Κινητός και Διάχυτος Υπολογισμός (Mobile & Pervasive Computing)

Δημήτριος Κατσαρός

Χειμώνας 2015

Διάλεξη 9η

Περιεχόμενα

- Αρχιτεκτονική δικτύου
- **Συνέπεια της cache (Cache Consistency)**

Σχήματα Cache coherency (1/2)

- Η γενική μέθοδος των Invalidation Reports
- Σχήματα No-Checking Caching
    1. Broadcasting Timestamps
    2. Amnesic Terminals
    3. Bit-Sequences
- Σχήματα Checking Caching
    1. Simple-checking caching scheme
    2. Simple-grouping caching scheme
    3. Grouping with cold update-set report

Σχήματα Cache coherency (2/2)

- Selective cache invalidation
    1. Group-based Cache Invalidation
    2. Hybrid Cache Invalidation
    3. Selective Cache Invalidation

Εισαγωγικά (1/3)

- Το caching μπορεί να ελαττώσει τις απαιτήσεις σε εύρος ζώνης στα κινητά δίκτυα
- Αφού χρησιμοποιήσουμε το caching, απαιτείται 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
- Για να εγγυηθούμε την εγκυρότητα, ο server περιοδικά(?) εκπέμπει invalidation reports

Εισαγωγικά (2/3)

- 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
- Ως αποτέλεσμα αυτού, ο κινητός υπολογιστής μπορεί να “χάσει” μερικές invalidation reports, με συνέπεια να αναγκαστεί να “πετάξει” όλα τα περιεχόμενα της cache του, όταν “ξυπνήσει”

Εισαγωγικά (3/3)

- Μπορούμε να κατηγοριοποίησουμε τον server
  - Stateful server
    - Ο server γνωρίζει ποια δεδομένα είναι cached από ποιους πελάτες
  - Stateless Server
    - Ο server δεν γνωρίζει την “κατάσταση” της cache των κινητών πελατών, αλλά ούτε και την κατάσταση του ίδιου του πελάτη, δηλ., εάν είναι αποσυνδεδεμένος ή σε ποια θέση βρίσκεται

Σχήμα IR

- Το σχήμα
  - Όταν ο χρήστης κάνει κάποιες αιτήσεις για αντικείμενα, ο κινητός υπολογιστής κρατά, τις αιτήσεις σε μια ουρά
  - Όταν ο κινητός υπολογιστής λάβει μια invalidation report που εκπέμπεται από τον server, θα ακυρώσουν όποια δεδομένα της cache υποδεικνύονται από την invalidation reports
  - Μετά την ακύρωση, ο κινητός υπολογιστής απαντά στις αιτήσεις της ουράς
  - Εάν τα δεδομένα της αίτησης βρίσκονται στην cache, θα προωθηθούν στην εφαρμογή του χρήστη από την cache.
  - Εάν τα δεδομένα της αίτησης δεν βρίσκονται στην cache, ο κινητός υπολογιστής θα κάνει την αίτηση για τα δεδομένα αυτά στον server

Κατηγοριοποίηση IR

- Μπορούμε να κατηγοριοποιήσουμε τις IR σύμφωνα με διαφορετικά κριτήρια, ως ακολούθως
  - **Πώς στέλνει ο server τις IR?**
    - Ασύγχρονα (Asynchronous)
      - Ο server εκπέμπει ένα μήνυμα ακύρωσης (invalidation message) για ένα αντικείμενο αμέσως μόλις αλλάξει η τιμή του αντικειμένου
    - Σύγχρονα (Synchronous)

  Όταν οι IR εκπέμπονται περιοδικά
  - **Πώς οργανώνεται η πληροφορία στην IR?**
    - Συμπιεσμένα (Uncompressed)
      - Οι αναφορές περιέχουν πληροφορία για κάθε αντικείμενο ξεχωριστά
    - Συμπιεσμένα
      - Οι αναφορές περιέχουν συνολική πληροφορία για υποσύνολα των αντικειμένων

Στόχοι

- Ελάττωση του Netware 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 Netware transformation cost περιλαμβάνει το IR transformation cost και το data transformation cost.
  - IR transformation cost: ποσότητα της IR που αποστέλλεται στους πελάτες
  - Data transformation cost: Οι ποσότητες των δεδομένων που πρέπει να γίνουν downloaded από τον server, όταν τα επερωτούμενα δεδομένα δεν είναι στην cache
- Ο επόμενος τύπος είναι
  - Stateless, Symmetric, Asynchronous

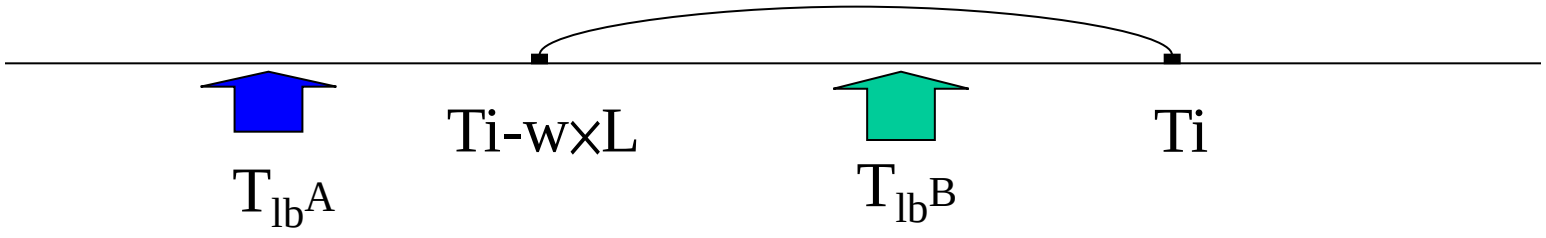
Στρατηγικές NO-Checking Caching

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

Μέθοδος Broadcasting Timestamps

- ΕΠΕΞΕΡΓΑΣΙΑ
  - Ο sever εκπέμπει την IR η οποία πειρέχει μια λίστα Ui που ορίζεται ως ακολούθως για τη χρονική στιγμή Ti = iL.
  - Ui = {[o_j,t_j] :o_j ∈ D (database) και t_j είναι το timestamp της τελευταίας ενημέρωσης του o_j τέτοιο ώστε Ti-w×L ≤ t_j ≤ Ti}
  - Ο MU καταγράφει τα [o_j,t_j^c] όλων των αντικειμένων της cache του, όπου o_j ∈ 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 Qi που ορίζεται ως ακολούθως:
  - Qi={$o_{j:}$: o_j έχει ζητηθεί στο διάστημα [Ti-1,Ti]}
  - Ο MU ακυρώνει τα αντικείμενα στην cache σύμφωνα με την IR
  - Μετά την ακύρωση, ο MU απαντά στις αιτήσεις των εφαρμογών

Drop ολόκληρη την cache ή όχι



T_{lb}^A : : αγνοούμε όλη την cache

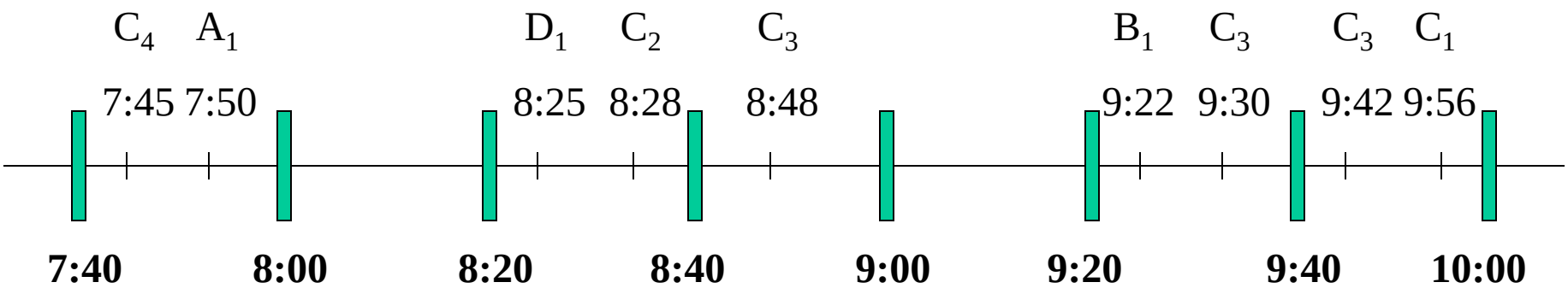
T_{lb}^B : : ο MU συγκρίνει τα $[o_j,t_j^c]$ στην cache του με τα $[o_j,t_j]$ στην U_i για να αποφασίσει εάν θα διατηρήσει στην cache του το o_j ή όχι

➢ ΑΛΓΟΡΙΘΜΟΣ 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)

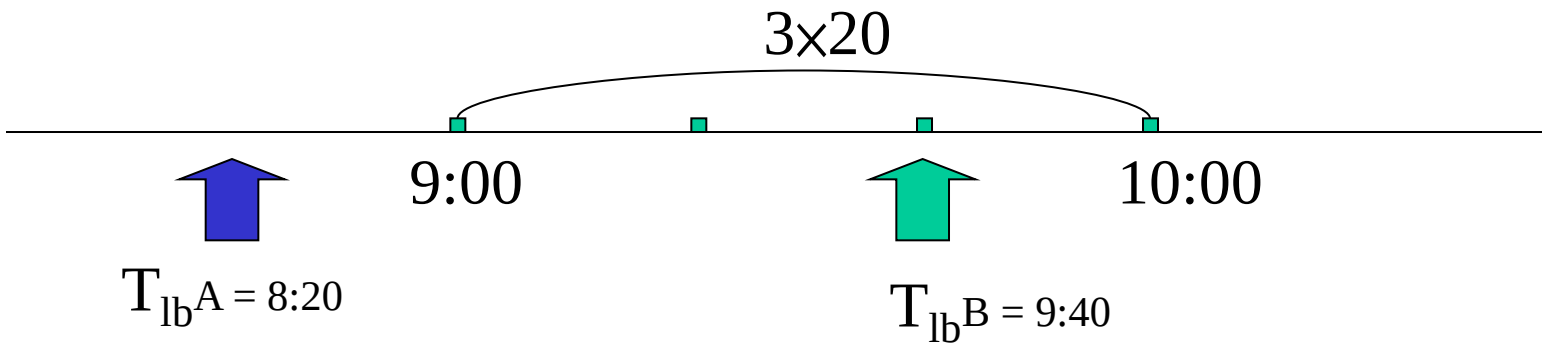
- Ο sever εκπέμπει τα ενημερωμένα αντικείμενα για το διάστημα 9:00 μέχρι 10:00

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

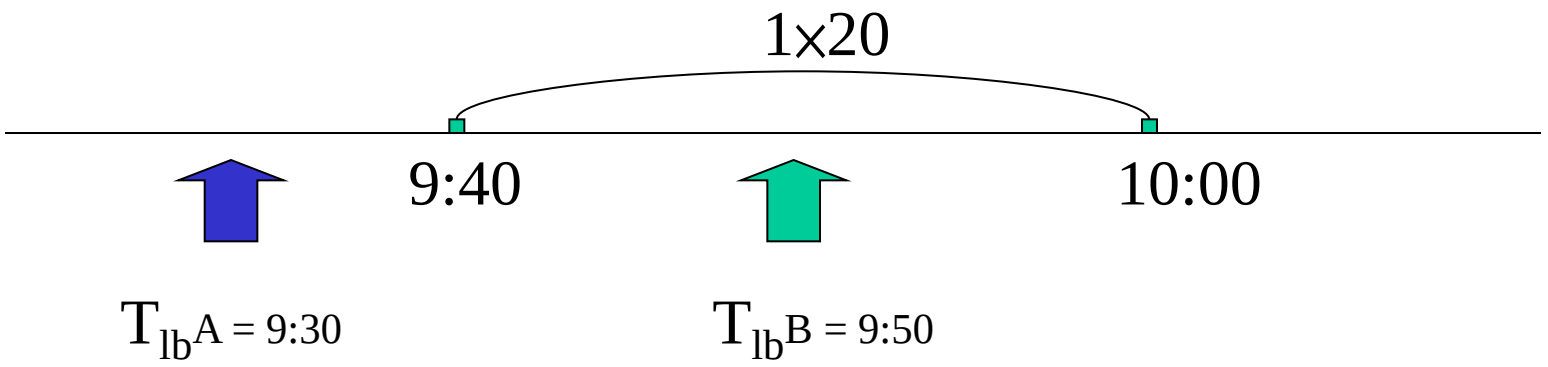
→uplink στον server

$T_{lb} \rightarrow$ Ti

Μέθοδος Amnesic Terminals

- ΕΠΕΞΕΡΓΑΣΙΑ
  - Ο sever εκπέμπει μόνο τους προσδιοριστές των αντικειμένων που τροποποιήθηκαν μετά την τελυταία invalidation report, δηλ., w = 1
  - Ui = {o_j :o_j ∈ D (database) και η τελευταία ενημέρωση του o_j συνέβη τη στιγμή t_j τέτοια ώστε $T_{i-1} \leq t_j \leq T_i$}
  - Ο MU κρατά το T_{lb} και μια λίστα Qi που ορίζεται ως ακολούθως:
  - Qi={o_j:o_j έχει ζητηθεί στο διάστημα [Ti-1,Ti]}
  - Ο MU ακυρώνει τα αντικείμενα στην cache σύμφωνα με την IR
  - Μετά την ακύρωση, ο MU απαντά στις αιτήσεις των εφαρμογών

Drop ολόκληρη την cache ή όχι



$T_{lb}A$: αγνοούμε όλη την cache

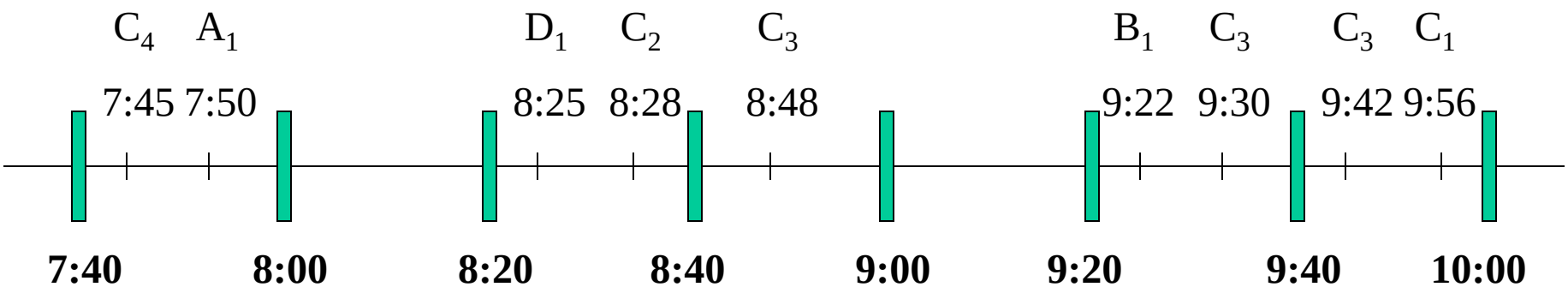
$T_{lb}B$: Εάν ένα cached item αναφέρεται, τότε ο MU το διώχνει από την cache του

➢ ΑΛΓΟΡΙΘΜΟΣ AMNESIC TERMINALS

if (T_i-T_{lb} > L) {drop the entire cache}
else{
for every item o_j in the MU cache
{if o_j in Ui {
throw o_j out of the cache}
}}
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 (1/3)

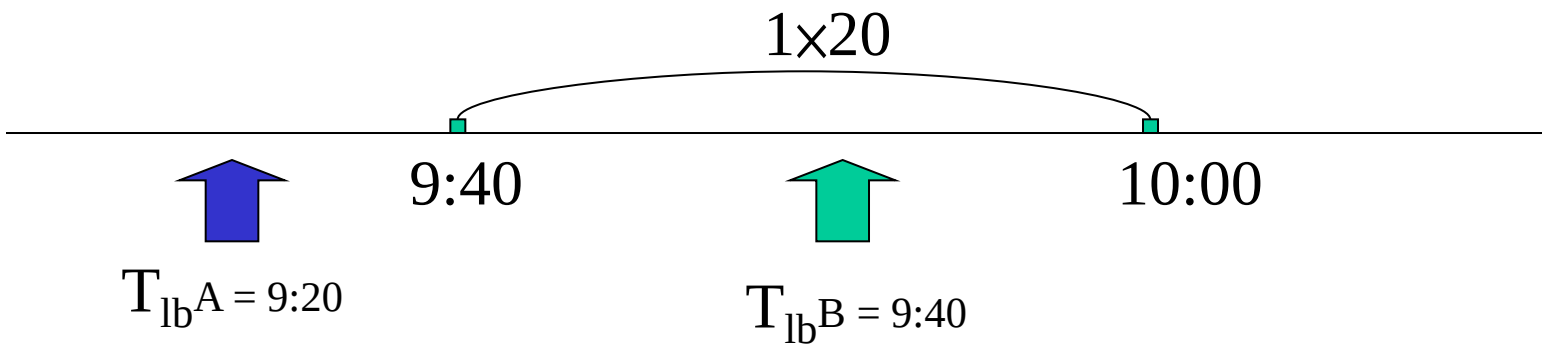
- Όλα τα ενημερωμένα αντικείμενα



- L = 20 min
- Ti = 10:00
- Ο sever εκπέμπει τα ενημερωμένα αντικείμενα από 9:40 μέχρι 10:00

C_3	C_1

Παράδειγμα (2/3)



$T_{lb}A$: Αρχική MU cache

T_{lb}	A_1	A_2	A_3	A_4	B_1	B_2	B_3	B_4
9:20	C_1	C_2	C_3	C_4	D_1	D_2	D_3	D_4

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

Παράδειγμα (3/3)

$T_{lb}B$: Αρχική MU cache

T_{lb}	A_1	A_2	A_3	A_4	B_1	B_2	B_3	B_4
9:40	C_1	C_2	C_3	C_4	D_1	D_2	D_3	D_4

Προκύπτουσα MU cache

T_{lb}	A_1	A_2	A_3	A_4	B_1	B_2	B_3	B_4
10:00	**C_1**	C_2	**C_3**	C_4	D_1	D_2	D_3	D_4

A, B, D : are not in IR→no change

C_1, C_3 : are in IR → throw C_1, C_3

$T_{lb} \rightarrow T_i$

Περιεχόμενα

- **Συνέπεια της cache (Cache Consistency)**
  - **BitSequences**

Σχήμα Bit-Sequences

- Παρατηρήστε ότι
  - Γενικά, υπάρχει ένα tradeoff μεταξύ του μεγέθους και της αποτελεσματικότητας των εκπεμπόμενων αναφορών
  - Στο επόμενο σχήμα, θα αντιμετωπίσουμε το πρόβλημα της βελτιστοποίησης του μεγέθους των εκπεμπόμενων αναφορών

Τεχνικές βελτιστοποίησης

- Ονοματισμός των bit_sequences (bit_sequences naming)
- Συσσώρευση των ενημερώσεων (Update aggregation)
- Ιεραρχική δόμηση των bit-sequences.

Εισαγωγή στις Bit-Sequences (1/2)

- Η ΔΟΜΗ
  - Η IR αποτελείται από ένα σύνολο ακολουθιών bit (bit sequences), κάθε μια από τις οποίες έχει το αντίστοιχο timestamp
  - Κάθε bit αναπαριστά ένα αντικείμενο της database
  - Ένα “1” bit σε μια ακολουθία σημαίνει ότι το αντικείμενο που αναπαρίσταται από το bit αυτό έχει τροποποιηθεί μετά το χρόνο που καθορίζεται από το timestamp της ακολουθίας (sequence)
  - Ένα “0” bit σε μια ακολουθία σημαίνει ότι το αντικείμενο δεν έχει τροποποιηθεί από τη στιγμή του timestamp
  - Το σύνολο των ακολουθιών οργανώνεται περαιτέρω σε μια ιεραρχική δομή, με την B_n στη δομή να έχει N bits, τα οποία αντιστοιχούν στα N αντικείμενα της βάσης

Εισαγωγή στις Bit-Sequences (2/2)

- Το πολύ μισά από τα bits στη B_n μπορούν να τεθούν στην τιμή “1” μετά $TS(B_n)$
- Η επόμενη ακολουθία, που συμβολίζεται με B_{n-1}, στη δομή έχει N/2 bits
- Το k-οστό bit στη B_{n-1} αντιστοιχεί στο k-οστό “1” bit στη B_n
- $N/2^2$ bits μπορούν να τεθούν στην τιμή “1” αφότου $TS(B_{n-1})$
- Η ιεραρχική δομή περιέχει B_k ($k=1,\ldots,n$, $2^n = N$)
- Μια επιπλέον dummy ακολουθία B_0 χρησιμοποιείται, όπου $TS(B_0)$ συμβολίζει το χρόνο μετά τον οποίο κανένα αντικείμενο δεν έχει τροποποιηθεί

➢ ΑΛΓΟΡΙΘΜΟΣ BIT-SEQUENCES

if $TS(B_0) \leq T_{lb}$

κανένα αντικείμενο της cache δεν ακυρώνεται και ο αλγόριθμος τερματίζεται

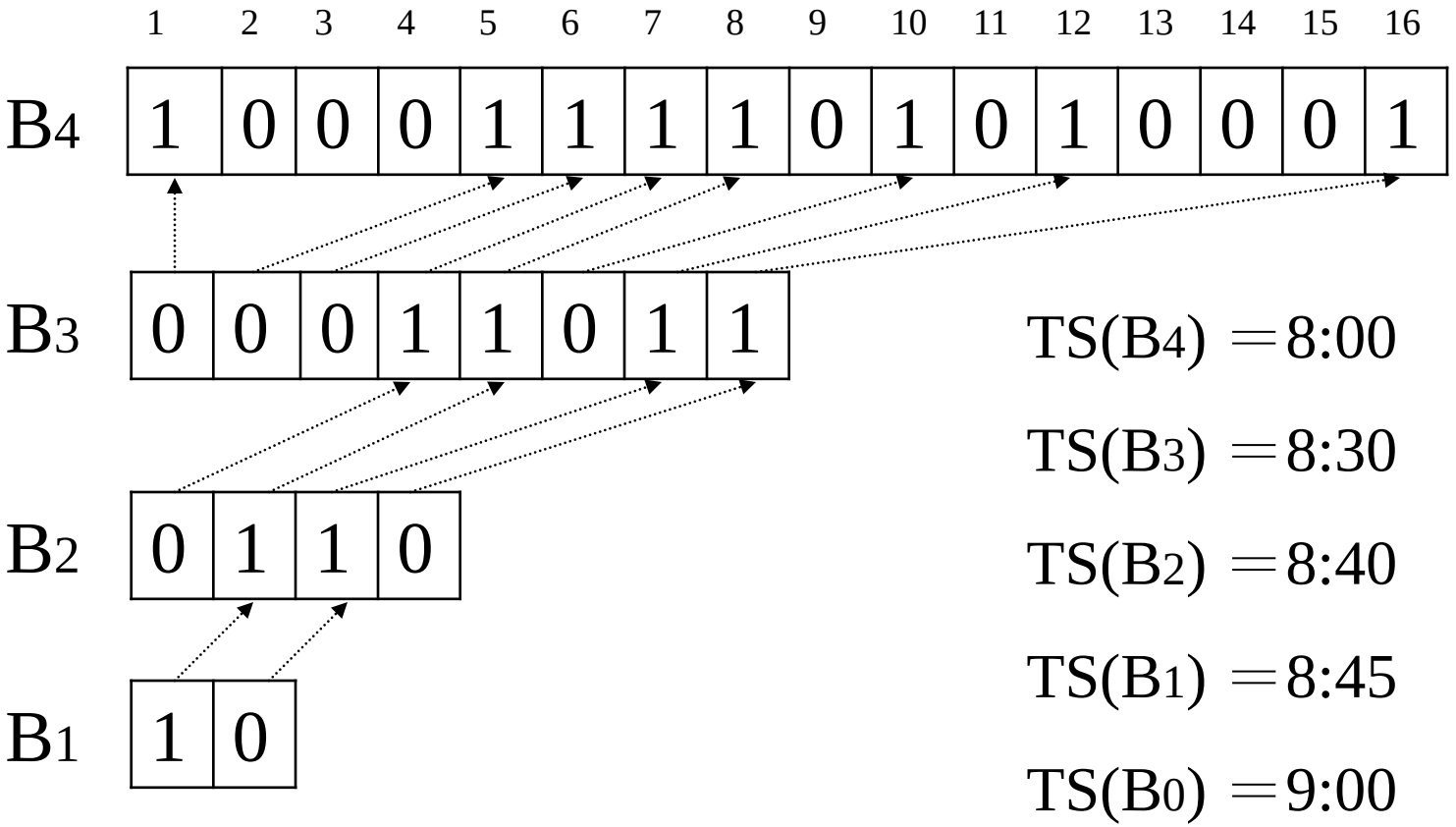
if $T_{lb} < TS(B_n)$

ολόκληρη η cache ακυρώνεται και ο αλγόριθμος τερματίζεται

Εντοπίζουμε την bit sequence B_j με timestamp $TS(B_j) \leq T_{lb} < TS(B_{j-1}) \; \forall \; j \; (1 \leq j \leq n)$

Ακυρώνουμε όλα τα αντικείμενα που αναπαρίστανται με “1” bits στην B_j

Παράδειγμα Bit-Sequences (1/2)



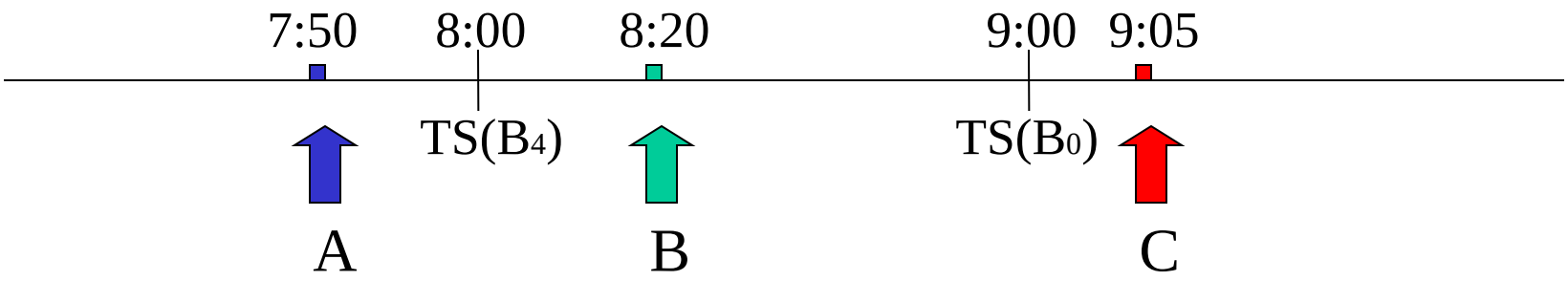
$TS(B_4) = 8:00$

$TS(B_3) = 8:30$

$TS(B_2) = 8:40$

$TS(B_1) = 8:45$

$TS(B_0) = 9:00$



Original cache：1 、2、 4、 7、 11、 15

•Περίπτωση A

$T_{lb} \le TS(B_4)$ ⟹ όλη η cache ακυρώνεται

•Περιπτωση B

$TS(B_4) \le T_{lb} \le TS(B_3)$ (invalidate with B4)

⟹ η προκύπτουσα cache μετά την τροποποίηση

2、4、11、15 .

•Περίπτωση C

$T_{lb} \ge TS(B_0)$ ⟹ καμία αλλαγή