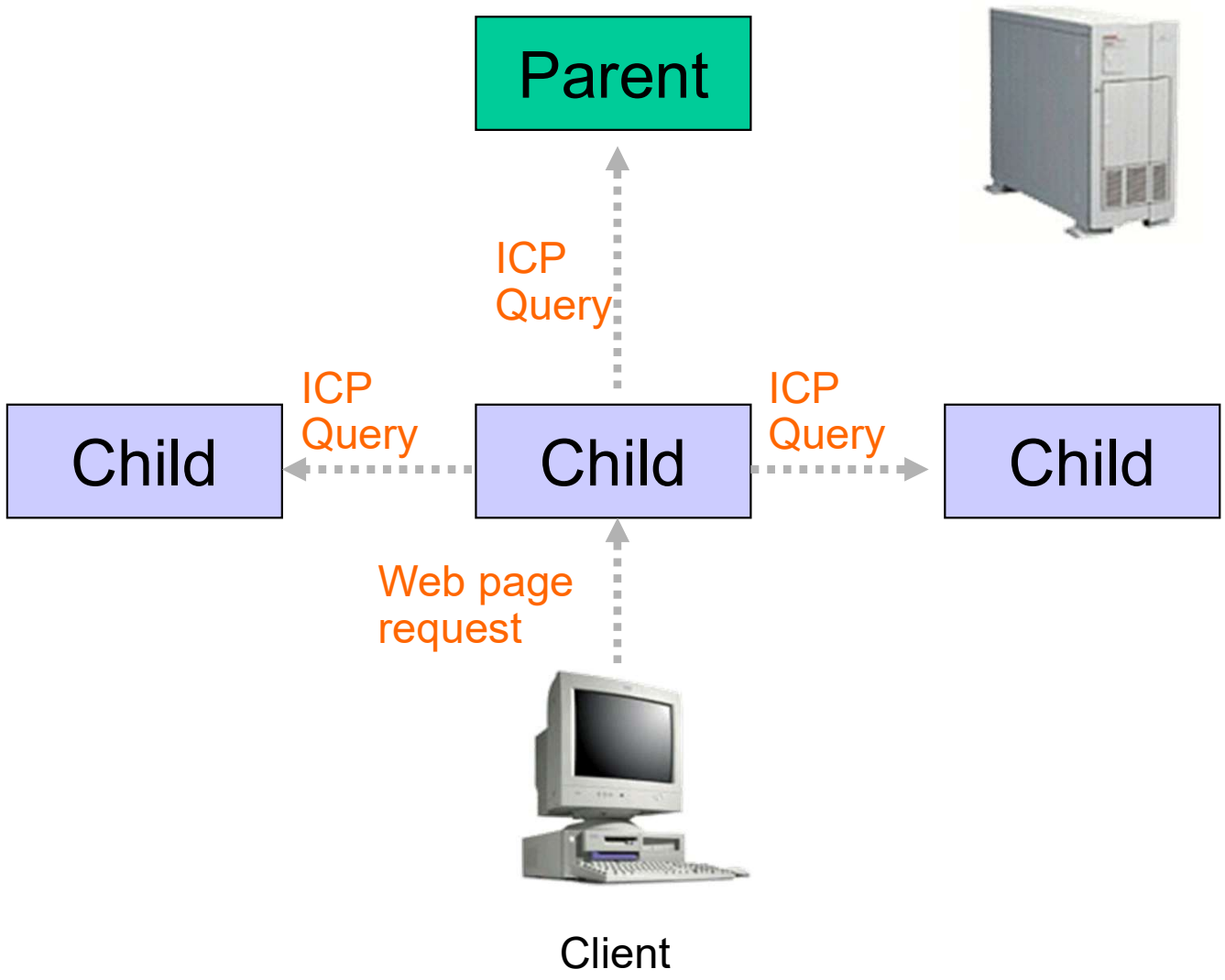
Squid



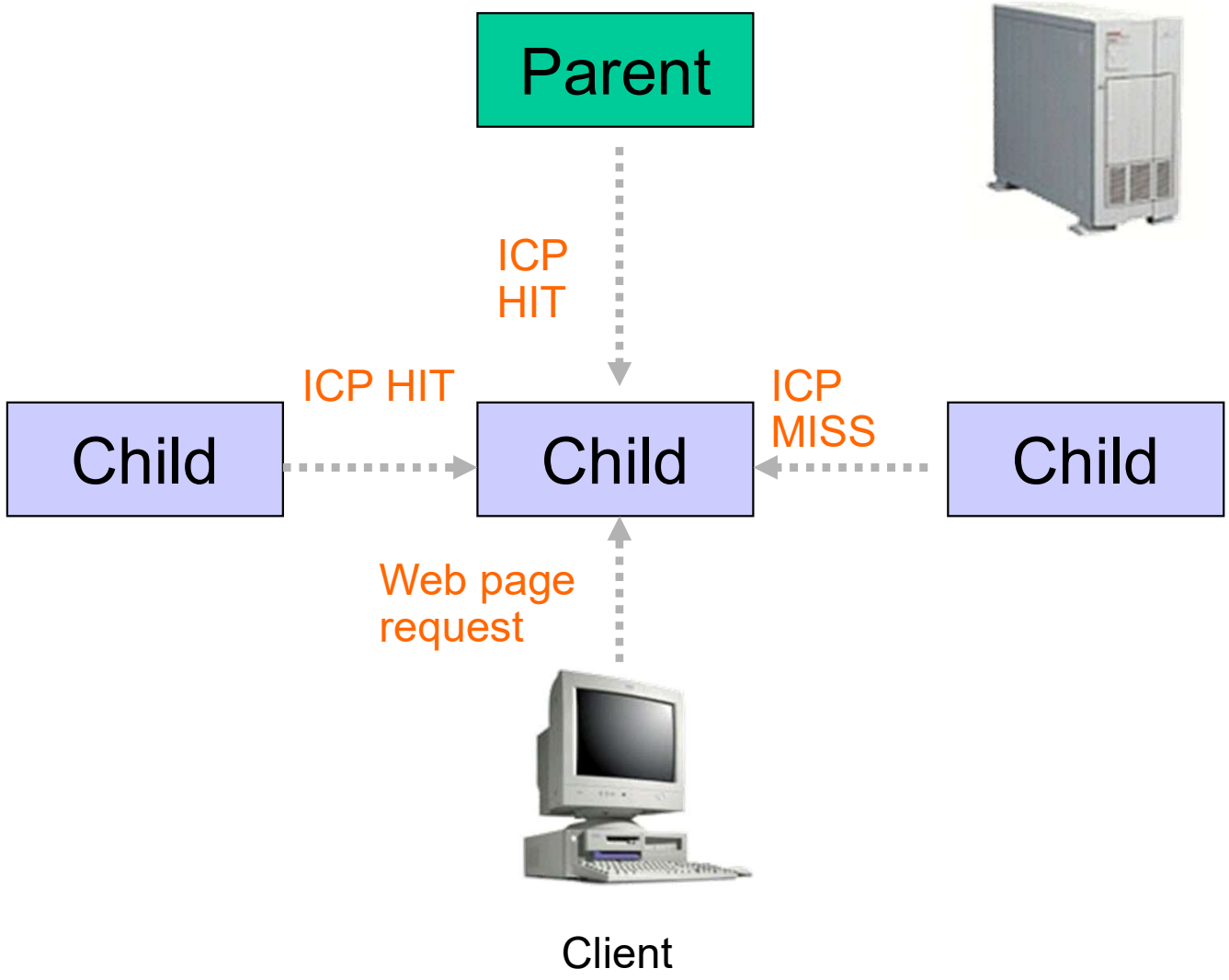
Squid



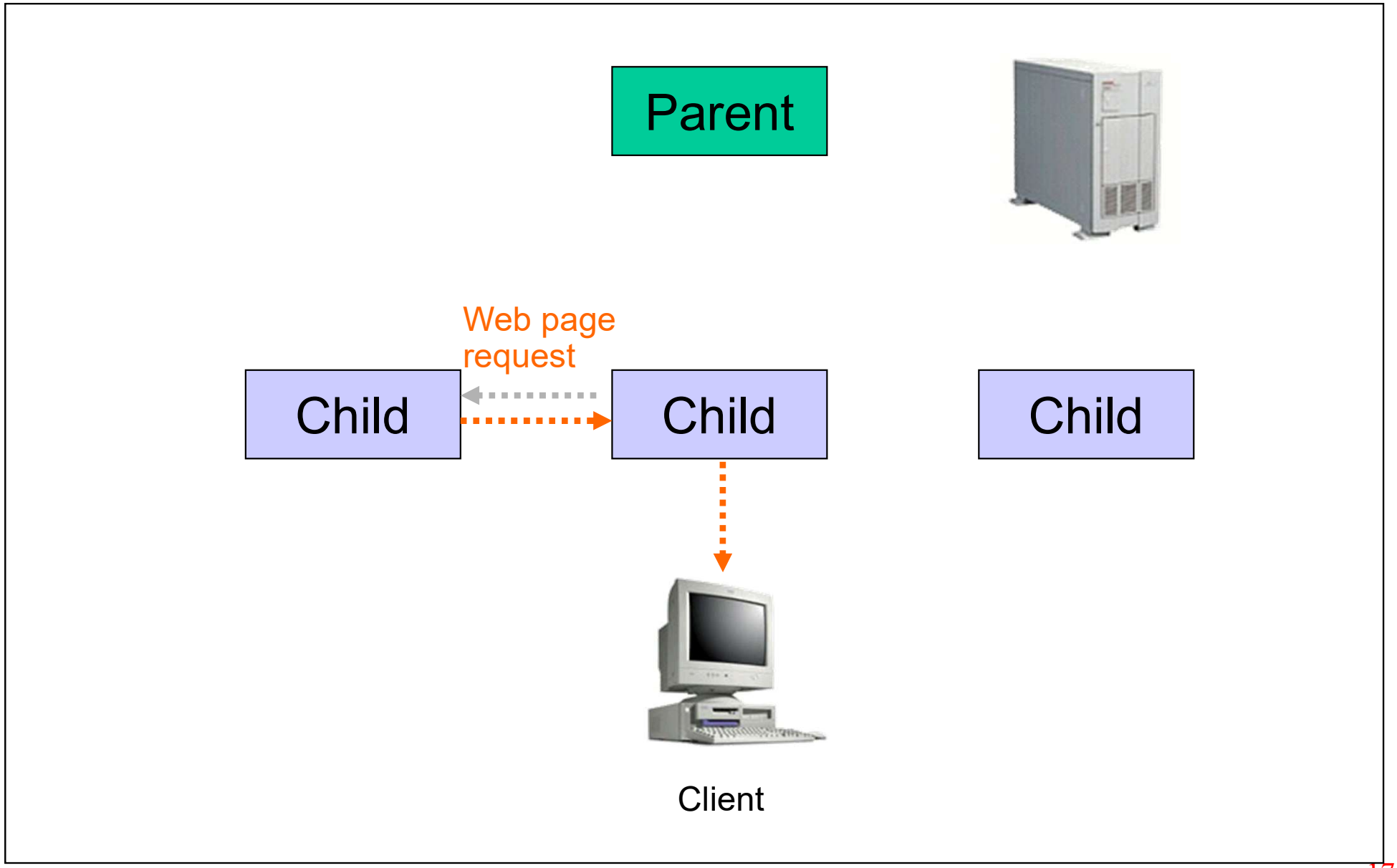
Squid


Parent
ICP
Query
ICP
Query
Child
ICP
Query
Child
Child
Web page
request
Client

Squid


Parent
ICP
HIT
ICP HIT
ICP
MISS
Child
Child
Child
Web page
request
Client

Squid



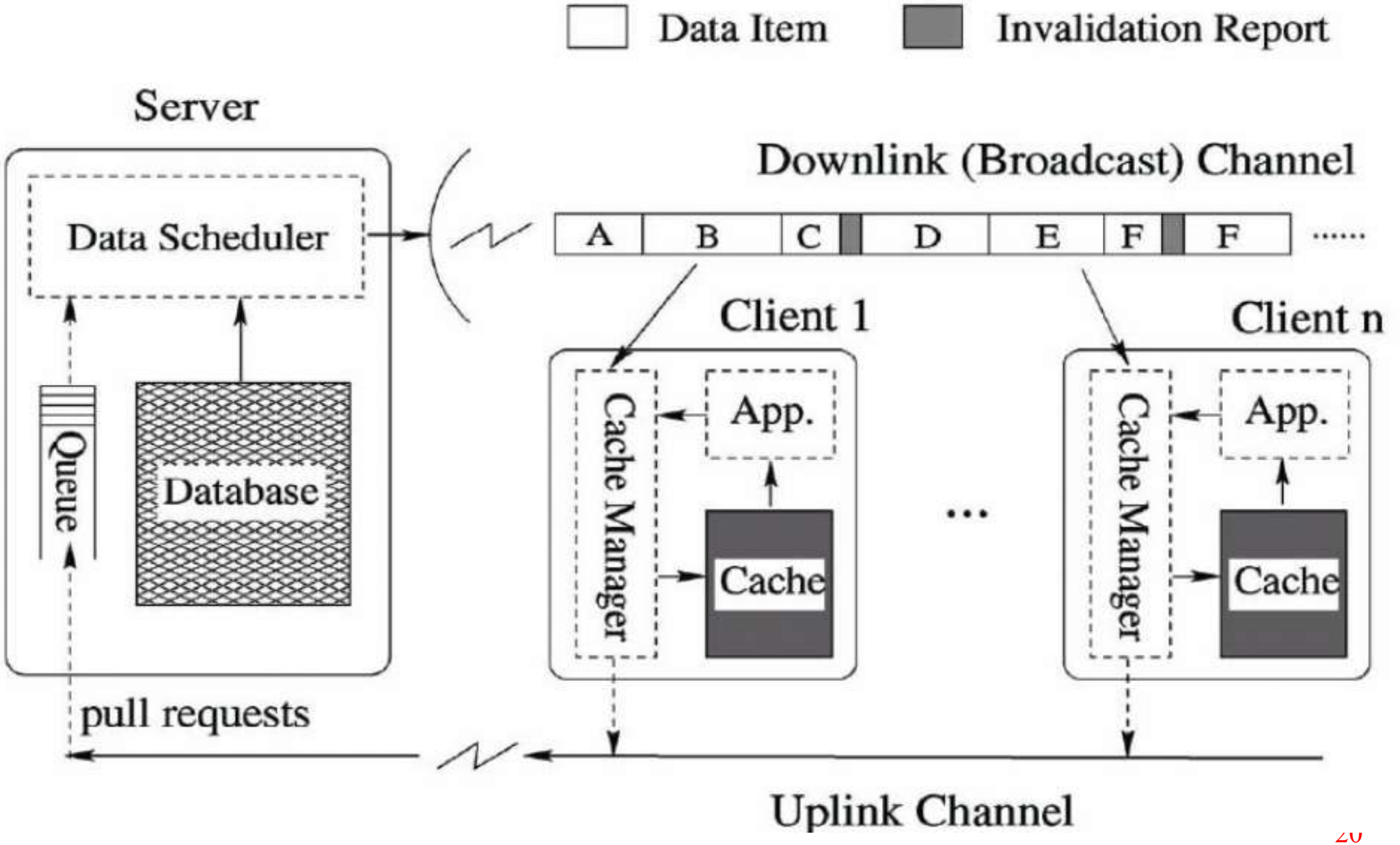
Μειονεκτήματα

- Οι **πολιτικές** που αναπτύχθηκαν για το Web θεωρούσαν:
  - Σταθερή δικτυακή τοπολογία
  - Ισχυρές υπολογιστικές και επικοινωνιακές δυνατότητες

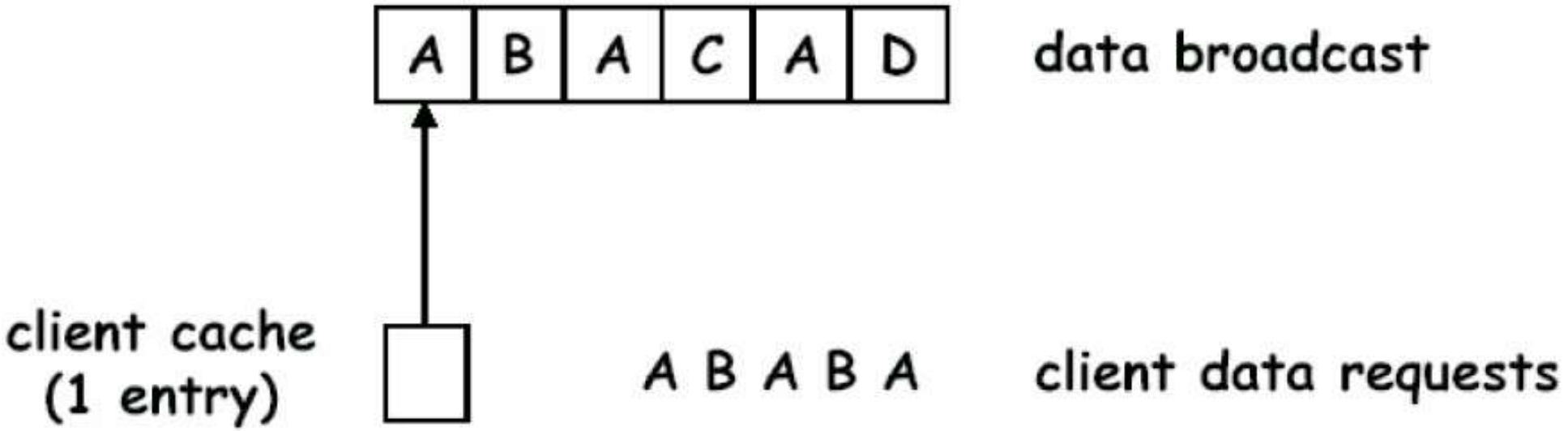
Caching στους κινητούς πελάτες

- Προγράμματα εκπομπής
  - ***Βασίζονται στη μέση πιθανότητα προσπέλασης***: μέσος όρος πάνω σε ΟΛΟΥΣ τους πελάτες
  - ***Όχι αναγκαστικά βέλτιστη για κάθε έναν πελάτη***
- Πώς μπορεί κάθε πελάτης να υποβοηθήσει τον εαυτό του?
  - Caching: προσωρινή αποθήκευση των δεδομένων που λαμβάνει
  - Πολιτική Αντικατάστασης Caching: όταν εξαναγκάζεται να αντικαταστήσει κάποιο αντικείμενο (επειδή η cache είναι πλήρης), αντικαθιστά εκείνα που είναι λιγότερο πιθανό να χρησιμεύσουν στο μέλλον

Το γενικό μοντέλο caching

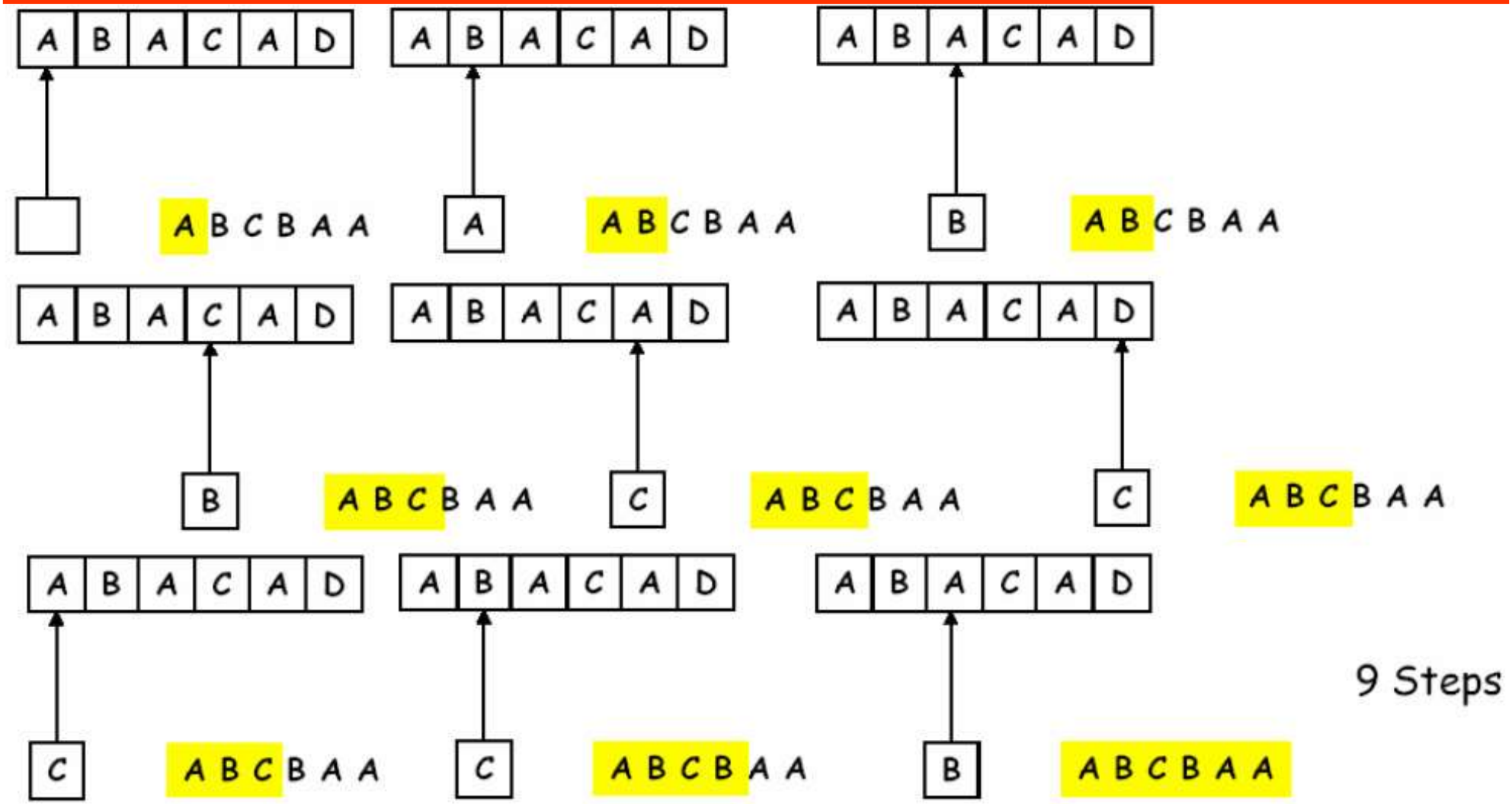


Caching στους κινητούς πελάτες



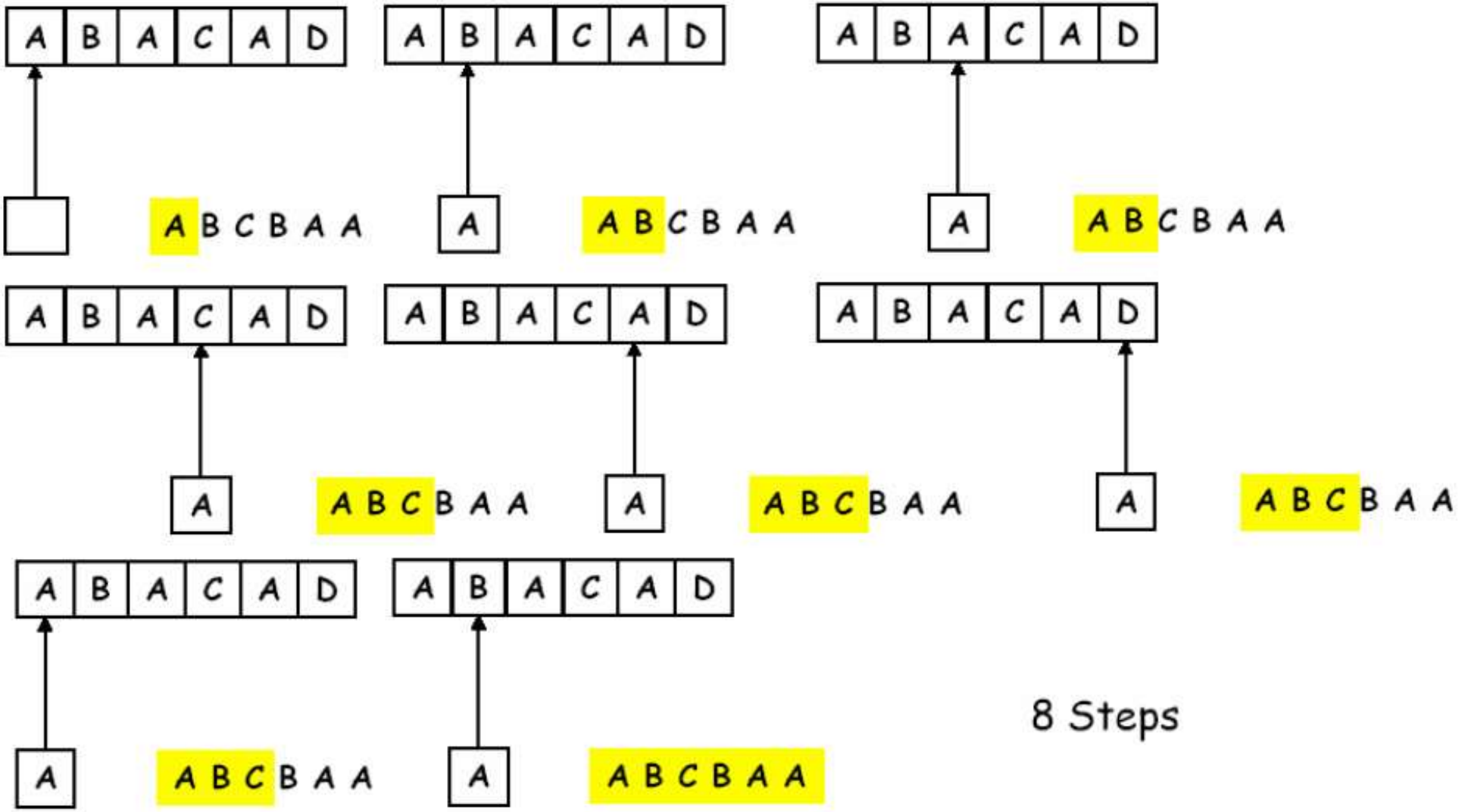
- Ποια **πολιτική αντικατάστασης** (replacement strategy ή policy ή algorithm) θα πρέπει να χρησιμοποιούν οι πελάτες;

Least Recently Used (LRU)



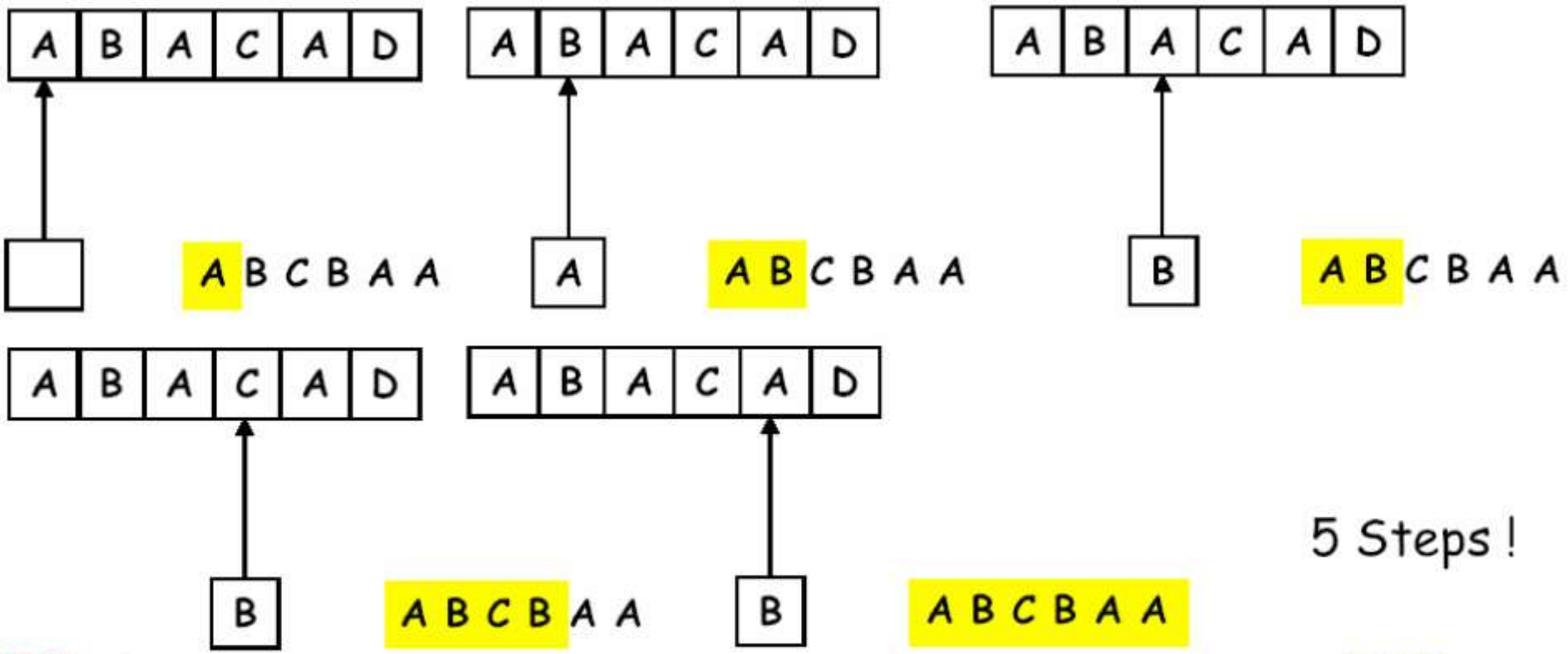
LRU: Αντικατάσταση του αντικειμένου που έχει χρησιμοποιηθεί παλιότερα στο παρελθόν

Most Probable Accessed (MPA)



MPA: Αντικατάσταση του αντικειμένου που θα χρησιμοποιηθεί λιγότερο συχνά στο μέλλον

Probability Inv. Broadc. Frequency



PIX: Αντικατάσταση του αντικειμένου με το μικρότερο P/X, όπου

- P: είναι η πιθανότητα προσπέλασης του αντικειμένου
- X: είναι η συχνότητα εκπομπής του αντικειμένου

PIX(A) = $\frac{1}{2}$ / $\frac{1}{2}$ = 1, **PIX(B) = 1/3 / 1/6 = 2**, PIX(C) = 1/6 / 1/6 = 1

Κίνητρο για prefetching

- Οι προηγούμενοι αλγόριθμοι αποθηκεύουν στην cache αντικείμενα, μόνο αφού ζητηθούν
- Εναλλακτικά, *ο κινητός πελάτης μπορεί να "κατεβάζει" αντικείμενα από το κανάλι, αφού ούτως ή άλλως το ακούει*
  - Ο στόχος είναι να ελαττώσει το χρόνο απόκρισης
- Μέθοδοι prefetcing:
  - Tag Team Caching
  - Ευριστικό Prefetching

Η έννοια του tag team caching

- Tag Team Caching – Τα αντικείμενα ***συνεχώς αντικαθιστούν το ένα το άλλο*** *μέσα στην* ***cache***
- Για παράδειγμα,
  - έστω δυο αντικείμενα x και y τα οποία εκπέμπονται από το κανάλι
  - Ο πελάτης αποθηκεύει το x όταν εκπέμπεται στο κανάλι
  - Εκδιώχνει το x και αποθηκεύει το y, όταν εκπέμπεται το y

Ευριστικό Prefetching

- Εκτελεί έναν υπολογισμό για κάθε αντικείμενο που εκπέμπεται στο κανάλι με βάση την **πιθανότητα προσπέλασης P** για το αντικείμενο και το **χρόνο T** που θα περάσει μέχρι να εμφανιστεί ξανά η σελίδα στο κανάλι εκπομπής
- Εάν η τιμή **P*T** της σελίδας που εκπέμπεται είναι υψηλότερη από μια σελίδα της cache, τότε η σελίδα με τη χαμηλότερη τιμή P*T, εκδιώκεται από την cache

Ευριστικό Prefetching

Page	Access Probability	Broadcast Frequency (per Period)	$\mathcal{PIX}$ Value
A	P	2	P/2
B	P/2	2	P/4
C	P/2	1	P/2

Table 2: Access and Frequency Values for the `pt` Example

Cooperative Caching – CoCa

- Το CoCa διευκολύνει τους κόμβους να **διαμοιράσουν τα περιεχόμενα των caches** μεταξύ τους για να **μειώσουν το πλήθος των αιτήσεων που φθάνουν στον εξυπηρετητή** και τον αριθμό των αποτυχημένων προσβάσεων (access misses).
- Το CoCa framework αποτελείται από 3 πρωτόκολλα:
  - CacheData
  - CachePath
  - HybridCache

CacheData

- Οι ενδιάμεσοι κόμβοι μπορούν να κάνουν cache τα δεδομένα για να εξυπηρετούν τις μελλοντικές αιτήσεις αντί να φέρνουν τα δεδομένα από το Data Center
- ***Ένας ενδιάμεσος κόμβος αποθηκεύει τοπικά ένα διερχόμενο αντικείμενο όταν βρίσκει ότι το αντικείμενο d_i είναι δημοφιλές (υπάρχουν πολλές αιτήσεις για αυτό) ή έχει αρκετό χώρο στη cache***.
  - **Ένας ενδιάμεσος κόμβος δεν αποθηκεύει ένα αντικείμενο όταν όλες οι αιτήσεις για αυτό προέρχονται από τον ίδιο κόμβο**.
- Ο κανόνας σχεδιάστηκε για να μειωθεί η απαίτηση για μεγάλη χωρητικότητα της cache.

CacheData – Παράδειγμα

- Τόσο ο κόμβος 6 όσο και ο 7 ζητούν το d_i μέσω του κόμβου 5. Ο κόμβος 5 αναγνωρίζει ότι το d_i είναι δημοφιλές και το τοποθετεί στη cache. Μελλοντικές αιτήσεις από τους κόμβους 3, 4, ή/και 5 μπορούν να εξυπηρετηθούν απευθείας από τον κόμβο 5.
- Υποθέστε ότι πολλές αιτήσεις για το di προωθούνται από το κόμβο 3. Οι κόμβοι 3, 4 και 5 μπορεί να νομίζουν ότι το di είναι δημοφιλές. Όμως, μεγάλο μέρος του χώρου της cache χάνεται εάν όλοι αποθηκεύσουν το di.

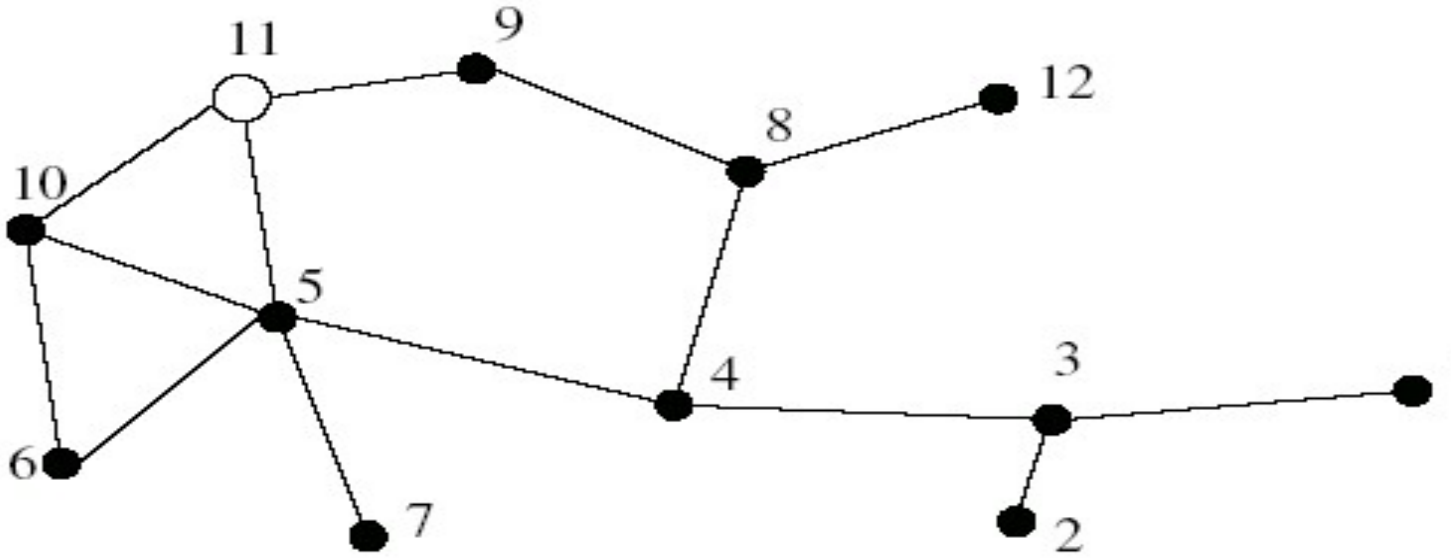


Fig. 1. An ad hoc network

CachePath

- ***Ένας κόμβος αποθηκεύει ένα μονοπάτι προς ένα γειτονικό κόμβο καθώς προωθεί τα δεδομένα και χρησιμοποιεί το μονοπάτι για να κάνει ανακατεύθυνση μελλοντικών αιτήσεων προς το γειτονικό (caching) κόμβο***.
  - Κάνοντας caching το μονοπάτι για κάθε ένα αντικείμενο, μπορούμε να μειώσουμε την καθυστέρηση πρόσβασης και το bandwidth επειδή το ζητούμενο αντικείμενο μπορεί να αποκτηθεί σε μικρότερο αριθμό από hops.
- *Βελτιώσεις:*
  - *Ένας ενδιάμεσος κόμβος μπορεί να αποθηκεύσει μόνο τον κόμβο προορισμό επειδή το μονοπάτι από τον τρέχον κόμβο (δρομολογητή) προς τον κόμβο προορισμό μπορεί να αποκτηθεί από το πρωτόκολλο δρομολόγησης.*
  - *Ο ενδιάμεσος κόμβος χρειάζεται να καταγράφει το μονοπάτι προς τα δεδομένα όταν ο* ***caching node βρίσκεται εγγύτερα από το data center****.*

CachePath

- Ένας κόμβος N_i caches το μονοπάτι προς τα δεδομένα μόνο όταν ο caching node, π.χ. N_j , είναι πολύ κοντά.
- ***Η εγγύτητα μπορεί να οριστεί σαν μια συνάρτηση της απόστασης από το data center και της απόστασης από το caching node***.

$$H_{save} = H(i,C) - H(i,j)$$

C=Data Center και j=caching node

- Η τιμή H_{save} πρέπει να είναι μεγαλύτερη από ένα κατώφλι που ορίζεται κατά την αρχικοποίηση του συστήματος.

CachePath

- Ένα μειονέκτημα του CachePath είναι ότι το cached μονοπάτι μπορεί να μην είναι αξιόπιστο και η χρήση του μπορεί να αυξήσει το επικοινωνιακό overhead.
- Ένα cached μονοπάτι μπορεί να μην είναι αξιόπιστο επειδή είτε το αντικείμενο έχει γίνει μη έγκυρο ή ο caching node δεν είναι προσβάσιμος.

HybridCache

- Το υβριδικό πρωτόκολλο **HybridCache** συνδυάζει τα θετικά των CacheData και CachePath, ενώ αποφεύγει τις αδυναμίες τους.
- Στο πρωτόκολλο HybridCache, όταν ένα κόμβος προωθεί ένα αντικείμενο, τότε αποθηκεύει στη cache είτε το αντικείμενο ή το μονοπάτι με βάση κάποια κριτήρια:
  - Μέγεθος του αντικειμένου s_i
  - Χρονικό όριο ζωής TTL_i
  - H_{save}

HybridCache

- Για ένα αντικείμενο d_i, οι παρακάτω ευριστικές τεχνικές χρησιμοποιούνται για να αποφασίσει ένας κόμβος εάν θα κάνει cache το αντικείμενο ή το μονοπάτι:
  - Εάν το s_i είναι **μικρό**, τότε το **CacheData** θα πρέπει να υιοθετηθεί επειδή το αντικείμενο χρειάζεται ένα πολύ μικρό μέρος της cache, διαφορετικά, CachePath θα πρέπει να χρησιμοποιηθεί για να εξοικονομηθεί χώρος στη cache. Η τιμή κατώφλι για το μέγεθος του αντικειμένου ορίζεται ως T_s.

HybridCache

- Εάν το $\boldsymbol{TTL_i}$ είναι **μικρό**, τότε το CachePath δεν είναι μια καλή επιλογή επειδή το αντικείμενο μπορεί να γίνει μη έγκυρο σύντομα. Χρησιμοποιώντας το CachePath μπορεί να έχει ως αποτέλεσμα την επιλογή λάθους μονοπατιού και εν' τέλει την επαναποστολή του query προς το data center. Έτσι, το **CacheData** θα πρέπει να χρησιμοποιηθεί. Εάν το TTL_i είναι μεγάλο, τότε πρέπει να χρησιμοποιηθεί το CachePath. Η τιμή κατώφλι για το *TTL* (system parameter) ορίζεται ως $\mathbf{T_{TTL}}$.

HybridCache

– Εάν η τιμή H_{save} είναι **μεγάλη**, τότε το **CachePath** είναι μια καλή επιλογή επειδή μπορεί να διασώσει ένα μεγάλο αριθμό από hops, διαφορετικά, το CacheData θα πρέπει να χρησιμοποιηθεί για να βελτιώσει την απόδοση εάν βέβαια υπάρχει αρκετός χώρος στη cache. Η τιμή κατώφλι για το H_{save} (system parameter) ορίζεται ως T_H.

HybridCache

(A) When a data item d_i arrives:
if (d_i is the requested data by the current node) **then**
cache data item d_i; return;
/* Data passing by */
if (an old version of d_i is in the cache) **then**
update the cached copy;
else if ($s_i < \mathcal{T}_s$ **or** there is an invalid copy in the cache
or there is a cached path for d_i) **then**
cache data item d_i;
else if ($H_{save} > \mathcal{T}_H$ **and** $TTL_i > \mathcal{T}_{TTL}$) **then**
cache the path of d_i;

(B) When cache replacement is necessary:
while (not enough free space **and**
there are invalid data items in the cache) **do**
Remove an invalid data item;
while (not enough free space) **do** /*still need space*/
Remove a valid data item;

(C) When a request for data item d_i arrives:
if (there is a valid copy in cache) **then**
send d_i to the requester;
else if (there is a valid path for d_i in the cache) **then**
forward the request to the caching node;
else
forward the request to the data center;