Προηγμένα Θέματα Αρχιτεκτονικής Συστημάτων

Θέματα Συνέπειας της Cache

Διδάσκων: Δημόκας Νικόλαος

Εισαγωγικά

- ***Η συνέπεια της cache ορίζεται ως βαθμός της διαφοράς μεταξύ του αυθεντικού δεδομένου που είναι αποθηκευμένο στην πηγή των δεδομένων και του cached αντιγράφου του.***

- Μία στρατηγική για τη διατήρηση της συνέπειας των δεδομένων που βρίσκονται στη cache μπορεί να επικεντρωθεί σε τρία σχεδιαστικά ζητήματα:
  - ***Επίπεδο Συνέπειας (Consistency Level)***
  - ***Έλεγχο Συνέπειας (Consistency Control)***
  - ***Διατήρηση της κατάστασης της cache (Cache Status Maintenance)***

Consistency Level (1/2)

- Υπάρχουν **3 επίπεδα συνέπειας** της cache:
  - **Ισχυρή** (SC – Strong Consistency)
  - **Δέλτα** (DC – Delta Consistency)
  - **Ασθενής** (WC – Weak Consistency)
- Η ισχυρή συνέπεια ονομάζεται επίσης και συνέπεια της τελευταίας τιμής. Κάθε κόμβος πελάτης θα χρησιμοποιήσει τα cached δεδομένα μόνο εάν αυτά δεν έχουν ενημερωθεί ποτέ από τον κόμβο πηγή από τη στιγμή που αποθηκεύτηκαν.
  - Το αντίθετο της ισχυρής συνέπειας είναι η ασθενής συνέπεια
- Εφαρμογές ισχυρής – ασθενής συνέπειας
  - Εφαρμογή διακίνησης μετοχών
  - On-line νέα

Consistency Level (2/2)

- Έστω V_i^t συμβολίζει την έκδοση ενός αντικειμένου D_i που βρίσκεται σε ένα κόμβο πηγή και T_i^t τη timestamp του D_i σε ένα κόμβο τη χρονική στιγμή t.
- Ο αριθμός έκδοσης = 0, όταν δημιουργείται το αντικείμενο αριθμός έκδοσης += 1 κάθε φορά που γίνεται ενημέρωση
- Εάν η **έκδοση του D_i** στο cache κόμβο **είναι επίκαιρη** και σύμφωνη με αυτή του κόμβου πηγή τότε όταν φθάνει στο cache κόμβο μία αίτηση για το D_i **ικανοποιείται η Ισχυρή Συνέπεια**.
- Εάν η έκδοση του D_i δεν είναι ποτέ **παλαιότερη από ένα χρονικό διάστημα Δ** σε σχέση με το αυθεντικό αντικείμενο τότε **ικανοποιείται η Δέλτα Συνέπεια**.
- Εάν η **έκδοση του D_i δεν** ανταποκρίνεται απαραίτητα στην πιο **ενήμερη** έκδοση του αυθεντικού αντικειμένου αλλά κάποια ορθή παλαιότερη τότε **ικανοποιείται η Ασθενής Συνέπεια**.

Consistency Control (1/3)

- Ο έλεγχος συνέπειας μπορεί να αρχικοποιηθεί τόσο από τον κόμβο πηγή όσο και από τους cache κόμβους.
  - **Source initiated**
  - **Cache initiated**

Consistency Control (2/3)

- ***Source Initiated***
  - Τα μηνύματα μη εγκυρότητας εκπέμπονται σε όλους τους cache κόμβους για να γνωστοποιηθεί η κατάσταση των αντικειμένων
  - Συνεπάγεται ότι ο φόρτος εργασίας για τη διατήρηση της συνέπειας της cache τον επωμίζεται ο κόμβος πηγή
  - Μπορεί να θεωρηθεί μια **push** λειτουργία καθώς ο κόμβος πηγή ωθεί τα μηνύματα των ενημερώσεων στους cache κόμβους

Consistency Control (3/3)

- ***Cache Initiated***
  - Οι cache κόμβοι στέλνουν μηνύματα στους κόμβους πηγές για να διαπιστώσουν εάν τα αντικείμενα είναι έγκαιρα.
  - Συνεπάγεται ότι ο φόρτος εργασίας για τη διατήρηση της συνέπειας της cache τον επωμίζεται ο cache κόμβος
  - Μπορεί να θεωρηθεί μια **pull** λειτουργία καθώς κάθε cache κόμβος ανακτά τη πληροφορία εγκυρότητας από τους κόμβους πηγές.

Cache status maintenance (1/3)

- Ο κόμβος πηγή (ή διαφορετικά ο server) έχει τη δυνατότητα είτε να καταγράφει την κατάσταση των δικό του cache κόμβων είτε όχι.
  - **Stateful**
  - **Stateless**

Cache status maintenance (2/3)

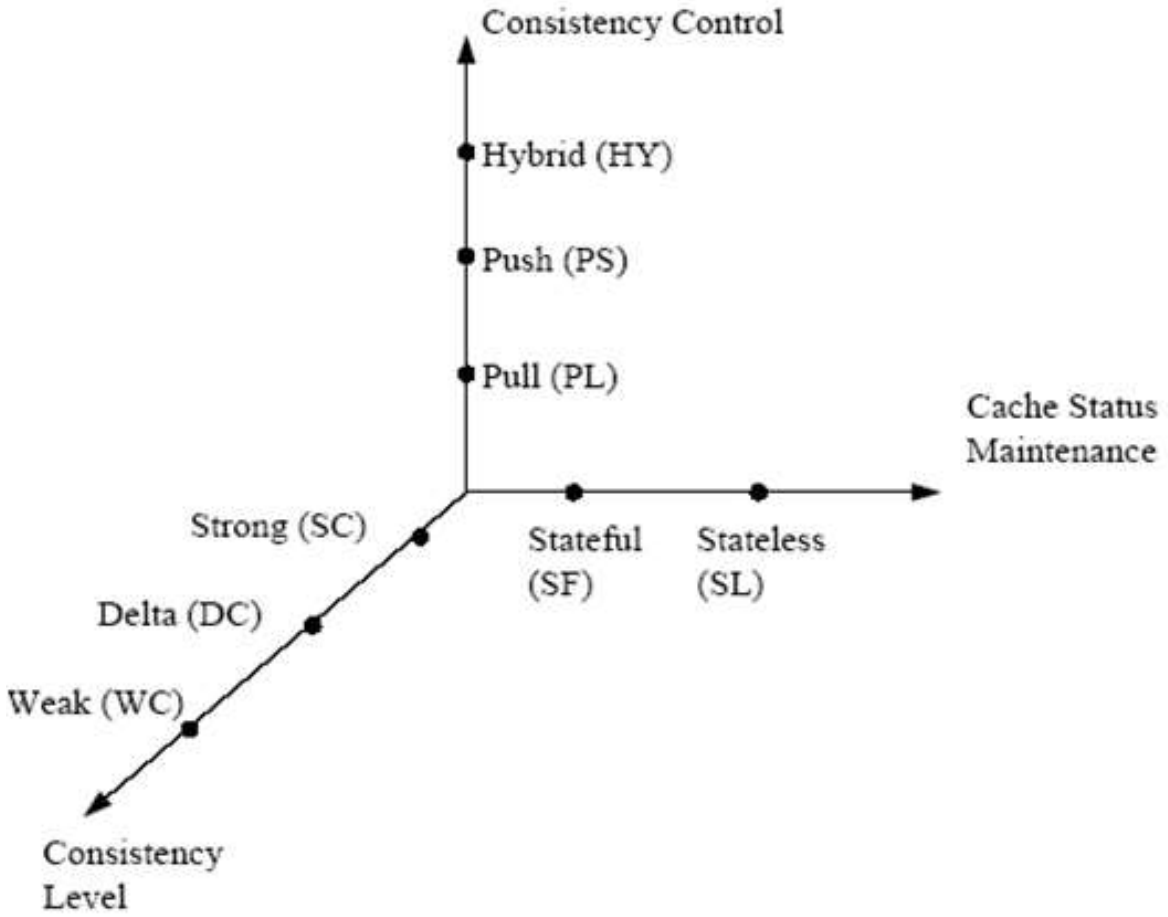
- **Stateful**
  - Ο κόμβος πηγή καταγράφει ποιοι κόμβοι έχουν αποθηκεύσει ποια αντικείμενα.
    - καταγράφει δεδομένα σχετικά με το ιστορικό των προσπελάσεων και ενημερώσεων των αντικειμένων
  - Πλεονέκτημα: Όταν ο κόμβος πηγή θέλει να επικοινωνήσει με τους cache κόμβους μπορεί να στείλει είτε unicast είτε multicast μηνύματα ➔ Μειώνει τη χρήση του εύρους ζώνης.
  - Μειονέκτημα: Ο κόμβος πηγή πρέπει να καταγράφει την κατάσταση των caches των cache κόμβων ➔ Μη επεκτάσιμη μέθοδος σε πολλούς κόμβους

Cache status maintenance (3/3)

- **Stateless**
  - Ο κόμβος πηγή δεν απαιτείται να γνωρίζει τη κατάσταση των cache κόμβων.
  - Ο κόμβος πηγή αποθηκεύει το ιστορικό ενημερώσεων και **σε περιοδικά διαστήματα πλημμυρίζει το δίκτυο με ακυρωτικά μηνύματα**.
  - Μειονέκτημα: Επιβαρύνει σημαντικά την επικοινωνία καθώς αυξάνεται το πλήθος των μηνυμάτων που διαδίδονται στο δίκτυο.

Τεχνικές

- Οι τεχνικές διατήρησης της cache μπορούν να χωρισθούν ανάλογα με την ακολουθούμενη στρατηγική ως προς τα 3 σχεδιαστικά ζητήματα.
- Γενικότερα μπορούν να κατηγοριοποιηθούν σε:
  - **push-based**
  - **pull-based**



Τεχνική – Pull Each Read

SL – PL – SC

- Περιλαμβάνει την αποστολή ενός μηνύματος ελέγχου εγκυρότητας στον αντίστοιχο κόμβο πηγή κάθε φορά που το ζητά κάποιος πελάτης.
  - Εάν το αντικείμενο είναι «φρέσκο» τότε στέλνεται κατευθείαν στον πελάτη.
- Πλεονέκτημα: Η cache ποτέ δεν αποστέλλει μη «φρέσκα» δεδομένα
- Μειονέκτημα:
  - Δημιουργεί μεγάλη δικτυακή κίνηση
  - Μεγάλη καθυστέρηση πρόσβασης των δεδομένων από τους πελάτες, στην περίπτωση που τα αντικείμενα δεν έχουν τροποποιηθεί.
  - Επιβάλλει μεγάλο φόρτο εργασίας στους cache κόμβους (εξατίας του Pull) και στις πηγές (εξαιτίας του strong consistency)

Τεχνική – Pull with timeout period

SL – PL – DC

- Τα cached δεδομένα παραμένουν έγκαιρα για μια χρονική περίοδο τουλάχιστον t, αφού ο κόμβος έχει επιβεβαιώσει τα δεδομένα.
  - Όταν t=0, η τεχνική ισοδυναμεί με την Pull each read.
- **Για μεγάλο t → αυξάνεται η πιθανότητα** οι cache κόμβοι να παρέχουν **μη «έγκυρα» δεδομένα** (θα είναι το πολύ t χρονικό διάστημα παλιά).
- **Για μικρό t →** ελαττώνουμε το ποσοστό των μη φρέσκων δεδομένων, αλλά **αυξάνονται τα μηνύματα ελέγχου** εγκυρότητας που ταξιδεύουν στο δίκτυο.

Push based Τεχνικές (1/2)

- Το caching μπορεί να ελαττώσει τις απαιτήσεις σε εύρος ζώνης στα κινητά δίκτυα
- Η χρήση του caching ***απαιτεί μια πολιτική ακύρωσης των δεδομένων της cache*** (cache invalidation strategy) για να εγγυηθεί την εγκυρότητα των δεδομένων της.
- Μπορούμε να χρησιμοποιήσουμε μια "***Αναφορά Ακύρωσης***" (**Invalidation Report, IR**) για να διατηρήσουμε την εγκυρότητα των δεδομένων του κινητού χρήστη
  - Ο server εκπέμπει περιοδικά invalidation reports

Push based Τεχνικές (2/2)

- Κάθε κινητός πελάτης, εάν είναι ενεργός, ακούει τις αναφορές και ακυρώνει τα σχετικά δεδομένα του
- Όμως, εξαιτίας των περιορισμών σε ενέργεια (μπαταρία), ένας “κινητός” υπολογιστής συχνά λειτουργεί σε doze ή αποσυνδεδεμένο τρόπο λειτουργίας
- Συνεπάγεται ότι ο κινητός υπολογιστής μπορεί να “χάσει” μερικά invalidation reports, με συνέπεια να αναγκαστεί να “πετάξει” όλα τα περιεχόμενα της cache του, όταν “ξυπνήσει”

Σχήμα IR

Το σχήμα

- Step1: Όταν ο χρήστης κάνει κάποιες αιτήσεις για αντικείμενα, ο κινητός υπολογιστής κρατά, τις **αιτήσεις σε μια ουρά**
- Step2: Όταν ο κινητός υπολογιστής λάβει μια invalidation report που εκπέμπεται από τον server, **ακυρώνει** όποια **δεδομένα της cache** υποδεικνύονται από την **invalidation report**
- Step3: ***Μετά την ακύρωση, ο κινητός υπολογιστής απαντά στις αιτήσεις της ουράς***

Σχήμα IR

Το σχήμα

- Step4: Εάν τα δεδομένα της αίτησης βρίσκονται στην cache, θα προωθηθούν στην εφαρμογή του χρήστη από την cache.
- Step5: Εάν τα δεδομένα της αίτησης δεν βρίσκονται στην cache, ο κινητός υπολογιστής θα αιτηθεί για τα δεδομένα αυτά στον server

Κατηγοριοποίηση IR

- Κατηγοριοποίηση των IR με διαφορετικά κριτήρια
  - *Πώς στέλνει ο server τα IR?*
    - **Ασύγχρονα (Asynchronous)**
      - Ο server εκπέμπει ένα μήνυμα ακύρωσης (invalidation message) για ένα αντικείμενο αμέσως μόλις αλλάξει η τιμή του αντικειμένου
    - **Σύγχρονα (Synchronous)**
      - Όταν τα IR εκπέμπονται περιοδικά
  - *Πώς οργανώνεται η πληροφορία στα IR?*
    - **Μη Συμπιεσμένα (Uncompressed)**
      - Οι αναφορές περιέχουν πληροφορία για κάθε αντικείμενο ξεχωριστά
    - **Συμπιεσμένα**
      - Οι αναφορές περιέχουν συνολική πληροφορία για υποσύνολα των αντικειμένων

Στόχοι

- Ελάττωση του Transformation Cost
  - Ελάττωση του μεγέθους της IR
  - Βελτιστοποίηση της δομής της IR
  - Να κάνουμε τους πελάτες να μην χάνουν “πολλή πληροφορία”, όταν είναι σε doze ή disconnected λειτουργία
- Το transformation cost περιλαμβάνει το IR transformation cost και το data transformation cost.
  - IR transformation cost: Το μέγεθος της IR που αποστέλλεται στους πελάτες
  - Data transformation cost: Τα δεδομένα που πρέπει να γίνουν download από το server, όταν τα αιτούμενα δεδομένα δεν είναι στη cache
- Ο επόμενος τύπος είναι
  - Stateless, Symmetric, Asynchronous

Ορολογία

- L: ο server εκπέμπει μια IR κάθε L secs
- w: invalidation broadcast window
- T_i: το τρέχον timestamp
- T_{lb}: το timestamp της πιο πρόσφατης invalidation report που λήφθηκε από τον κινητό πελάτη (MU)
- o_j: το id του αντικειμένου
- t_j: το αντίστοιχο timestamp της πιο πρόσφατης αλλαγής / τροποποίησης του αντικειμένου
- t_j^c: το timestamp της cache για το αντικείμενο j
- IR: invalidation report

Μέθοδος Broadcasting Timestamps

ΕΠΕΞΕΡΓΑΣΙΑ

- Ο sever εκπέμπει την IR η οποία πειρέχει μια λίστα U_i που ορίζεται ως ακολούθως για τη χρονική στιγμή T_i.
- U_i= {[o_j ,t_j] : o_j ϵ D (database) και t_j είναι το ***timestamp*** της τελευταίας ***ενημέρωσης*** του o_j τέτοιο ώστε T_i-w×L ≤ t_j ≤ T_i}
- Ο MU καταγράφει τα [o_j, t_j^c] όλων των αντικειμένων της cache του, όπου o_j ϵ D (database) and t_j^c είναι το timestamp της cache του για το o_j
- Ο MU κρατά επίσης το T_{lb} και μια λίστα Q_i που περιλαμβάνει τα αντικείμενα που έχουν ζητηθεί
  - Ο MU ακυρώνει τα αντικείμενα στην cache σύμφωνα με την IR
  - Μετά την ακύρωση, ο MU απαντά στις αιτήσεις των εφαρμογών

Drop ολόκληρη τη cache ή όχι

T_{lbA} : : αγνοούμε όλη την cache

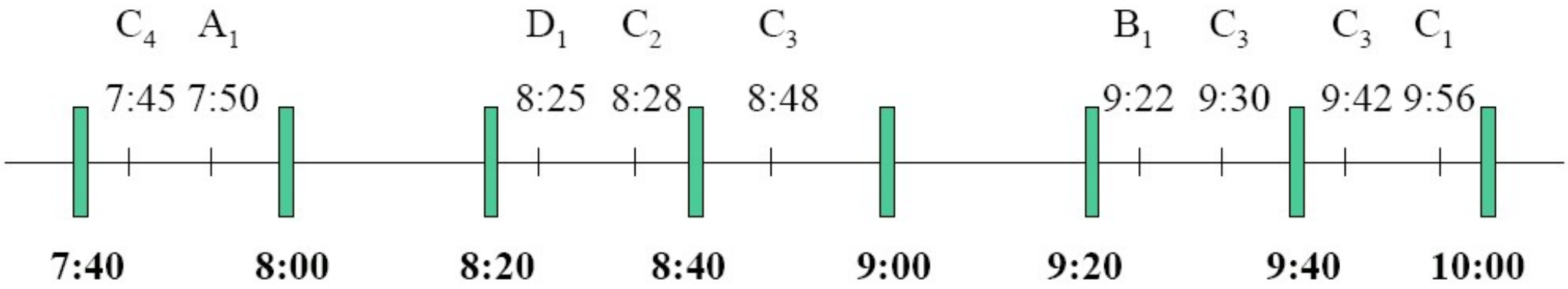
T_{lbB} : : ο MU συγκρίνει τα $[o_j, t_j^c]$ στην cache του με τα $[o_j, t_j]$ στην U_i για να αποφασίσει εάν θα διατηρήσει στην cache του το o_j ή όχι

Αλγόριθμος Broadcasting Timestamps

if ($Ti-T_{lb} > w \times L$) {drop the entire cache}
else{
for every item o_j in the MU cache
{if there is a pair[o_j,t_j] in Ui {
if $t_j^c < t_j$ {
throw o_j out of the cache}
else {$t_j^c = Ti$}
}}}
for every item $o_j \in Qi$
{
if o_j is in the cache
{ use the cache's value to answer the query }
else
{ go uplink with the query }
$T_{lb}:=Ti$ }

Παράδειγμα (1/5)

Όλα τα ενημερωμένα αντικείμενα



- L = 20 min
- w = 3
- Ti = 10:00

Παράδειγμα (2/5)

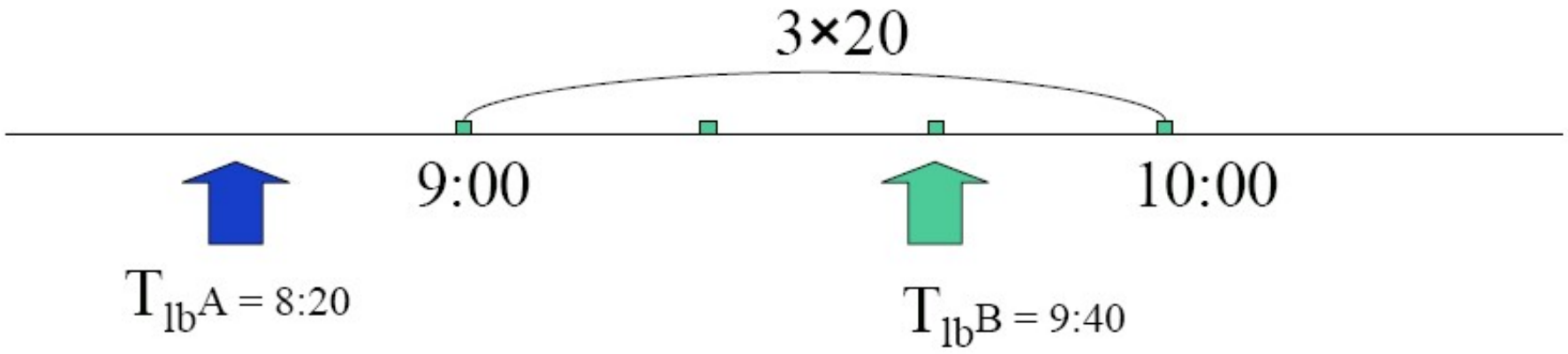
- Ο sever εκπέμπει τα ενημερωμένα αντικείμενα για το διάστημα 9:00 μέχρι 10:00

B_1	C_3	C_1
9:22	9:42	9:56

- Q: τα αντικείμενα που ζητήθηκαν από 9:40 μέχρι 10:00

E_1	C_3

Παράδειγμα (3/5)



$T_{lb}A$: Αρχική MU cache

T_{lb}	A_1	A_2	A_3	A_4	B_1	B_2	B_3	B_4
8:20	7:50	6:15	6:25	6:35	6:22	6:45	6:55	7:05
	C_1	C_2	C_3	C_4	D_1	D_2	D_3	D_4
	7:15	7:28	7:30	7:40	7:28	7:25	7:35	7:40

Προκύπτουσα MU cache:T_{lb}: 10:00 $\Rightarrow$ no item στην cache

Παράδειγμα (4/5)

$T_{lb}{}^{B}$: Αρχική MU cache

T_{lb}	A_1	A_2	A_3	A_4	B_1	B_2	B_3	B_4
9:40	7:50	6:15	6:25	6:35	9:22	6:45	6:55	7:05
	C_1	C_2	C_3	C_4	D_1	D_2	D_3	D_4
	7:15	8:28	9:30	7:40	8:25	7:25	7:35	7:40

Προκύπτουσα MU cache

T_{lb}	A_1	A_2	A_3	A_4	**B_1**	B_2	B_3	B_4
9:40	7:50	6:15	6:25	6:35	**10:00**	6:45	6:55	7:05
	C_1	C_2	**C_3**	C_4	D_1	D_2	D_3	D_4
	7:15	8:28	**9:30**	7:40	8:25	7:25	7:35	7:40

A,D is not in IR→no change

C_1,C_3 are in IR and $t_j{}^c < t_j \rightarrow$ throw C_1, C_3

B_1 : is in IR and $t_j{}^c >= t_j \rightarrow t_j{}^c = Ti$

Παράδειγμα (5/5)

- Απάντηση αιτήσεων

T_{lb}	A_1	A_2	A_3	A_4	B_1	B_2	B_3	B_4		
10:00	7:50	6:15	6:25	6:35	10:00	6:45	6:55	7:05		
		C_2	**C_3**	C_4	D_1	D_2	D_3	D_4	C_3	**E_1**
		8:28	**9:42**	7:40	8:25	7:25	7:35	7:40	9:42	**6:00**

C_3, E_1 : δεν είναι στην cache, αλλά είναι στην ουρά αιτήσεων

→uplink στον server