

## INFORMATION RETRIEVAL

## • ESERCIZIO 2

• Rsync  $F_{old} = \text{"ABRACADABRA"} \text{ (client)}$   $B = 3$   
 $F_{new} = \text{"ABRACABRIA"} \text{ (server)}$

• Partendo dal file  $F_{old}$  sul client, suddivido il file da aggiornare in blocchi di 3 char:

$h_1 = \text{"ABR"} \quad h_2 = \text{"ACA"} \quad h_3 = \text{"DAB"} \quad h_4 = \text{"RAS"}$

• I blocchi  $\{h_1, h_2, h_3, h_4\}$  vengono inviati al server. Eseguendo uno scanning su  $F_{new}$ , si verifica quali elementi  $\{h_1, \dots, h_4\}$  sono presenti in  $F_{new}$ . Se è presente un elemento  $h_i$ , viene saltato nello scanning e sono saltati i caratteri dell'intero blocco e si continua lo scanning per i successivi caratteri

CLIENT		SERVER
$\{h_1, h_2, h_3, h_4\}$	→	<u>ABR</u> <u>ACA</u> <u>BRIA</u> $h_1 \quad h_2 \quad h_3$

• Viene generata una codifica di  $F_{new}$  generata dai blocchi trovati nello scanning di  $F_{new}$  e i caratteri mancanti. La codifica viene inviata dal server al client, così da ricostruire e aggiornare  $F_{old}$

SERVER		CLIENT
$h_1 h_2 \dots h_1 IA$	→	Aggiorno $F_{old}$

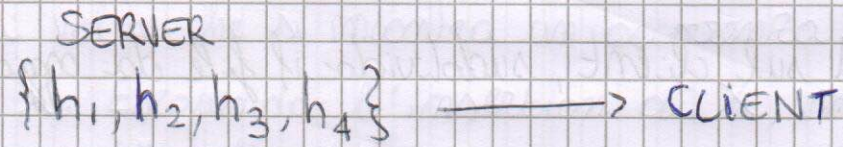
• ~~I~~ Gli elementi  $\{h_1, h_2, h_3, h_4\}$  sono ~~le~~ funzioni hash dei caratteri nel blocco

- Zsync

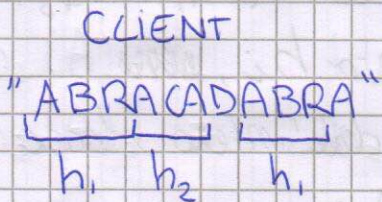
- F\_new sul server viene suddiviso in blocchi di dimensione = 3:

$$F_{\text{new}} \quad h_1 = \text{"ABR"} \quad h_2 = \text{"ACA"} \quad h_3 = \text{"LAB"} \quad h_4 = \text{"RIA"}$$

- Viene eseguito un hashing sui blocchi come per rsync. Gli hash vengono inviati dal server al client.



- Viene eseguito uno scanning su F\_old per identificare quali elementi di {h<sub>1</sub>, h<sub>2</sub>, h<sub>3</sub>, h<sub>4</sub>} sono presenti. A differenza di rsync, lo scanning è eseguito carattere per carattere

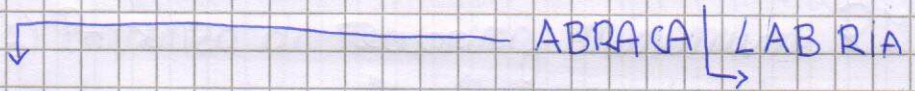
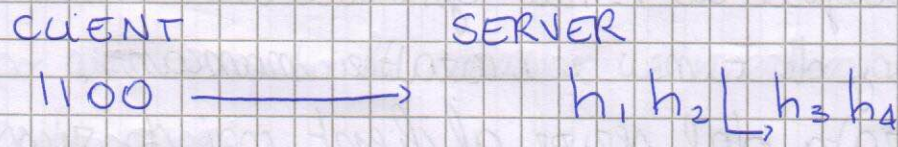


- Viene generata una bitmask per indicare quali h<sub>i</sub> sono presenti in F\_old

b = 1100

1 2 3 4

- Il client invia la bitmask al server il quale esegue l'algoritmo di compressione degli elementi h<sub>3</sub> e h<sub>4</sub> mancanti, basandosi su h<sub>1</sub> e h<sub>2</sub> presenti



<0,0,< <del></del></p>

<7,3,1>

<4,1,\$>

- h<sub>3</sub> e h<sub>4</sub> compressi vengono inviati al client che decomprime e ordina gli elementi così da ricostruire F\_new

ESERCIZIO 3

NODE	REFERENCE	OUTDEGR	COPYLIST	COPY BLOCK	EXTRA
14	0	9			A
15	1	10	CL	CB	E

↳ compressione basata su 1 NODO precedente (14)

A: extra nodes di 14 = LISTA di 74  
3, 10, 12, 13, 14, 17, 19, 21, 25

CL: copy list di 15, ovvero una bitmask con cardinalità = |LISTA 14|  
dove vengono settati a 1 gli elementi comuni delle liste di 14 e 15

14:	3	10	12	13	14	17	19	21	25				
15:	↓	5	10	11	12	↓	14	17	19	21	24	↓	33
		↓	↓	↓	↓	↓	↓	↓	↓	↓		↓	
CL =	0	1	1	0	1	1	1	1	1	0			

CL = 011011110

E: i nodi di 15 non in comune con la lista 14  
E = {5, 11, 24, 33}

CB: copy block del nodo 15 rispetto alla sua copylist.  
Il primo bit è lo stesso della copylist. Successivamente si calcola la frequenza dei bit consecutivi uguali della copylist

	0	1	1	0	1	1	1	1	0
Freq:	1	2	1	4	1				

• Il copyBlock CB è codificato con il primo bit = primo bit della copylist, seguito dalle FREQUENZE - 1

CB = 0 | (1-1) | (2-1) | (1-1) | (4-1) | (1-1)

primo bit —

= 001030 — ultimo bit: possibile ometterlo conoscendo la CARDINALITÀ (OUTDEGREE) della lista